Administration de systèmes UNIX
Formation ARS 2010 – 2011

Partie 1

Thierry Besançon

Formation Permanente de l'Université
Pierre et Marie Curie

Chapitre 1

UNIX : généralités, historique

Un système d'exploitation est une application comme une autre **en gros**.
Par exemple, noyau LINUX version REDHAT 2.6.18-164 :

- 43682 fichiers dont
  - 18205 fichiers « `.c` »
  - 17699 fichiers « `.h` »
  - 1805 fichiers « `.S` »

Les missions d'un système d'exploitation sont :

- mise à disposition de ressources matérielles : espace disque, temps d'exécution sur le microprocesseur central, espace mémoire, etc.
- partage équitable de ces ressources entre les utilisateurs pour atteindre le but de système multi-utilisateurs

◇ Constructeurs de hardware

Marque	*Site web*	*Version d'UNIX*
APPLE	`http ://www.apple.com`	MacOS X 10.x
CRAY	`http ://www.cray.com`	Unicos ?. ?
HP **+ COMPAQ** **+ DIGITAL**	`http ://www.hp.com` `http ://www.digital.com`	HP-UX 11.x Tru64 Unix 5.x
IBM	`http ://www.ibm.com`	AIX 5.x
SGI	`http ://www.sgi.com`	IRIX 6.x.y
SUN	`http ://www.sun.com`	Solaris 10 OpenSolaris 10

Ces UNIX ne sont pas interchangeables :

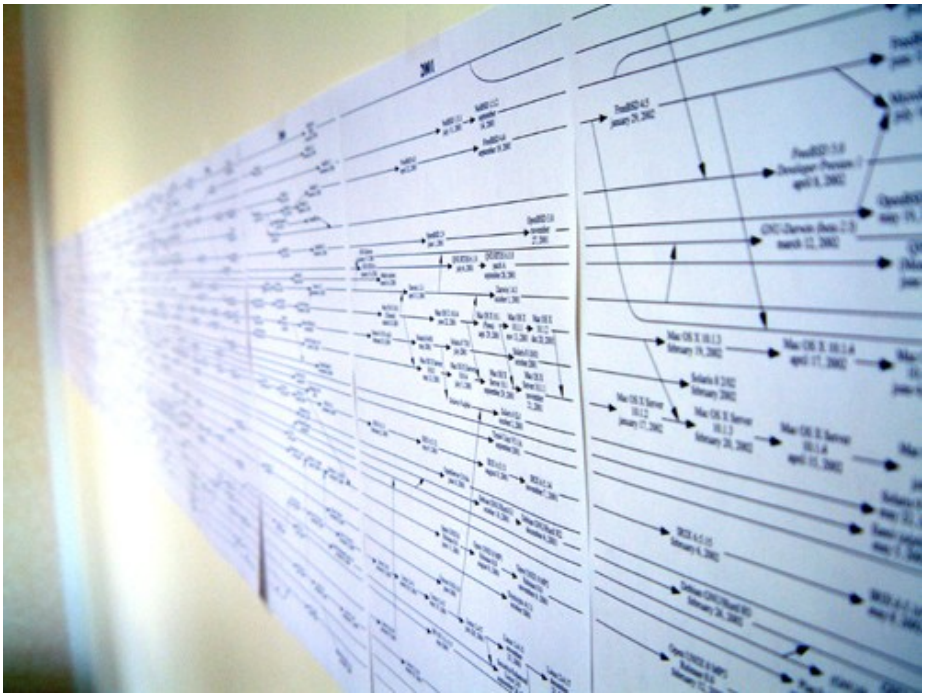
- Solaris pour machines de marque SUN
- MacOS X pour machines de marque APPLE
- AIX pour machines de marque IBM
- etc.

◇ Non constructeurs de hardware

Marque	*Site web*	*Version d'UNIX*
SANTA CRUZ	`http ://www.sco.com`	Unixware 7.x
LINUX	`http ://www.kernel.org`	noyau 2.6.x
FREEBSD	`http ://www.freebsd.org`	FreeBSD 8.x
NETBSD	`http ://www.netbsd.org`	NetBSD 1.x
OPENBSD	`http ://www.openbsd.org`	OpenBSD 3.x

Principales plateforme matérielle : le « PC » (les autres plateformes sont anecdotiques).

L'arbre généalogique d'UNIX est très complexe. Cf
`http://www.levenez.com/unix/`



Se reporter à l'annexe pour un schéma détaillé.

Chapitre 1 • UNIX : généralités, historique
§1.3 • Les différentes familles d'UNIX

Du point de vue de l'utilisateur, les divers UNIX se ressemblent beaucoup.
Du point de vue de l'administration, les divers UNIX ont chacun des spécificités (les commandes liées au hardware varient, on trouve des extensions propres à chaque constructeur). En pratique, l'administrateur attend toujours.
Plusieurs tentatives d'unification :

- *System V Interface Definition* de AT&T (SVID, SVID2, SVID3 en 1989)
- IEEE POSIX (POSIX1003.1 en 1990)
- X/OPEN Portability Guide (XPG4 en 1993) du consortium X/OPEN (créé en 1984)

Mais...

Il reste 2 grandes familles d'UNIX issues d'un schisme :

- la famille **System V** avec notamment la dernière version connue sous le nom de **System V release 4** (alias *SVR4*)
- la famille **BSD** issue de l'université de Berkeley (BSD ≡ *Berkeley Software Distribution*)

Votre rôle : connaître les principes et les mécanismes d'UNIX afin de savoir s'adapter à n'importe quel UNIX.

Ken Thompson et Dennis M. Ritchie, les parents d'UNIX
On notera les teletypes 33 !

Il existe beaucoup de distributions LINUX car LINUX n'est la propriété de personnes mais de toute la communauté de programmeurs informatiques.

Les principales distributions sont :

- Red Hat, `http://www.redhat.com`
- Centos (clone de Red Hat), `http://www.centos.org`
- Suse, `http://www.suse.com`
- OpenSuse (clone de Suse), `http://www.opensuse.org`
- Mandriva, `http://www.mandriva.com`
- Debian, `http://www.debian.org`
- Ubuntu, `http://www.ubuntu.com`
- gentoo, `http://www.gentoo.com`
- Knoppix, `http://www.knoppix.org`

La salle de TP de la Formation Permanente est équipée de PC sous Mandriva (information non confirmée au moment de l'écriture de ce support de cours) et les TP se feront sur des machines virtuelles fonctionnant sous le logiciel VMWARE ou sous le logiciel VIRTUAL BOX.

Vos interlocuteurs (dans cet ordre décroissant d'importance) :

- Vassiliki Spathis
  email : `Vassiliki.Spathis@formation.jussieu.fr`
  Responsable des formations, elle informera les autres techniciens des interventions à réaliser.
- adresse email « `assistance` »
  email : `assistance@formation.jussieu.fr`

ATTENTION : votre compte informatique le 1er octobre de l'an prochain.

Vous **devez** signaler :

- tout problème de compte ; **faire au plus vite si vous soupçonnez que votre compte est piraté.**
- problème sur les imprimantes : cartouche d'encre vide, bourrage papier, etc. ;
- problème sur le poste de travail : terminal en mode inhabituel, clavier cassé, souris hors service, etc. ;
- problème anormal avec un logiciel : le logiciel ne fonctionne plus comme d'habitude, un logiciel a disparu, le logiciel ne fonctionne pas du tout comme le précise la documentation, etc. ;

Vous devez signaler les problèmes comme un patient donne ses symptômes à un médecin.
Un jour prochain, c'est à vous que les utilisateurs signaleront les problèmes. Alors mettez-vous à notre place dès maintenant en adoptant une attitude d'administrateur système : précision, détails, etc.

Chapitre 1 • UNIX : généralités, historique

§1.6 • Annexe 1

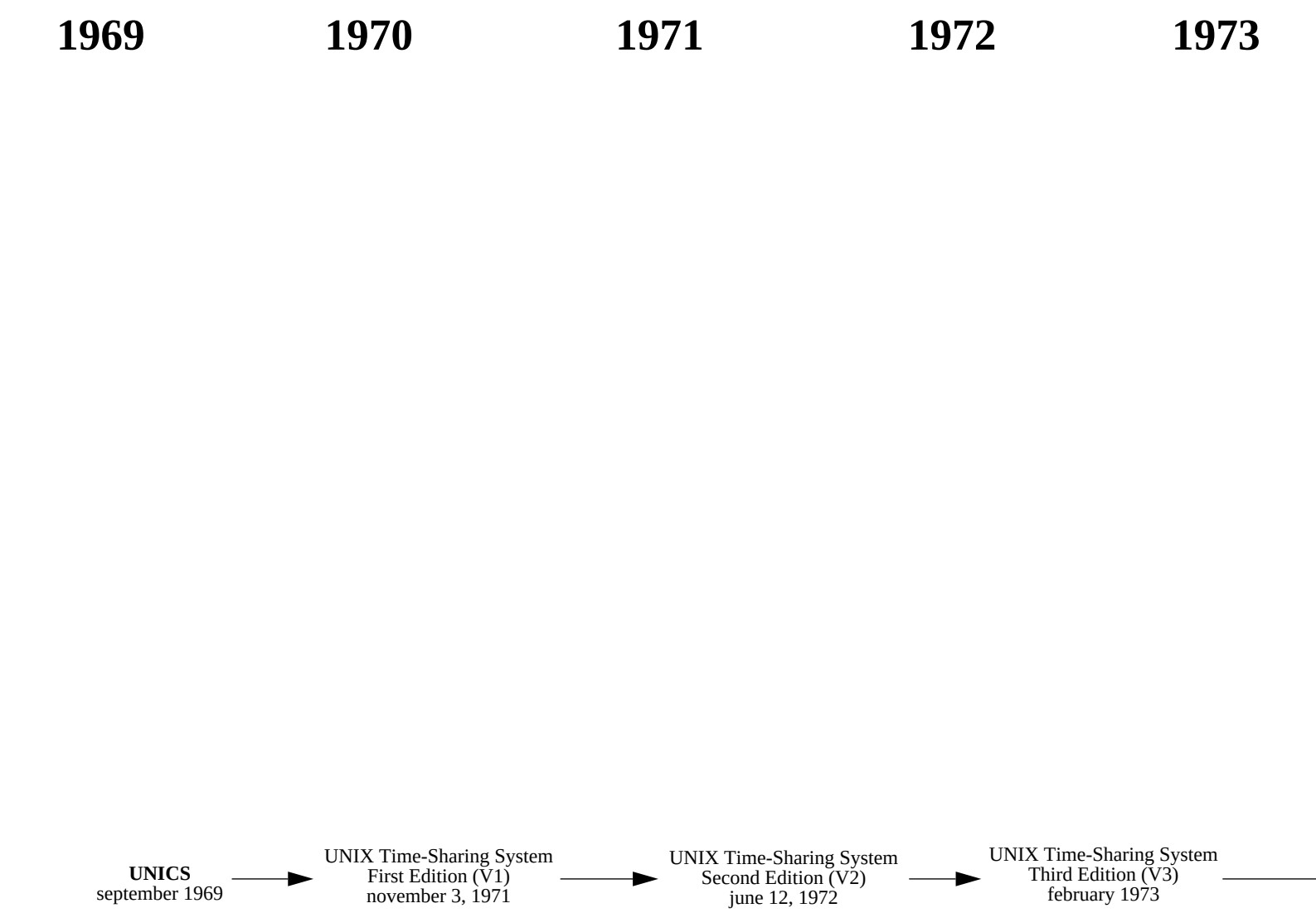
Ci joint dans la version imprimée de ce cours, un index des différentes version d'UNIX.

Adresse web du document fourni : « `http://www.levenez.com/unix/` »

1978 1BSD
1979 2BSD
1980 3BSD
1980 4BSD
1994 4.4BSD Lite 1
1995 4.4BSD Lite 2
1992 386 BSD
1986 A/UX
1989 Acorn RISC iX
1988 Acorn RISC Unix
1990 AIX
2000 AIX 5L
1989 AIX PS/2
1990 AIX/370
1989 AIX/6000
1991 AIX/ESA
1986 AIX/RT
1990 AMiX
1995 AOS Lite
1992 AOS Reno
2007 AppleTV
1994 ArchBSD
1991 ASV
1989 Atari Unix
1989 BOS
1979 BRL Unix
1988 BSD Net/1
1991 BSD Net/2
1991 BSD/386
1992 BSD/OS
1978 CB Unix
1986 Chorus
1988 Chorus/MiX
1983 Coherent
1987 CTIX
1984 CXOS
1999 Darwin
2000 Debian GNU/Hurd
1995 DEC OSF/1 ACP
1989 Dell Unix
2005 DesktopBSD
1995 Digital Unix
2003 DragonFly BSD
1984 Dynix
1993 Dynix/ptx
2003 ekkoBSD
1977 Eunice
2004 FireFly BSD
1993 FreeBSD
2006 FreeDarwin
1986 GNU
2001 GNU-Darwin
2005 Gnuppix GNU/Hurd-L4
1987 HPBSD
1983 HP-UX
2000 HP-UX 11i
1991 HP-UX BLS
1988 IBM AOS
1985 IBM IX/370
1985 Interactive 386/ix
1978 Interactive IS
2007 iPhone OS X
2007 iPod OS X
1983 IRIS GL2
1986 IRIX
1991 Linux
1994 Lites
1977 LSX
1999 Mac OS X
1999 Mac OS X Server
1985 Mach
1974 MERT
2002 MicroBSD
1977 Mini Unix
1984 Minix
2005 Minix 3
2000 Minix-VMD
1985 MIPS OS RISC/os
2002 MirBSD
1996 Mk Linux
1998 Monterey
1988 more/BSD
1983 mt Xinu
1993 MVS/ESA OpenEdition
1993 NetBSD
1988 NeXTSTEP
1987 NonStop-UX
1994 Open Desktop
2001 Open UNIX 8
1995 OpenBSD
2003 OpenDarwin
1995 OpenServer 5
2005 OpenSolaris
1996 OPENSTEP
1996 OS/390 OpenEdition
1997 OS/390 Unix
1990 OSF/1
2005 PC-BSD
1982 PC/IX
1986 Plan 9
1982 Plurix
2007 PureDarwin
1977 PWB
1974 PWB/UNIX
1984 QNX
2001 QNX RTOS
1996 QNX/Neutrino
1981 QUNIX
1997 ReliantUnix
1997 Rhapsody
1991 RISC iX
1977 RT
1994 SCO UNIX
2002 SCO UnixWare 7
1984 SCO Xenix
1987 SCO Xenix System V/386
2001 Security-Enhanced Linux
2004 Silver OS
1983 Sinix
1995 Sinix ReliantUnix
1990 Solaris 1
1992 Solaris 2
1982 SPIX
1982 SunOS
2004 Triance OS
1999 Tru64 Unix
1995 Trusted IRIX/B
1998 Trusted Solaris
1991 Trusted Xenix
1977 TS
1981 Tunis
1980 UCLA Locus
1979 UCLA Secure Unix
1988 Ultrix
1984 Ultrix 32M
1982 Ultrix-11
1986 Unicos
1996 Unicos/mk
2002 Unicos/mp
1993 Unicox-max
1969 UNICS
1981 UniSoft UniPlus
1979 UNIX 32V
1991 UNIX Interactive
1981 UNIX System III
1982 UNIX System IV
1983 UNIX System V
1984 UNIX System V Release 2
1986 UNIX System V Release 3
1988 UNIX System V Release 4
1985 UNIX System V/286
1986 UNIX System V/386
1971 UNIX Time-Sharing System
1993 UnixWare
1998 UnixWare 7
1976 UNSW
1977 USG
1982 Venix
1980 Xenix OS
1984 Xinu
1998 xMach
2001 z/OS Unix System Services

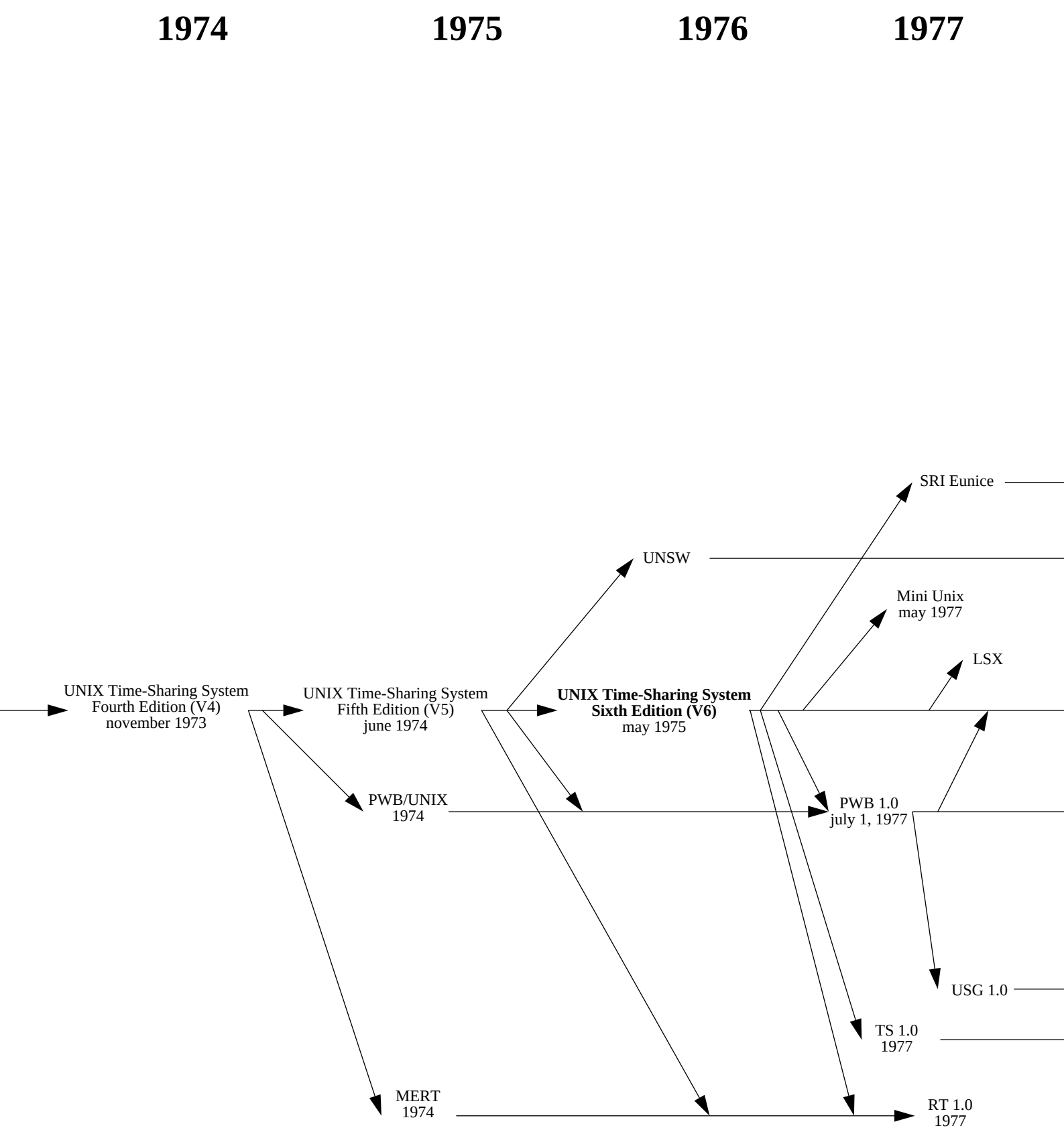
Ci joint dans la version imprimée de ce cours, un historique des différentes version d'UNIX.

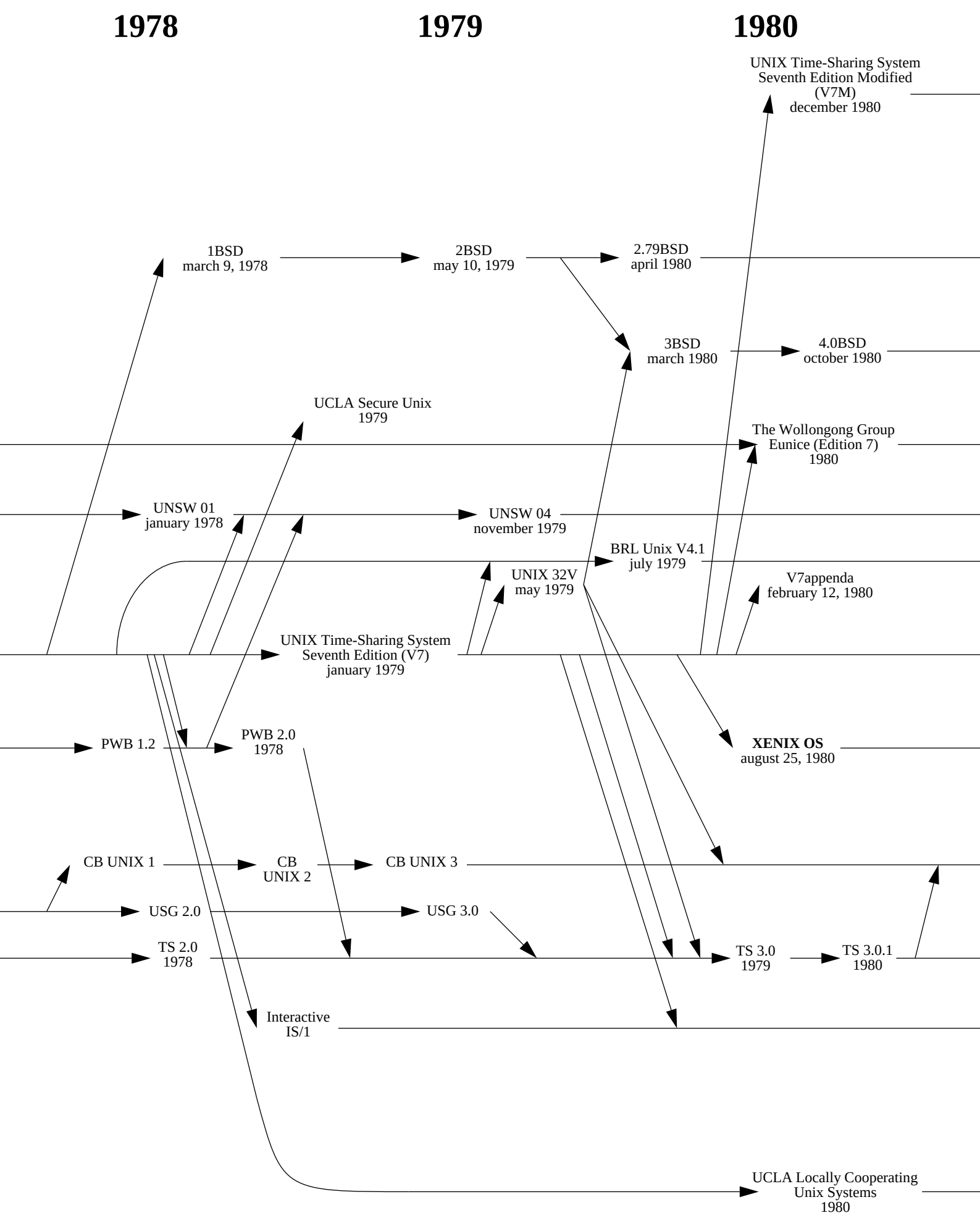
Adresse web du document fourni : « `http://www.levenez.com/unix/` »



Open Systems
december 27, 2009

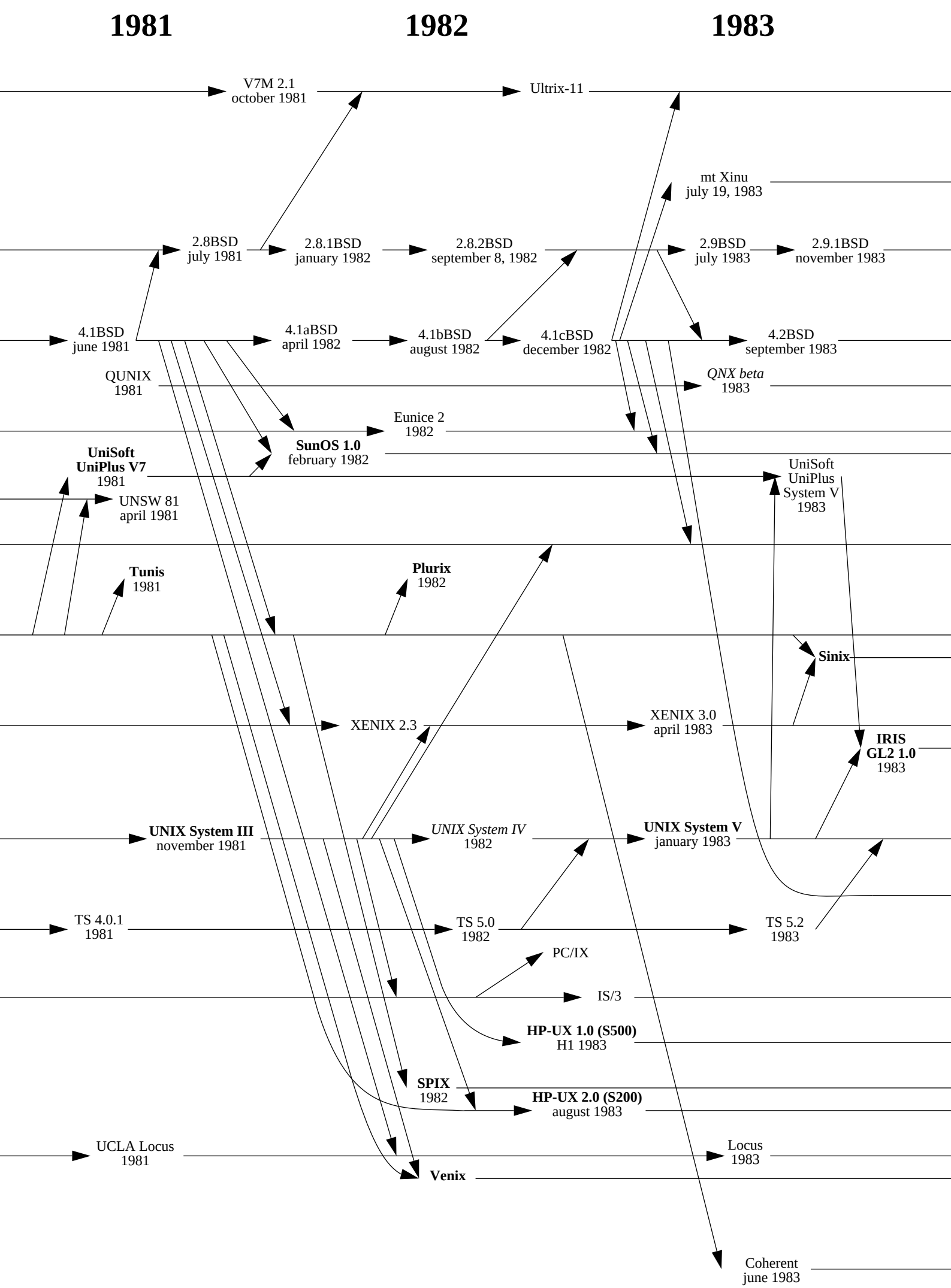
<http://www.levenez.com/unix/>

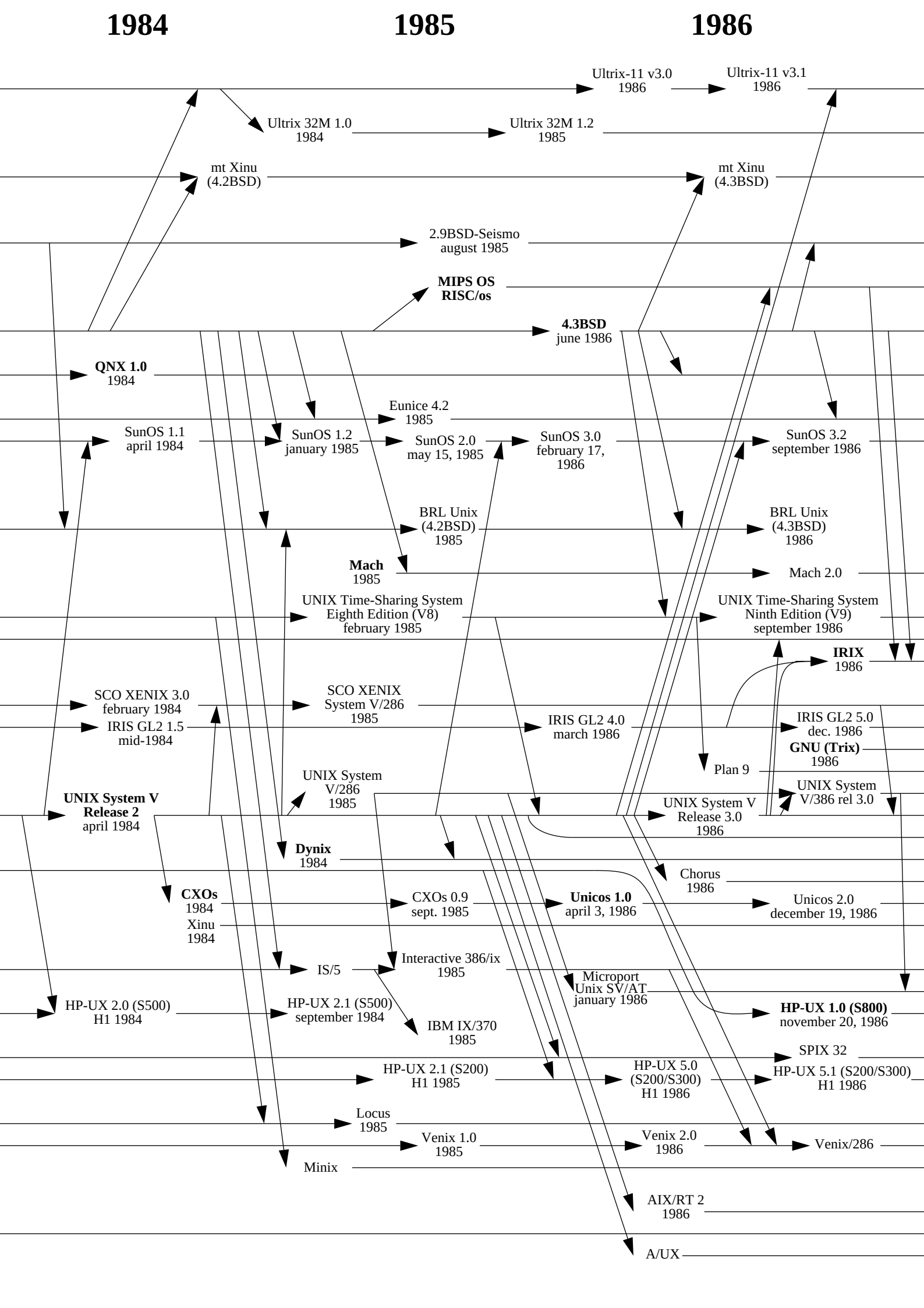

1974
1975
1976
1977
UNIX Time-Sharing System
Fourth Edition (V4)
november 1973
UNIX Time-Sharing System
Fifth Edition (V5)
june 1974
UNSW
SRI Eunice
Mini Unix
may 1977
LSX
UNIX Time-Sharing System
Sixth Edition (V6)
may 1975
PWB/UNIX
1974
PWB 1.0
july 1, 1977
USG 1.0
TS 1.0
1977
MERT
1974
RT 1.0
1977

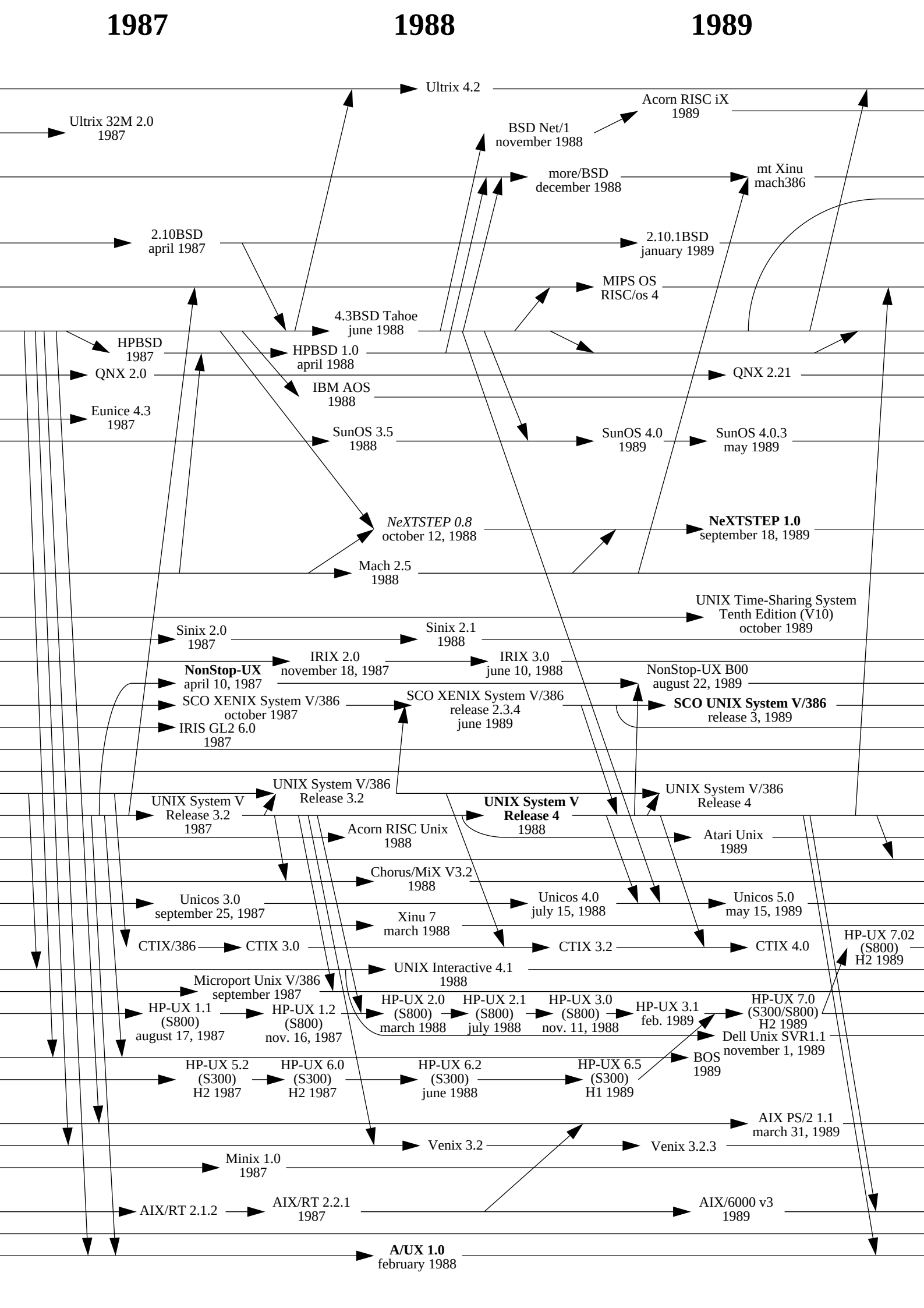


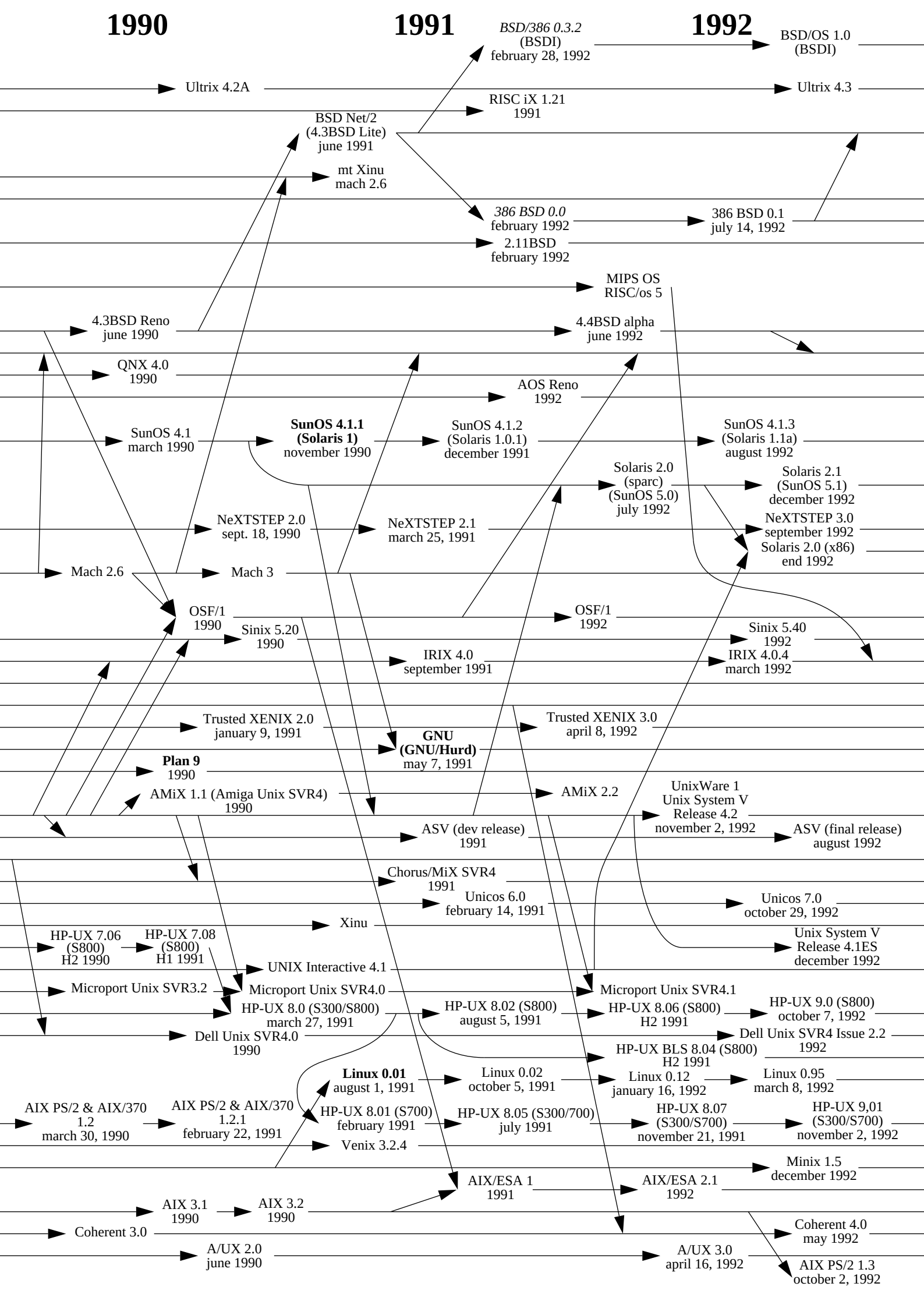
Note 1 : an arrow indicates an inheritance like a compatibility, it is not only a matter of source code.

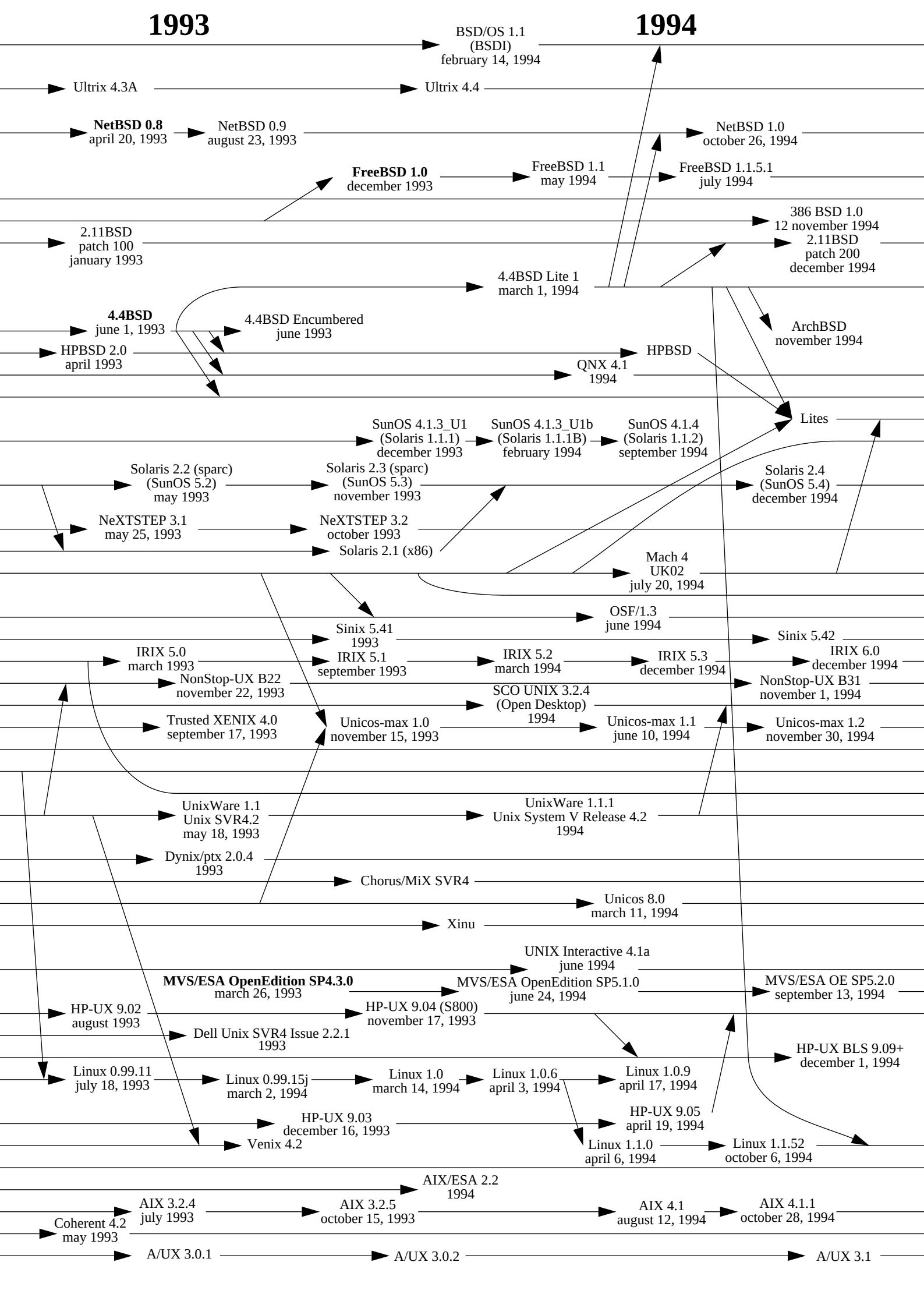
Note 2 : this diagram shows complete systems and [micro]kernels like Mach, Linux, the Hurd... This is because sometimes kernel versions are more appropriate to see the evolution of the system.

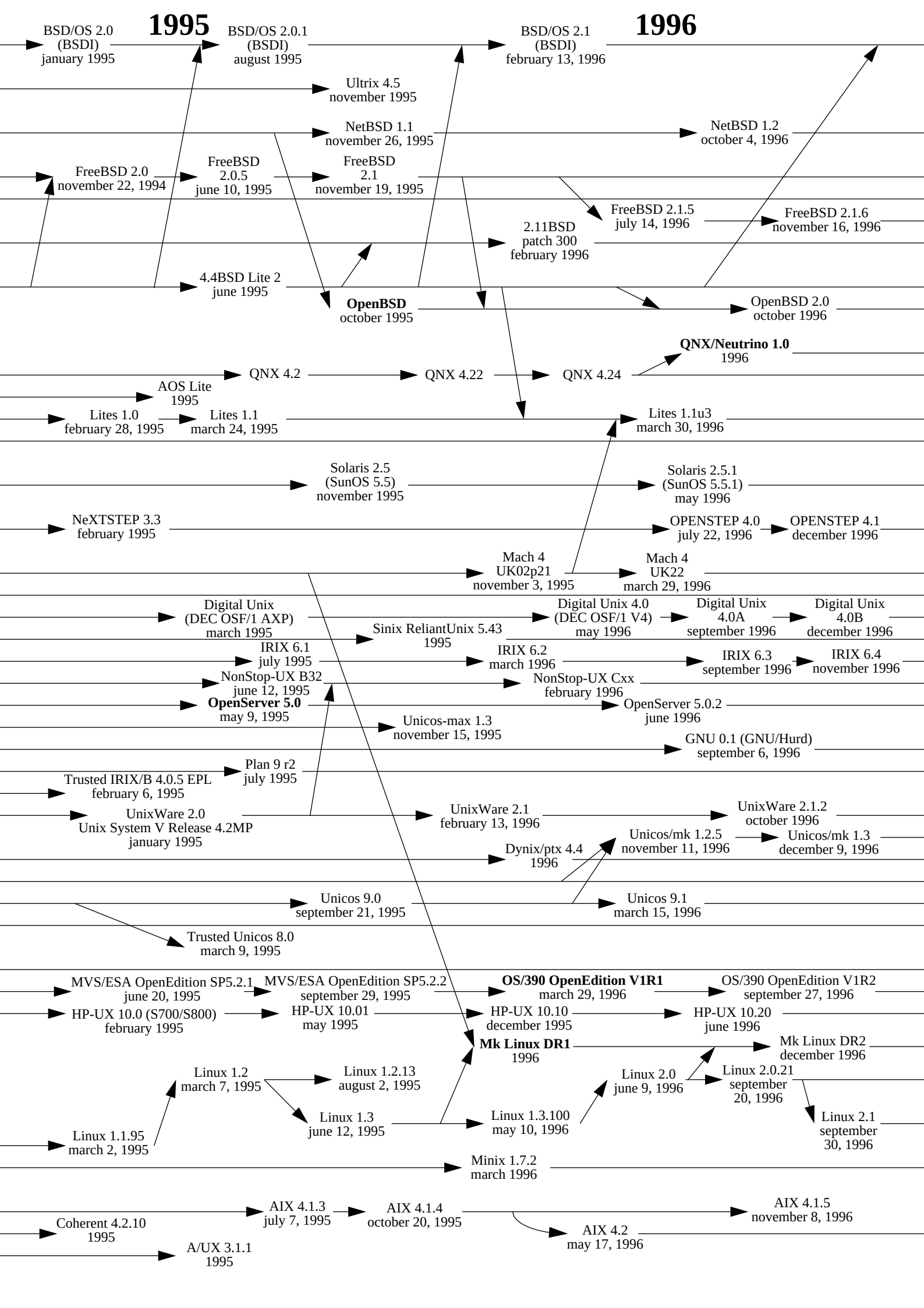

1981
1982
1983
V7M 2.1
october 1981
Ultrix-11
mt Xinu
july 19, 1983
2.8BSD
july 1981
2.8.1BSD
january 1982
2.8.2BSD
september 8, 1982
2.9BSD
july 1983
2.9.1BSD
november 1983
4.1BSD
june 1981
4.1aBSD
april 1982
4.1bBSD
august 1982
4.1cBSD
december 1982
4.2BSD
september 1983
QUNIX
1981
QNX beta
1983
Eunice 2
1982
SunOS 1.0
february 1982
UniSoft
UniPlus V7
1981
UniSoft
UniPlus
System V
1983
UNSW 81
april 1981
Tunis
1981
Plurix
1982
Sinix
XENIX 2.3
XENIX 3.0
april 1983
IRIS
GL2 1.0
1983
UNIX System III
november 1981
UNIX System IV
1982
UNIX System V
january 1983
TS 4.0.1
1981
TS 5.0
1982
TS 5.2
1983
PC/IX
IS/3
HP-UX 1.0 (S500)
H1 1983
SPIX
1982
HP-UX 2.0 (S200)
august 1983
UCLA Locus
1981
Locus
1983
Venix
Coherent
june 1983

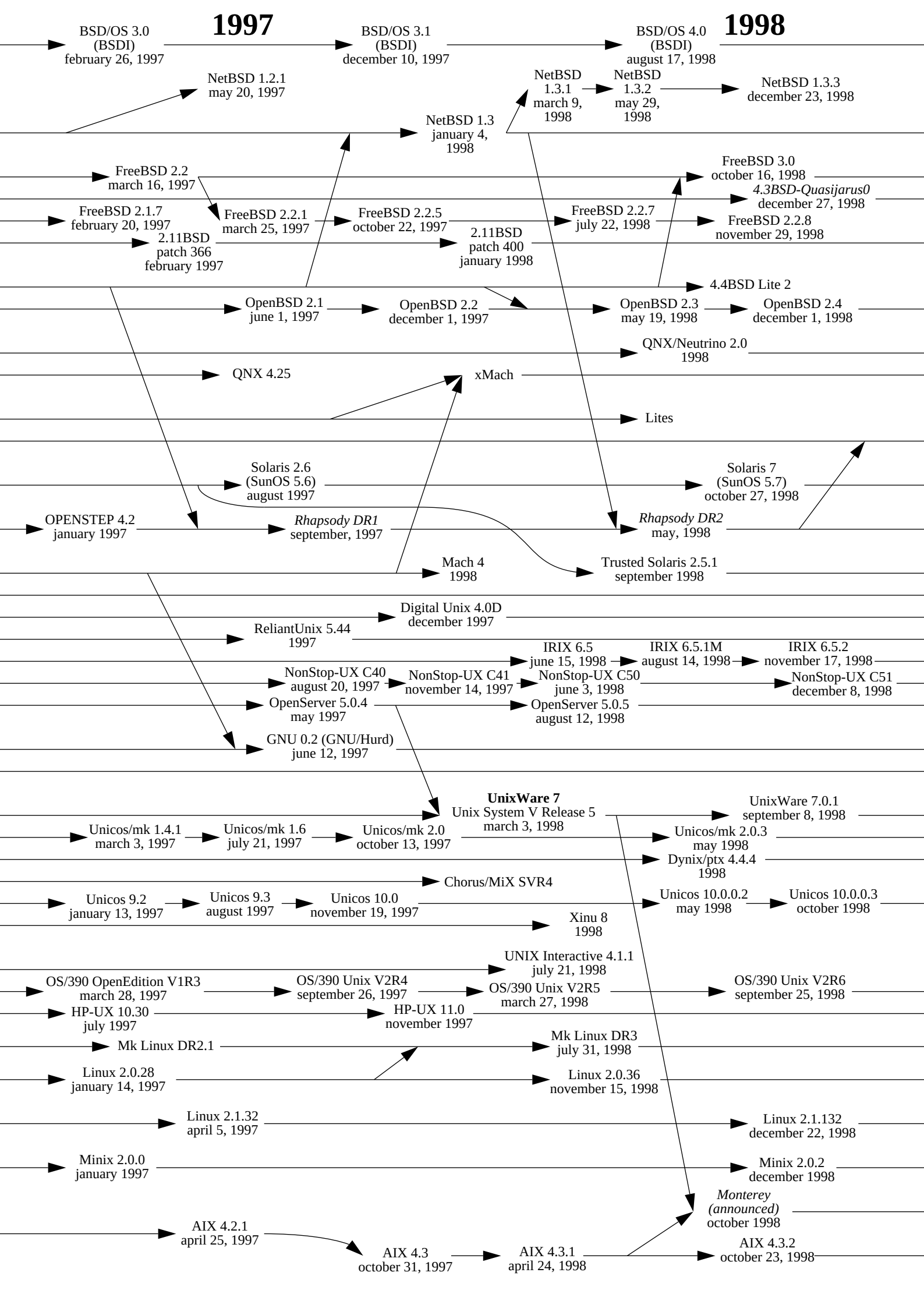

1984
1985
1986
Ultrix-11 v3.0
1986
Ultrix-11 v3.1
1986
Ultrix 32M 1.0
1984
Ultrix 32M 1.2
1985
mt Xinu
(4.2BSD)
mt Xinu
(4.3BSD)
2.9BSD-Seismo
august 1985
MIPS OS
RISC/os
4.3BSD
june 1986
QNX 1.0
1984
Eunice 4.2
1985
SunOS 1.1
april 1984
SunOS 1.2
january 1985
SunOS 2.0
may 15, 1985
SunOS 3.0
february 17,
1986
SunOS 3.2
september 1986
BRL Unix
(4.2BSD)
1985
BRL Unix
(4.3BSD)
1986
Mach
1985
Mach 2.0
UNIX Time-Sharing System
Eighth Edition (V8)
february 1985
UNIX Time-Sharing System
Ninth Edition (V9)
september 1986
IRIX
1986
SCO XENIX 3.0
february 1984
SCO XENIX
System V/286
1985
IRIS GL2 1.5
mid-1984
IRIS GL2 4.0
march 1986
IRIS GL2 5.0
dec. 1986
GNU (Trix)
1986
Plan 9
UNIX System
V/286
1985
UNIX System V
Release 2
april 1984
UNIX System V
Release 3.0
1986
UNIX System
V/386 rel 3.0
Dynix
1984
Chorus
1986
CXOs
1984
CXOs 0.9
sept. 1985
Unicos 1.0
april 3, 1986
Unicos 2.0
december 19, 1986
Xinu
1984
IS/5
Interactive 386/ix
1985
Microport
Unix SV/AT
january 1986
HP-UX 2.0 (S500)
H1 1984
HP-UX 2.1 (S500)
september 1984
IBM IX/370
1985
HP-UX 1.0 (S800)
november 20, 1986
SPIX 32
HP-UX 2.1 (S200)
H1 1985
HP-UX 5.0
(S200/S300)
H1 1986
HP-UX 5.1 (S200/S300)
H1 1986
Locus
1985
Venix 1.0
1985
Venix 2.0
1986
Venix/286
Minix
AIX/RT 2
1986
A/UX


1987
1988
1989
Ultrix 4.2
Ultrix 32M 2.0
1987
Acorn RISC iX
1989
BSD Net/1
november 1988
more/BSD
december 1988
mt Xinu
mach386
2.10BSD
april 1987
2.10.1BSD
january 1989
MIPS OS
RISC/os 4
4.3BSD Tahoe
june 1988
HPBSD
1987
HPBSD 1.0
april 1988
QNX 2.0
QNX 2.21
IBM AOS
1988
Eunice 4.3
1987
SunOS 3.5
1988
SunOS 4.0
1989
SunOS 4.0.3
may 1989
NeXTSTEP 0.8
october 12, 1988
NeXTSTEP 1.0
september 18, 1989
Mach 2.5
1988
UNIX Time-Sharing System
Tenth Edition (V10)
october 1989
Sinix 2.0
1987
Sinix 2.1
1988
IRIX 2.0
november 18, 1987
IRIX 3.0
june 10, 1988
NonStop-UX
april 10, 1987
NonStop-UX B00
august 22, 1989
SCO XENIX System V/386
october 1987
SCO XENIX System V/386
release 2.3.4
june 1989
SCO UNIX System V/386
release 3, 1989
IRIS GL2 6.0
1987
UNIX System V/386
Release 3.2
UNIX System V/386
Release 4
UNIX System V
Release 3.2
1987
UNIX System V
Release 4
1988
Acorn RISC Unix
1988
Atari Unix
1989
Chorus/MiX V3.2
1988
Unicos 3.0
september 25, 1987
Unicos 4.0
july 15, 1988
Unicos 5.0
may 15, 1989
Xinu 7
march 1988
CTIX/386
CTIX 3.0
CTIX 3.2
CTIX 4.0
HP-UX 7.02
(S800)
H2 1989
UNIX Interactive 4.1
1988
Microport Unix V/386
september 1987
HP-UX 1.1
(S800)
august 17, 1987
HP-UX 1.2
(S800)
nov. 16, 1987
HP-UX 2.0
(S800)
march 1988
HP-UX 2.1
(S800)
july 1988
HP-UX 3.0
(S800)
nov. 11, 1988
HP-UX 3.1
feb. 1989
HP-UX 7.0
(S300/S800)
H2 1989
Dell Unix SVR1.1
november 1, 1989
BOS
1989
HP-UX 5.2
(S300)
H2 1987
HP-UX 6.0
(S300)
H2 1987
HP-UX 6.2
(S300)
june 1988
HP-UX 6.5
(S300)
H1 1989
AIX PS/2 1.1
march 31, 1989
Venix 3.2
Venix 3.2.3
Minix 1.0
1987
AIX/RT 2.1.2
AIX/RT 2.2.1
1987
AIX/6000 v3
1989
A/UX 1.0
february 1988

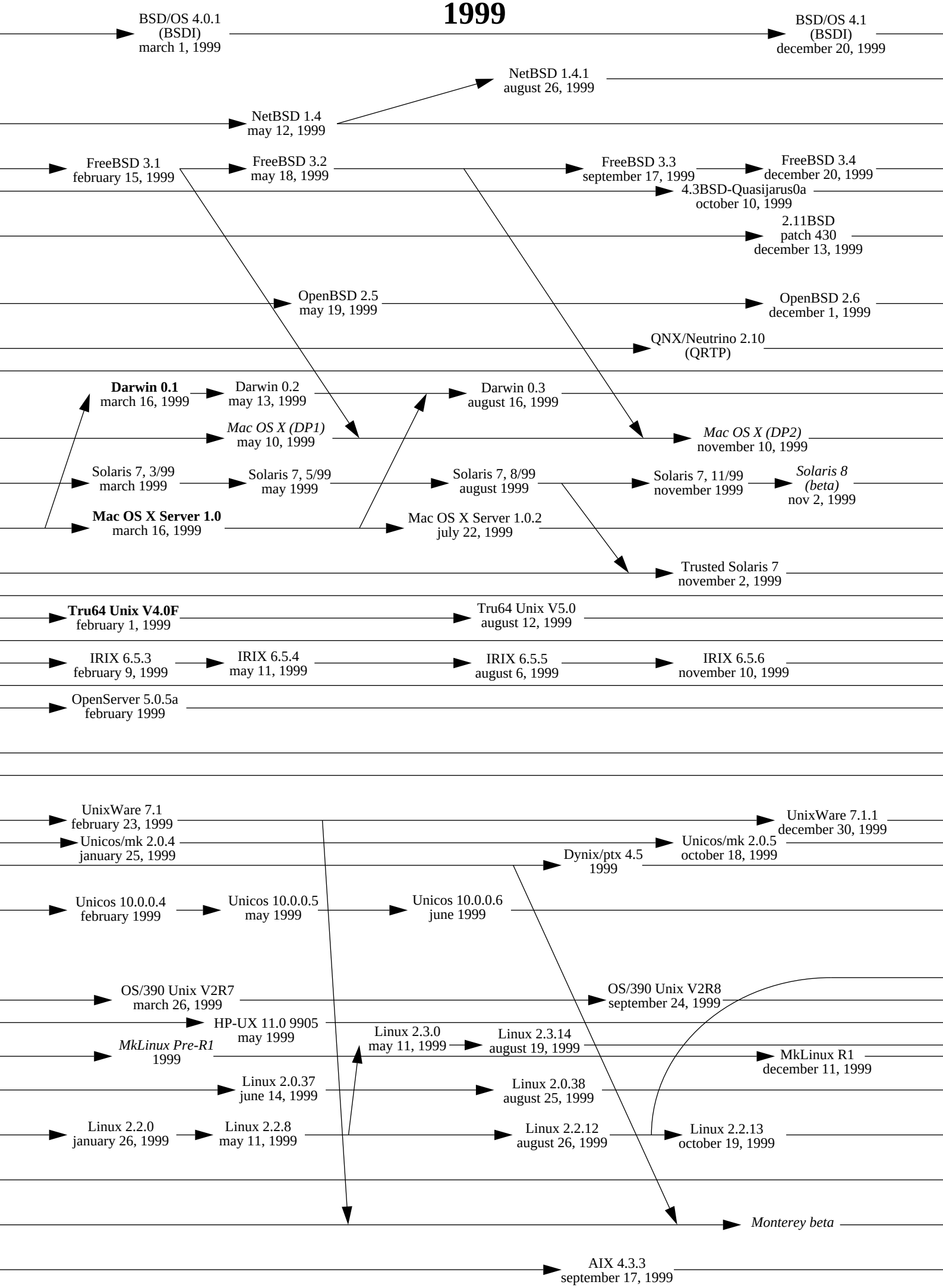

1990
1991
1992
BSD/386 0.3.2
(BSDI)
february 28, 1992
BSD/OS 1.0
(BSDI)
Ultrix 4.2A
Ultrix 4.3
RISC iX 1.21
1991
BSD Net/2
(4.3BSD Lite)
june 1991
mt Xinu
mach 2.6
386 BSD 0.0
february 1992
386 BSD 0.1
july 14, 1992
2.11BSD
february 1992
MIPS OS
RISC/os 5
4.3BSD Reno
june 1990
4.4BSD alpha
june 1992
QNX 4.0
1990
AOS Reno
1992
SunOS 4.1
march 1990
SunOS 4.1.1
(Solaris 1)
november 1990
SunOS 4.1.2
(Solaris 1.0.1)
december 1991
SunOS 4.1.3
(Solaris 1.1a)
august 1992
Solaris 2.0
(sparc)
(SunOS 5.0)
july 1992
Solaris 2.1
(SunOS 5.1)
december 1992
NeXTSTEP 2.0
sept. 18, 1990
NeXTSTEP 2.1
march 25, 1991
NeXTSTEP 3.0
september 1992
Solaris 2.0 (x86)
end 1992
Mach 2.6
Mach 3
OSF/1
1990
OSF/1
1992
Sinix 5.20
1990
Sinix 5.40
1992
IRIX 4.0
september 1991
IRIX 4.0.4
march 1992
Trusted XENIX 2.0
january 9, 1991
Trusted XENIX 3.0
april 8, 1992
GNU
(GNU/Hurd)
may 7, 1991
Plan 9
1990
AMiX 1.1 (Amiga Unix SVR4)
1990
AMiX 2.2
UnixWare 1
Unix System V
Release 4.2
november 2, 1992
ASV (dev release)
1991
ASV (final release)
august 1992
Chorus/MiX SVR4
1991
Unicos 6.0
february 14, 1991
Unicos 7.0
october 29, 1992
Xinu
HP-UX 7.06
(S800)
H2 1990
HP-UX 7.08
(S800)
H1 1991
Unix System V
Release 4.1ES
december 1992
UNIX Interactive 4.1
Microport Unix SVR3.2
Microport Unix SVR4.0
Microport Unix SVR4.1
HP-UX 8.0 (S300/S800)
march 27, 1991
HP-UX 8.02 (S800)
august 5, 1991
HP-UX 8.06 (S800)
H2 1991
HP-UX 9.0 (S800)
october 7, 1992
Dell Unix SVR4.0
1990
Dell Unix SVR4 Issue 2.2
1992
HP-UX BLS 8.04 (S800)
H2 1991
Linux 0.01
august 1, 1991
Linux 0.02
october 5, 1991
Linux 0.12
january 16, 1992
Linux 0.95
march 8, 1992
AIX PS/2 & AIX/370
1.2
march 30, 1990
AIX PS/2 & AIX/370
1.2.1
february 22, 1991
HP-UX 8.01 (S700)
february 1991
HP-UX 8.05 (S300/700)
july 1991
HP-UX 8.07
(S300/S700)
november 21, 1991
HP-UX 9,01
(S300/S700)
november 2, 1992
Venix 3.2.4
Minix 1.5
december 1992
AIX/ESA 1
1991
AIX/ESA 2.1
1992
AIX 3.1
1990
AIX 3.2
1990
Coherent 3.0
Coherent 4.0
may 1992
A/UX 2.0
june 1990
A/UX 3.0
april 16, 1992
AIX PS/2 1.3
october 2, 1992


1993
1994
BSD/OS 1.1
(BSDI)
february 14, 1994
Ultrix 4.3A
Ultrix 4.4
NetBSD 0.8
april 20, 1993
NetBSD 0.9
august 23, 1993
NetBSD 1.0
october 26, 1994
FreeBSD 1.0
december 1993
FreeBSD 1.1
may 1994
FreeBSD 1.1.5.1
july 1994
386 BSD 1.0
12 november 1994
2.11BSD
patch 100
january 1993
2.11BSD
patch 200
december 1994
4.4BSD Lite 1
march 1, 1994
4.4BSD
june 1, 1993
4.4BSD Encumbered
june 1993
ArchBSD
november 1994
HPBSD 2.0
april 1993
HPBSD
QNX 4.1
1994
Lites
SunOS 4.1.3_U1
(Solaris 1.1.1)
december 1993
SunOS 4.1.3_U1b
(Solaris 1.1.1B)
february 1994
SunOS 4.1.4
(Solaris 1.1.2)
september 1994
Solaris 2.2 (sparc)
(SunOS 5.2)
may 1993
Solaris 2.3 (sparc)
(SunOS 5.3)
november 1993
Solaris 2.4
(SunOS 5.4)
december 1994
NeXTSTEP 3.1
may 25, 1993
NeXTSTEP 3.2
october 1993
Solaris 2.1 (x86)
Mach 4
UK02
july 20, 1994
OSF/1.3
june 1994
Sinix 5.41
1993
Sinix 5.42
IRIX 5.0
march 1993
IRIX 5.1
september 1993
IRIX 5.2
march 1994
IRIX 5.3
december 1994
IRIX 6.0
december 1994
NonStop-UX B22
november 22, 1993
NonStop-UX B31
november 1, 1994
SCO UNIX 3.2.4
(Open Desktop)
1994
Trusted XENIX 4.0
september 17, 1993
Unicos-max 1.0
november 15, 1993
Unicos-max 1.1
june 10, 1994
Unicos-max 1.2
november 30, 1994
UnixWare 1.1
Unix SVR4.2
may 18, 1993
UnixWare 1.1.1
Unix System V Release 4.2
1994
Dynix/ptx 2.0.4
1993
Chorus/MiX SVR4
Unicos 8.0
march 11, 1994
Xinu
UNIX Interactive 4.1a
june 1994
MVS/ESA OpenEdition SP4.3.0
march 26, 1993
MVS/ESA OpenEdition SP5.1.0
june 24, 1994
MVS/ESA OE SP5.2.0
september 13, 1994
HP-UX 9.02
august 1993
HP-UX 9.04 (S800)
november 17, 1993
Dell Unix SVR4 Issue 2.2.1
1993
HP-UX BLS 9.09+
december 1, 1994
Linux 0.99.11
july 18, 1993
Linux 0.99.15j
march 2, 1994
Linux 1.0
march 14, 1994
Linux 1.0.6
april 3, 1994
Linux 1.0.9
april 17, 1994
HP-UX 9.03
december 16, 1993
HP-UX 9.05
april 19, 1994
Venix 4.2
Linux 1.1.0
april 6, 1994
Linux 1.1.52
october 6, 1994
AIX/ESA 2.2
1994
AIX 3.2.4
july 1993
AIX 3.2.5
october 15, 1993
AIX 4.1
august 12, 1994
AIX 4.1.1
october 28, 1994
Coherent 4.2
may 1993
A/UX 3.0.1
A/UX 3.0.2
A/UX 3.1

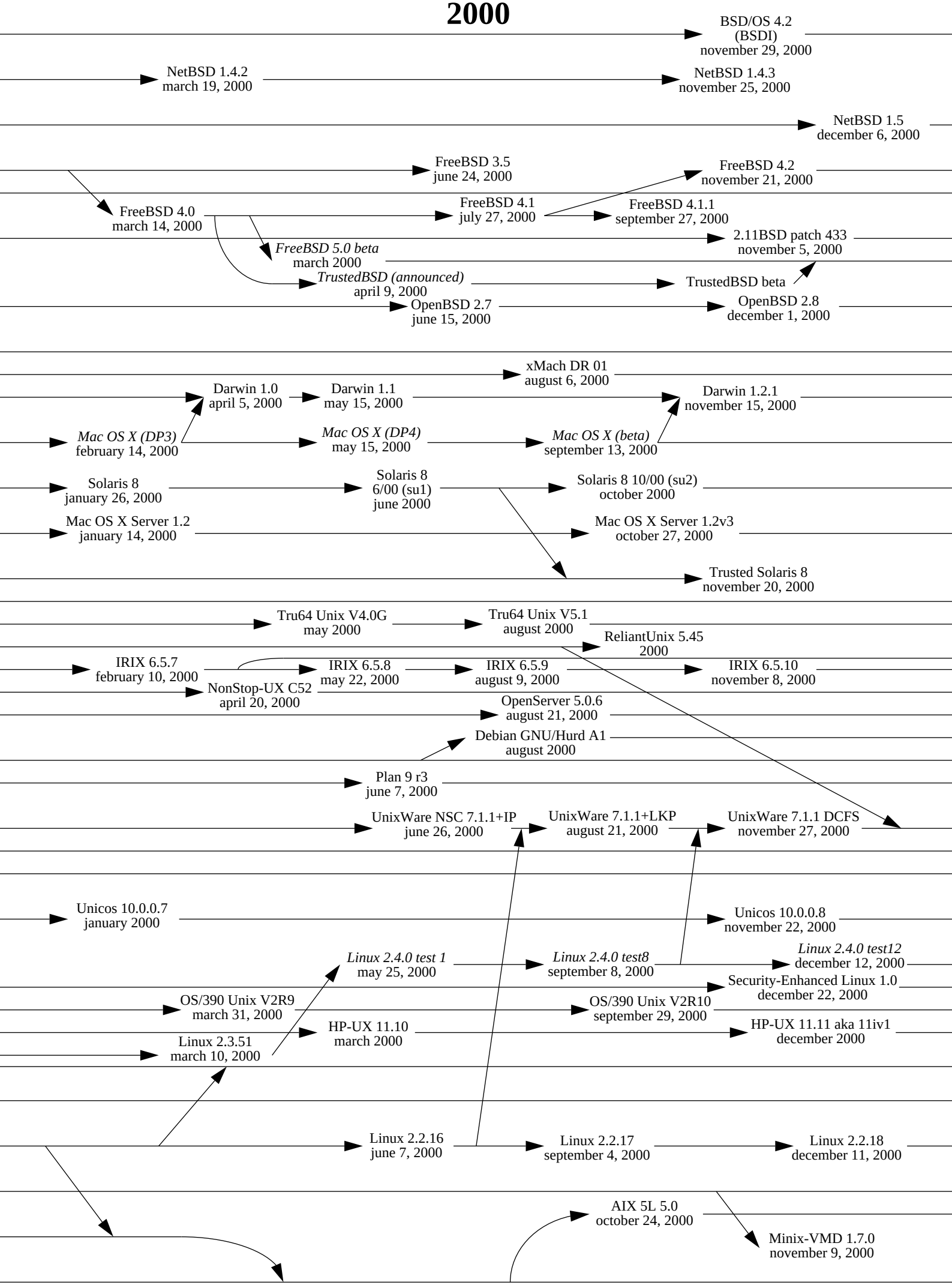

1995
1996
BSD/OS 2.0 (BSDI) january 1995
BSD/OS 2.0.1 (BSDI) august 1995
BSD/OS 2.1 (BSDI) february 13, 1996
Ultrix 4.5 november 1995
NetBSD 1.1 november 26, 1995
NetBSD 1.2 october 4, 1996
FreeBSD 2.0 november 22, 1994
FreeBSD 2.0.5 june 10, 1995
FreeBSD 2.1 november 19, 1995
FreeBSD 2.1.5 july 14, 1996
FreeBSD 2.1.6 november 16, 1996
2.11BSD patch 300 february 1996
4.4BSD Lite 2 june 1995
OpenBSD october 1995
OpenBSD 2.0 october 1996
QNX/Neutrino 1.0 1996
QNX 4.2
QNX 4.22
QNX 4.24
AOS Lite 1995
Lites 1.0 february 28, 1995
Lites 1.1 march 24, 1995
Lites 1.1u3 march 30, 1996
Solaris 2.5 (SunOS 5.5) november 1995
Solaris 2.5.1 (SunOS 5.5.1) may 1996
NeXTSTEP 3.3 february 1995
OPENSTEP 4.0 july 22, 1996
OPENSTEP 4.1 december 1996
Mach 4 UK02p21 november 3, 1995
Mach 4 UK22 march 29, 1996
Digital Unix (DEC OSF/1 AXP) march 1995
Digital Unix 4.0 (DEC OSF/1 V4) may 1996
Digital Unix 4.0A september 1996
Digital Unix 4.0B december 1996
Sinix ReliantUnix 5.43 1995
IRIX 6.1 july 1995
IRIX 6.2 march 1996
IRIX 6.3 september 1996
IRIX 6.4 november 1996
NonStop-UX B32 june 12, 1995
NonStop-UX Cxx february 1996
OpenServer 5.0 may 9, 1995
OpenServer 5.0.2 june 1996
Unicos-max 1.3 november 15, 1995
GNU 0.1 (GNU/Hurd) september 6, 1996
Plan 9 r2 july 1995
Trusted IRIX/B 4.0.5 EPL february 6, 1995
UnixWare 2.0 Unix System V Release 4.2MP january 1995
UnixWare 2.1 february 13, 1996
UnixWare 2.1.2 october 1996
Unicos/mk 1.2.5 november 11, 1996
Unicos/mk 1.3 december 9, 1996
Dynix/ptx 4.4 1996
Unicos 9.0 september 21, 1995
Unicos 9.1 march 15, 1996
Trusted Unicos 8.0 march 9, 1995
MVS/ESA OpenEdition SP5.2.1 june 20, 1995
MVS/ESA OpenEdition SP5.2.2 september 29, 1995
OS/390 OpenEdition V1R1 march 29, 1996
OS/390 OpenEdition V1R2 september 27, 1996
HP-UX 10.0 (S700/S800) february 1995
HP-UX 10.01 may 1995
HP-UX 10.10 december 1995
HP-UX 10.20 june 1996
Mk Linux DR1 1996
Mk Linux DR2 december 1996
Linux 1.2 march 7, 1995
Linux 1.2.13 august 2, 1995
Linux 2.0 june 9, 1996
Linux 2.0.21 september 20, 1996
Linux 1.3 june 12, 1995
Linux 1.3.100 may 10, 1996
Linux 2.1 september 30, 1996
Linux 1.1.95 march 2, 1995
Minix 1.7.2 march 1996
AIX 4.1.3 july 7, 1995
AIX 4.1.4 october 20, 1995
AIX 4.1.5 november 8, 1996
Coherent 4.2.10 1995
AIX 4.2 may 17, 1996
A/UX 3.1.1 1995


1997
1998
BSD/OS 3.0 (BSDI) february 26, 1997
BSD/OS 3.1 (BSDI) december 10, 1997
BSD/OS 4.0 (BSDI) august 17, 1998
NetBSD 1.2.1 may 20, 1997
NetBSD 1.3 january 4, 1998
NetBSD 1.3.1 march 9, 1998
NetBSD 1.3.2 may 29, 1998
NetBSD 1.3.3 december 23, 1998
FreeBSD 2.2 march 16, 1997
FreeBSD 3.0 october 16, 1998
4.3BSD-Quasijarus0 december 27, 1998
FreeBSD 2.1.7 february 20, 1997
FreeBSD 2.2.1 march 25, 1997
FreeBSD 2.2.5 october 22, 1997
FreeBSD 2.2.7 july 22, 1998
FreeBSD 2.2.8 november 29, 1998
2.11BSD patch 366 february 1997
2.11BSD patch 400 january 1998
4.4BSD Lite 2
OpenBSD 2.1 june 1, 1997
OpenBSD 2.2 december 1, 1997
OpenBSD 2.3 may 19, 1998
OpenBSD 2.4 december 1, 1998
QNX/Neutrino 2.0 1998
QNX 4.25
xMach
Lites
Solaris 2.6 (SunOS 5.6) august 1997
Solaris 7 (SunOS 5.7) october 27, 1998
OPENSTEP 4.2 january 1997
Rhapsody DR1 september, 1997
Rhapsody DR2 may, 1998
Mach 4 1998
Trusted Solaris 2.5.1 september 1998
Digital Unix 4.0D december 1997
ReliantUnix 5.44 1997
IRIX 6.5 june 15, 1998
IRIX 6.5.1M august 14, 1998
IRIX 6.5.2 november 17, 1998
NonStop-UX C40 august 20, 1997
NonStop-UX C41 november 14, 1997
NonStop-UX C50 june 3, 1998
NonStop-UX C51 december 8, 1998
OpenServer 5.0.4 may 1997
OpenServer 5.0.5 august 12, 1998
GNU 0.2 (GNU/Hurd) june 12, 1997
UnixWare 7 Unix System V Release 5 march 3, 1998
UnixWare 7.0.1 september 8, 1998
Unicos/mk 1.4.1 march 3, 1997
Unicos/mk 1.6 july 21, 1997
Unicos/mk 2.0 october 13, 1997
Unicos/mk 2.0.3 may 1998
Dynix/ptx 4.4.4 1998
Chorus/MiX SVR4
Unicos 9.2 january 13, 1997
Unicos 9.3 august 1997
Unicos 10.0 november 19, 1997
Unicos 10.0.0.2 may 1998
Unicos 10.0.0.3 october 1998
Xinu 8 1998
UNIX Interactive 4.1.1 july 21, 1998
OS/390 OpenEdition V1R3 march 28, 1997
OS/390 Unix V2R4 september 26, 1997
OS/390 Unix V2R5 march 27, 1998
OS/390 Unix V2R6 september 25, 1998
HP-UX 10.30 july 1997
HP-UX 11.0 november 1997
Mk Linux DR2.1
Mk Linux DR3 july 31, 1998
Linux 2.0.28 january 14, 1997
Linux 2.0.36 november 15, 1998
Linux 2.1.32 april 5, 1997
Linux 2.1.132 december 22, 1998
Minix 2.0.0 january 1997
Minix 2.0.2 december 1998
Monterey (announced) october 1998
AIX 4.2.1 april 25, 1997
AIX 4.3 october 31, 1997
AIX 4.3.1 april 24, 1998
AIX 4.3.2 october 23, 1998

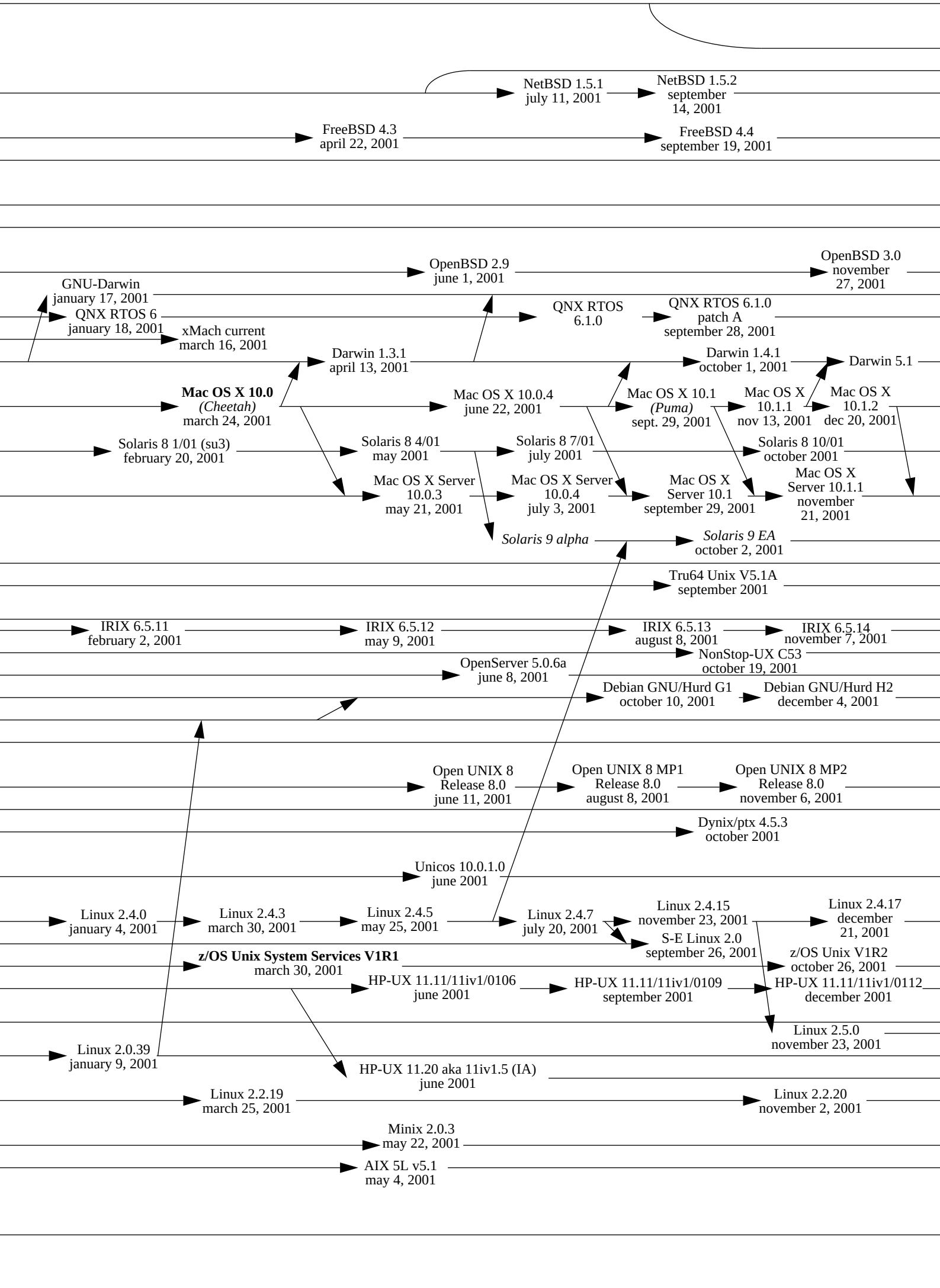
1999


BSD/OS 4.0.1
(BSDI)
march 1, 1999
BSD/OS 4.1
(BSDI)
december 20, 1999
NetBSD 1.4.1
august 26, 1999
NetBSD 1.4
may 12, 1999
FreeBSD 3.1
february 15, 1999
FreeBSD 3.2
may 18, 1999
FreeBSD 3.3
september 17, 1999
FreeBSD 3.4
december 20, 1999
4.3BSD-Quasijarus0a
october 10, 1999
2.11BSD
patch 430
december 13, 1999
OpenBSD 2.5
may 19, 1999
OpenBSD 2.6
december 1, 1999
QNX/Neutrino 2.10
(QRTP)
Darwin 0.1
march 16, 1999
Darwin 0.2
may 13, 1999
Darwin 0.3
august 16, 1999
Mac OS X (DP1)
may 10, 1999
Mac OS X (DP2)
november 10, 1999
Solaris 7, 3/99
march 1999
Solaris 7, 5/99
may 1999
Solaris 7, 8/99
august 1999
Solaris 7, 11/99
november 1999
Solaris 8
(beta)
nov 2, 1999
Mac OS X Server 1.0
march 16, 1999
Mac OS X Server 1.0.2
july 22, 1999
Trusted Solaris 7
november 2, 1999
Tru64 Unix V4.0F
february 1, 1999
Tru64 Unix V5.0
august 12, 1999
IRIX 6.5.3
february 9, 1999
IRIX 6.5.4
may 11, 1999
IRIX 6.5.5
august 6, 1999
IRIX 6.5.6
november 10, 1999
OpenServer 5.0.5a
february 1999
UnixWare 7.1
february 23, 1999
UnixWare 7.1.1
december 30, 1999
Unicos/mk 2.0.4
january 25, 1999
Unicos/mk 2.0.5
october 18, 1999
Dynix/ptx 4.5
1999
Unicos 10.0.0.4
february 1999
Unicos 10.0.0.5
may 1999
Unicos 10.0.0.6
june 1999
OS/390 Unix V2R7
march 26, 1999
OS/390 Unix V2R8
september 24, 1999
HP-UX 11.0 9905
may 1999
MkLinux Pre-R1
1999
Linux 2.3.0
may 11, 1999
Linux 2.3.14
august 19, 1999
MkLinux R1
december 11, 1999
Linux 2.0.37
june 14, 1999
Linux 2.0.38
august 25, 1999
Linux 2.2.0
january 26, 1999
Linux 2.2.8
may 11, 1999
Linux 2.2.12
august 26, 1999
Linux 2.2.13
october 19, 1999
Monterey beta
AIX 4.3.3
september 17, 1999

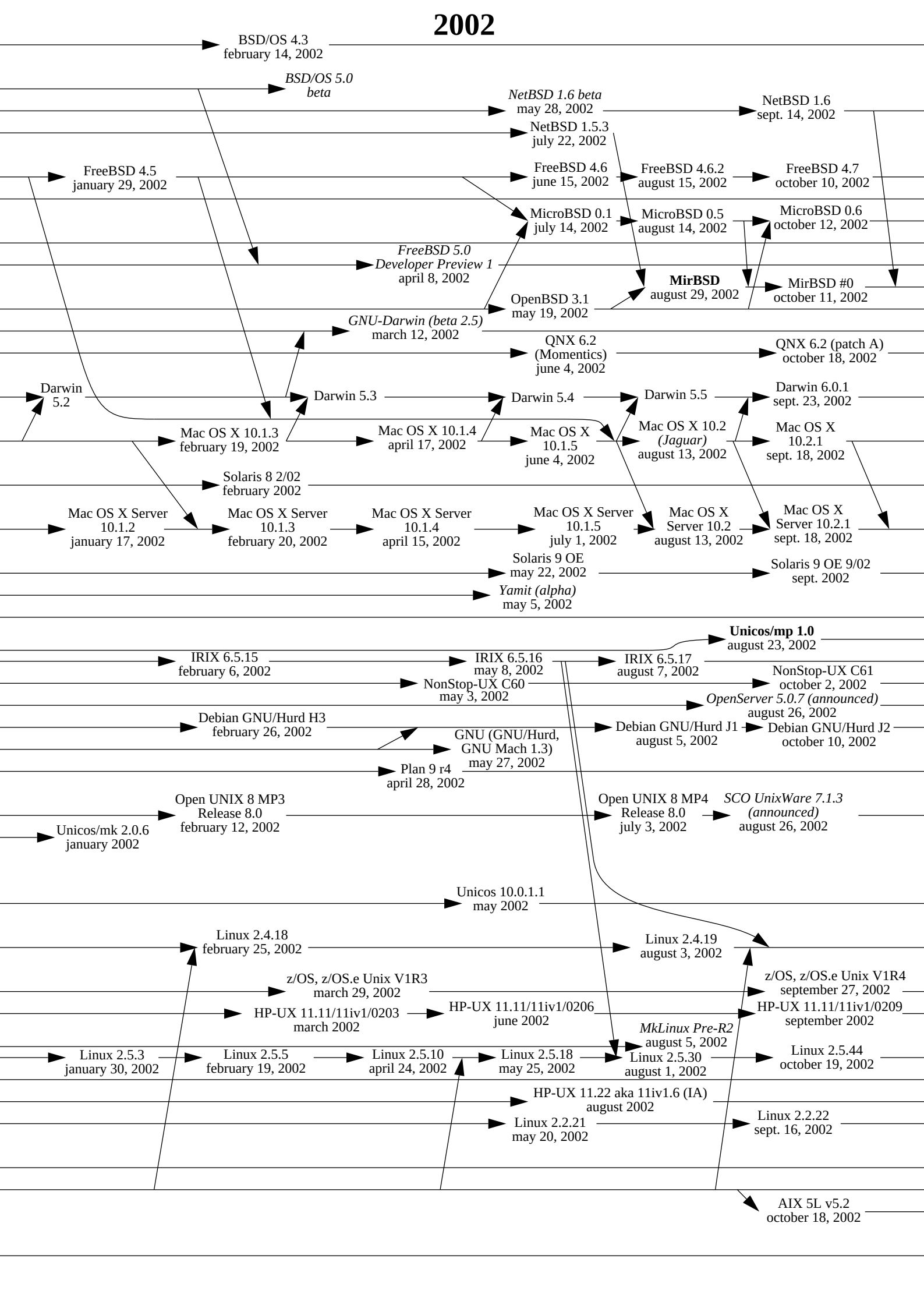
2000


BSD/OS 4.2
(BSDI)
november 29, 2000
NetBSD 1.4.2
march 19, 2000
NetBSD 1.4.3
november 25, 2000
NetBSD 1.5
december 6, 2000
FreeBSD 3.5
june 24, 2000
FreeBSD 4.2
november 21, 2000
FreeBSD 4.0
march 14, 2000
FreeBSD 4.1
july 27, 2000
FreeBSD 4.1.1
september 27, 2000
2.11BSD patch 433
november 5, 2000
FreeBSD 5.0 beta
march 2000
TrustedBSD (announced)
april 9, 2000
TrustedBSD beta
OpenBSD 2.7
june 15, 2000
OpenBSD 2.8
december 1, 2000
xMach DR 01
august 6, 2000
Darwin 1.0
april 5, 2000
Darwin 1.1
may 15, 2000
Darwin 1.2.1
november 15, 2000
Mac OS X (DP3)
february 14, 2000
Mac OS X (DP4)
may 15, 2000
Mac OS X (beta)
september 13, 2000
Solaris 8
january 26, 2000
Solaris 8
6/00 (su1)
june 2000
Solaris 8 10/00 (su2)
october 2000
Mac OS X Server 1.2
january 14, 2000
Mac OS X Server 1.2v3
october 27, 2000
Trusted Solaris 8
november 20, 2000
Tru64 Unix V4.0G
may 2000
Tru64 Unix V5.1
august 2000
ReliantUnix 5.45
2000
IRIX 6.5.7
february 10, 2000
IRIX 6.5.8
may 22, 2000
IRIX 6.5.9
august 9, 2000
IRIX 6.5.10
november 8, 2000
NonStop-UX C52
april 20, 2000
OpenServer 5.0.6
august 21, 2000
Debian GNU/Hurd A1
august 2000
Plan 9 r3
june 7, 2000
UnixWare NSC 7.1.1+IP
june 26, 2000
UnixWare 7.1.1+LKP
august 21, 2000
UnixWare 7.1.1 DCFS
november 27, 2000
Unicos 10.0.0.7
january 2000
Unicos 10.0.0.8
november 22, 2000
Linux 2.4.0 test 1
may 25, 2000
Linux 2.4.0 test8
september 8, 2000
Linux 2.4.0 test12
december 12, 2000
Security-Enhanced Linux 1.0
december 22, 2000
OS/390 Unix V2R9
march 31, 2000
OS/390 Unix V2R10
september 29, 2000
HP-UX 11.10
march 2000
HP-UX 11.11 aka 11iv1
december 2000
Linux 2.3.51
march 10, 2000
Linux 2.2.16
june 7, 2000
Linux 2.2.17
september 4, 2000
Linux 2.2.18
december 11, 2000
AIX 5L 5.0
october 24, 2000
Minix-VMD 1.7.0
november 9, 2000

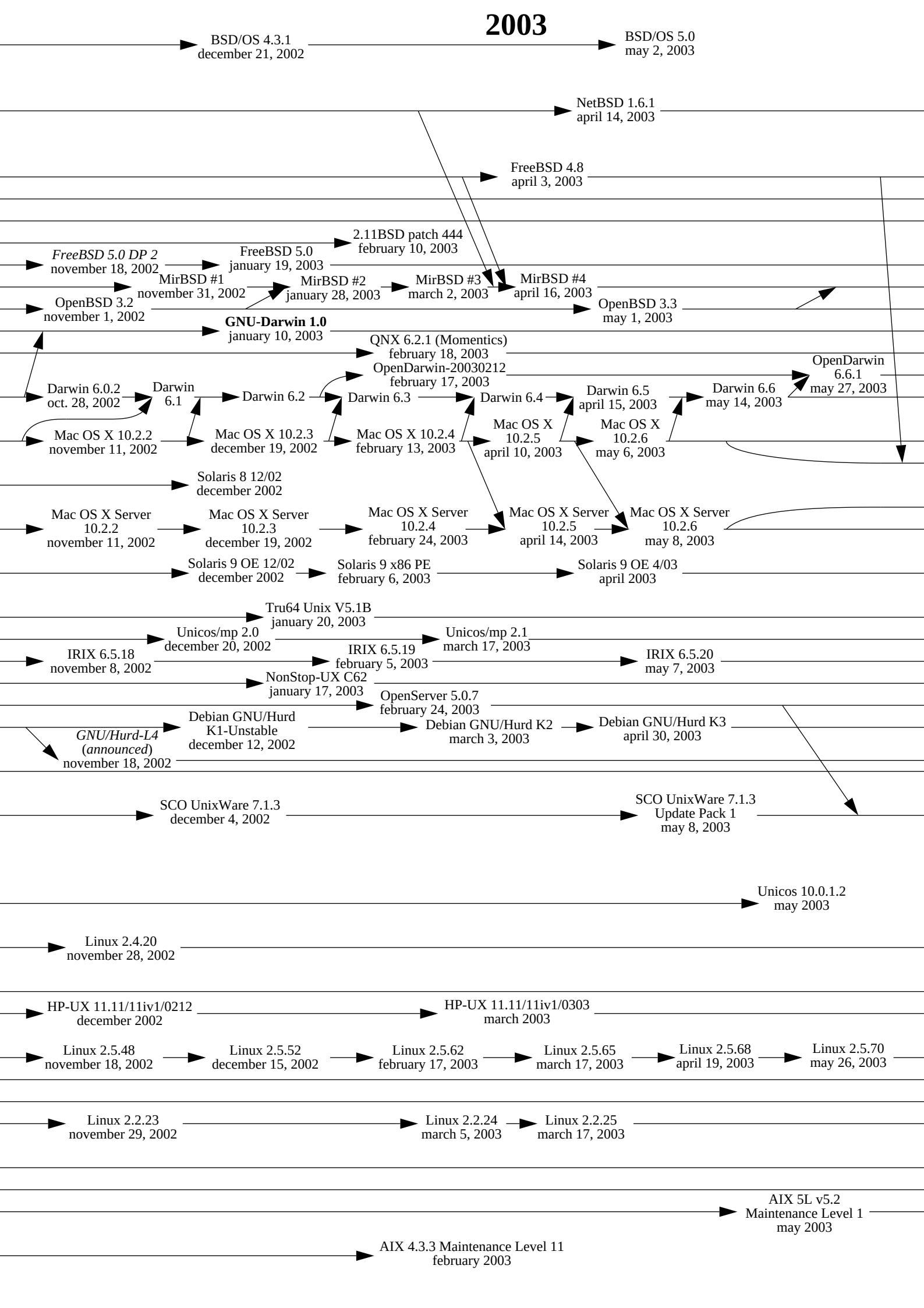
2001

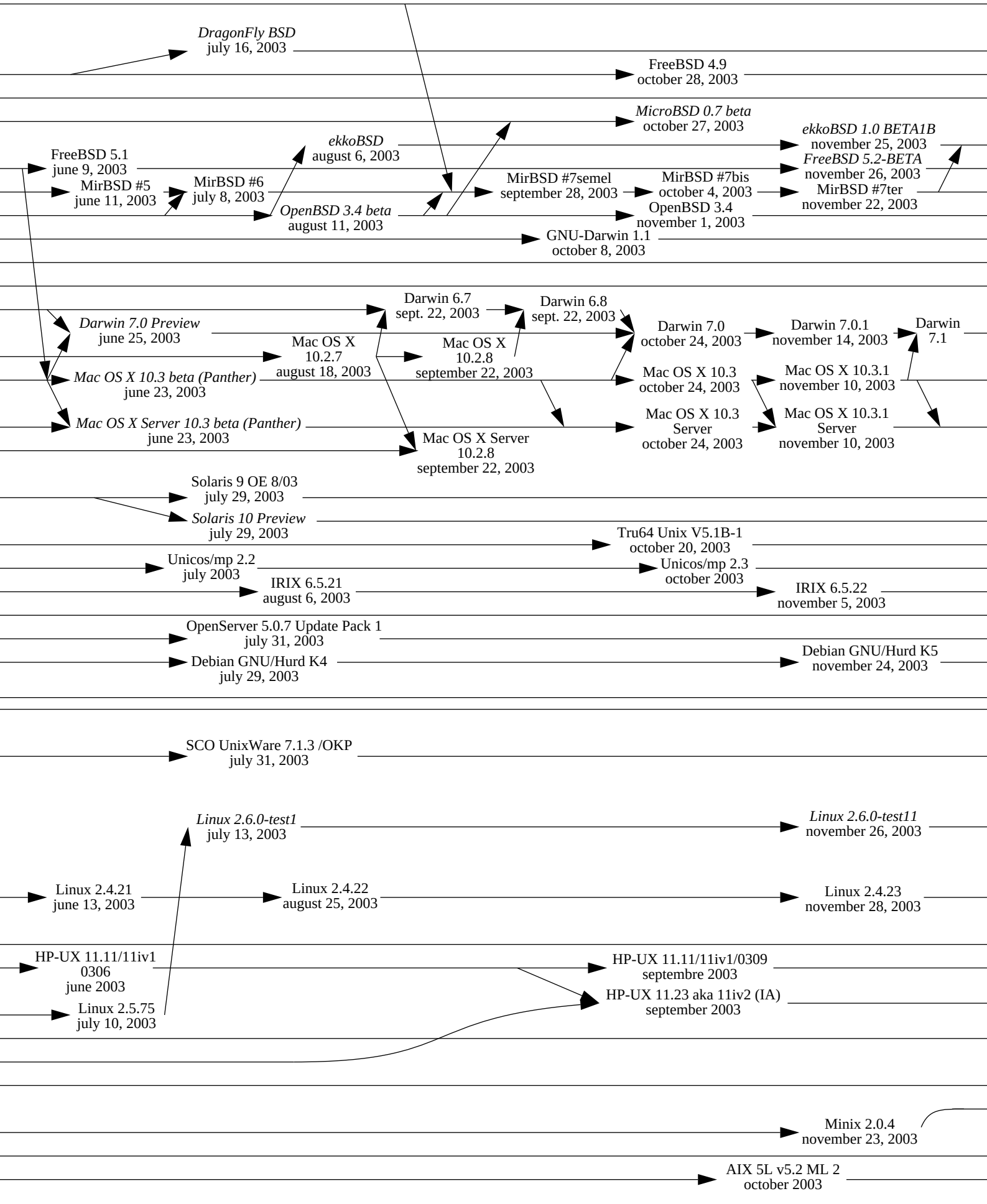

NetBSD 1.5.1
july 11, 2001
NetBSD 1.5.2
september
14, 2001
FreeBSD 4.3
april 22, 2001
FreeBSD 4.4
september 19, 2001
OpenBSD 2.9
june 1, 2001
OpenBSD 3.0
november
27, 2001
GNU-Darwin
january 17, 2001
QNX RTOS 6
january 18, 2001
QNX RTOS
6.1.0
QNX RTOS 6.1.0
patch A
september 28, 2001
xMach current
march 16, 2001
Darwin 1.3.1
april 13, 2001
Darwin 1.4.1
october 1, 2001
Darwin 5.1
Mac OS X 10.0
(Cheetah)
march 24, 2001
Mac OS X 10.0.4
june 22, 2001
Mac OS X 10.1
(Puma)
sept. 29, 2001
Mac OS X
10.1.1
nov 13, 2001
Mac OS X
10.1.2
dec 20, 2001
Solaris 8 1/01 (su3)
february 20, 2001
Solaris 8 4/01
may 2001
Solaris 8 7/01
july 2001
Solaris 8 10/01
october 2001
Mac OS X Server
10.0.3
may 21, 2001
Mac OS X Server
10.0.4
july 3, 2001
Mac OS X
Server 10.1
september 29, 2001
Mac OS X
Server 10.1.1
november
21, 2001
Solaris 9 alpha
Solaris 9 EA
october 2, 2001
Tru64 Unix V5.1A
september 2001
IRIX 6.5.11
february 2, 2001
IRIX 6.5.12
may 9, 2001
IRIX 6.5.13
august 8, 2001
IRIX 6.5.14
november 7, 2001
NonStop-UX C53
october 19, 2001
OpenServer 5.0.6a
june 8, 2001
Debian GNU/Hurd G1
october 10, 2001
Debian GNU/Hurd H2
december 4, 2001
Open UNIX 8
Release 8.0
june 11, 2001
Open UNIX 8 MP1
Release 8.0
august 8, 2001
Open UNIX 8 MP2
Release 8.0
november 6, 2001
Dynix/ptx 4.5.3
october 2001
Unicos 10.0.1.0
june 2001
Linux 2.4.0
january 4, 2001
Linux 2.4.3
march 30, 2001
Linux 2.4.5
may 25, 2001
Linux 2.4.7
july 20, 2001
Linux 2.4.15
november 23, 2001
Linux 2.4.17
december
21, 2001
S-E Linux 2.0
september 26, 2001
z/OS Unix System Services V1R1
march 30, 2001
z/OS Unix V1R2
october 26, 2001
HP-UX 11.11/11iv1/0106
june 2001
HP-UX 11.11/11iv1/0109
september 2001
HP-UX 11.11/11iv1/0112
december 2001
Linux 2.5.0
november 23, 2001
Linux 2.0.39
january 9, 2001
HP-UX 11.20 aka 11iv1.5 (IA)
june 2001
Linux 2.2.19
march 25, 2001
Linux 2.2.20
november 2, 2001
Minix 2.0.3
may 22, 2001
AIX 5L v5.1
may 4, 2001

2002

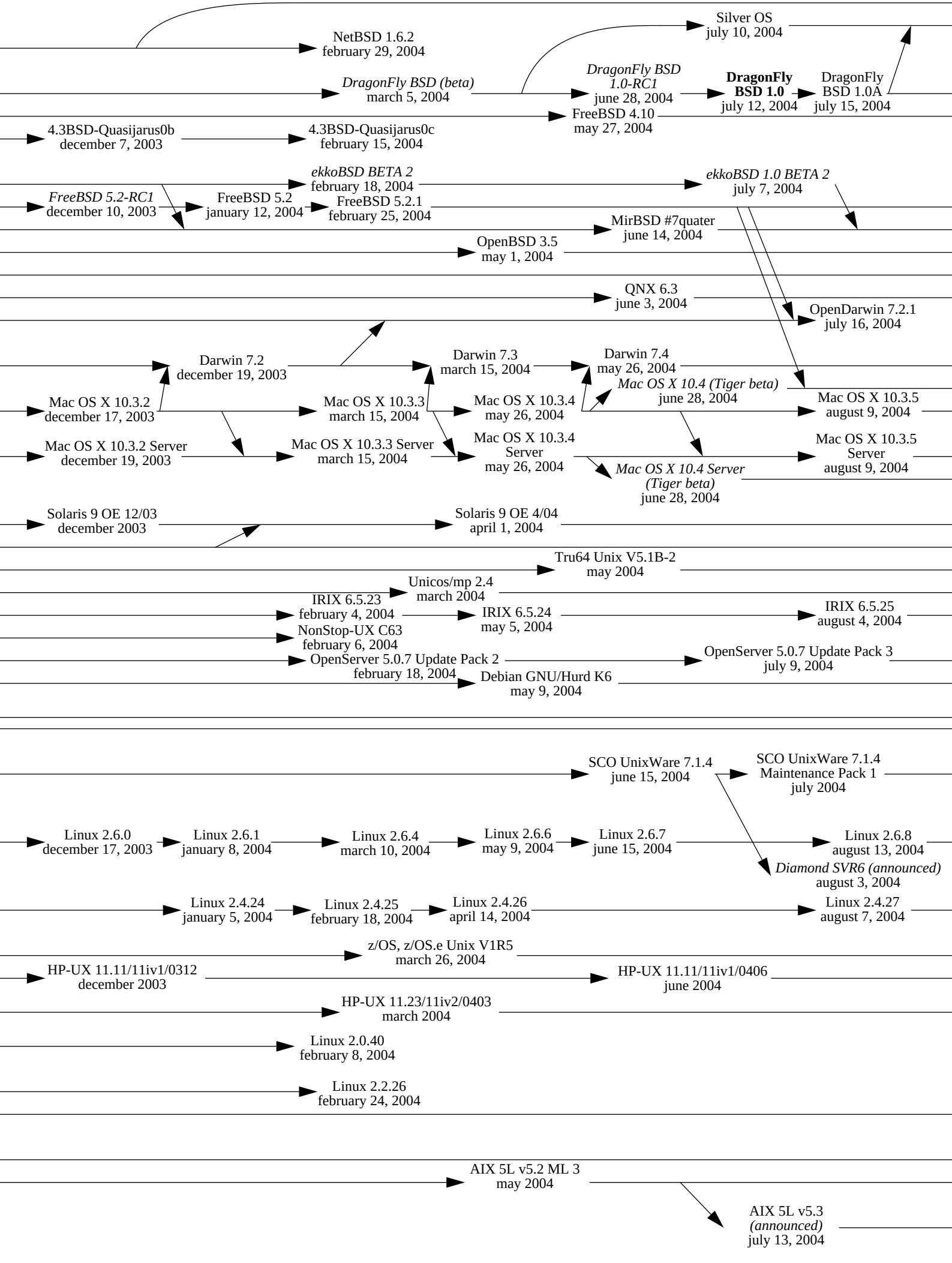

BSD/OS 4.3
february 14, 2002
BSD/OS 5.0
beta
NetBSD 1.6 beta
may 28, 2002
NetBSD 1.6
sept. 14, 2002
NetBSD 1.5.3
july 22, 2002
FreeBSD 4.5
january 29, 2002
FreeBSD 4.6
june 15, 2002
FreeBSD 4.6.2
august 15, 2002
FreeBSD 4.7
october 10, 2002
MicroBSD 0.1
july 14, 2002
MicroBSD 0.5
august 14, 2002
MicroBSD 0.6
october 12, 2002
FreeBSD 5.0
Developer Preview 1
april 8, 2002
MirBSD
august 29, 2002
MirBSD #0
october 11, 2002
OpenBSD 3.1
may 19, 2002
GNU-Darwin (beta 2.5)
march 12, 2002
QNX 6.2
(Momentics)
june 4, 2002
QNX 6.2 (patch A)
october 18, 2002
Darwin
5.2
Darwin 5.3
Darwin 5.4
Darwin 5.5
Darwin 6.0.1
sept. 23, 2002
Mac OS X 10.1.3
february 19, 2002
Mac OS X 10.1.4
april 17, 2002
Mac OS X
10.1.5
june 4, 2002
Mac OS X 10.2
(Jaguar)
august 13, 2002
Mac OS X
10.2.1
sept. 18, 2002
Solaris 8 2/02
february 2002
Mac OS X Server
10.1.2
january 17, 2002
Mac OS X Server
10.1.3
february 20, 2002
Mac OS X Server
10.1.4
april 15, 2002
Mac OS X Server
10.1.5
july 1, 2002
Mac OS X
Server 10.2
august 13, 2002
Mac OS X
Server 10.2.1
sept. 18, 2002
Solaris 9 OE
may 22, 2002
Solaris 9 OE 9/02
sept. 2002
Yamit (alpha)
may 5, 2002
Unicos/mp 1.0
august 23, 2002
IRIX 6.5.15
february 6, 2002
IRIX 6.5.16
may 8, 2002
IRIX 6.5.17
august 7, 2002
NonStop-UX C60
may 3, 2002
NonStop-UX C61
october 2, 2002
OpenServer 5.0.7 (announced)
august 26, 2002
Debian GNU/Hurd H3
february 26, 2002
Debian GNU/Hurd J1
august 5, 2002
Debian GNU/Hurd J2
october 10, 2002
GNU (GNU/Hurd,
GNU Mach 1.3)
may 27, 2002
Plan 9 r4
april 28, 2002
Open UNIX 8 MP3
Release 8.0
february 12, 2002
Open UNIX 8 MP4
Release 8.0
july 3, 2002
SCO UnixWare 7.1.3
(announced)
august 26, 2002
Unicos/mk 2.0.6
january 2002
Unicos 10.0.1.1
may 2002
Linux 2.4.18
february 25, 2002
Linux 2.4.19
august 3, 2002
z/OS, z/OS.e Unix V1R3
march 29, 2002
z/OS, z/OS.e Unix V1R4
september 27, 2002
HP-UX 11.11/11iv1/0203
march 2002
HP-UX 11.11/11iv1/0206
june 2002
HP-UX 11.11/11iv1/0209
september 2002
MkLinux Pre-R2
august 5, 2002
Linux 2.5.3
january 30, 2002
Linux 2.5.5
february 19, 2002
Linux 2.5.10
april 24, 2002
Linux 2.5.18
may 25, 2002
Linux 2.5.30
august 1, 2002
Linux 2.5.44
october 19, 2002
HP-UX 11.22 aka 11iv1.6 (IA)
august 2002
Linux 2.2.21
may 20, 2002
Linux 2.2.22
sept. 16, 2002
AIX 5L v5.2
october 18, 2002

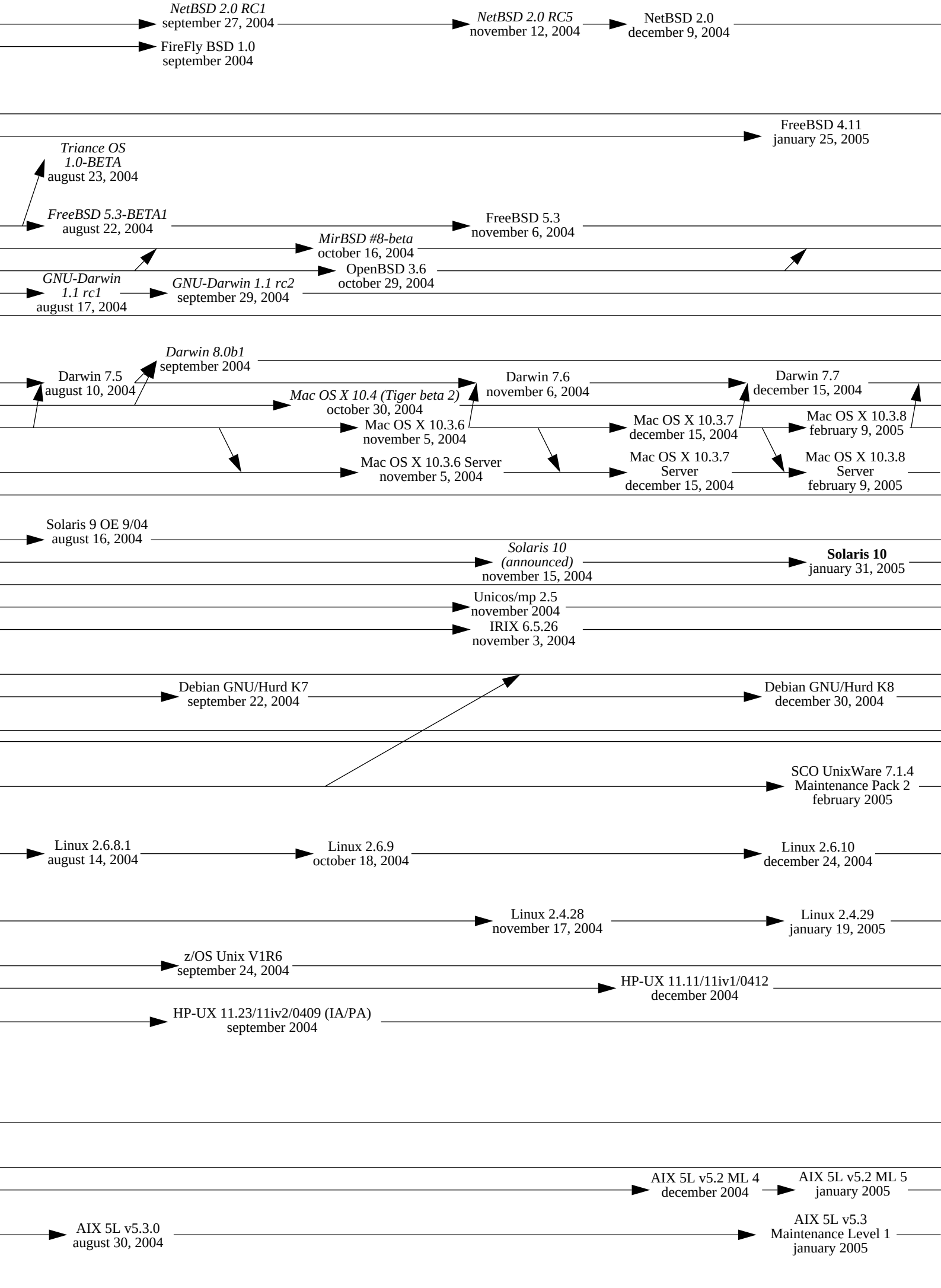
2003


BSD/OS 4.3.1
december 21, 2002
BSD/OS 5.0
may 2, 2003
NetBSD 1.6.1
april 14, 2003
FreeBSD 4.8
april 3, 2003
2.11BSD patch 444
february 10, 2003
FreeBSD 5.0 DP 2
november 18, 2002
FreeBSD 5.0
january 19, 2003
MirBSD #1
november 31, 2002
MirBSD #2
january 28, 2003
MirBSD #3
march 2, 2003
MirBSD #4
april 16, 2003
OpenBSD 3.2
november 1, 2002
OpenBSD 3.3
may 1, 2003
GNU-Darwin 1.0
january 10, 2003
QNX 6.2.1 (Momentics)
february 18, 2003
OpenDarwin-20030212
february 17, 2003
OpenDarwin
6.6.1
may 27, 2003
Darwin 6.0.2
oct. 28, 2002
Darwin
6.1
Darwin 6.2
Darwin 6.3
Darwin 6.4
Darwin 6.5
april 15, 2003
Darwin 6.6
may 14, 2003
Mac OS X 10.2.2
november 11, 2002
Mac OS X 10.2.3
december 19, 2002
Mac OS X 10.2.4
february 13, 2003
Mac OS X
10.2.5
april 10, 2003
Mac OS X
10.2.6
may 6, 2003
Solaris 8 12/02
december 2002
Mac OS X Server
10.2.2
november 11, 2002
Mac OS X Server
10.2.3
december 19, 2002
Mac OS X Server
10.2.4
february 24, 2003
Mac OS X Server
10.2.5
april 14, 2003
Mac OS X Server
10.2.6
may 8, 2003
Solaris 9 OE 12/02
december 2002
Solaris 9 x86 PE
february 6, 2003
Solaris 9 OE 4/03
april 2003
Tru64 Unix V5.1B
january 20, 2003
Unicos/mp 2.0
december 20, 2002
Unicos/mp 2.1
march 17, 2003
IRIX 6.5.18
november 8, 2002
IRIX 6.5.19
february 5, 2003
IRIX 6.5.20
may 7, 2003
NonStop-UX C62
january 17, 2003
OpenServer 5.0.7
february 24, 2003
Debian GNU/Hurd
K1-Unstable
december 12, 2002
Debian GNU/Hurd K2
march 3, 2003
Debian GNU/Hurd K3
april 30, 2003
GNU/Hurd-L4
(announced)
november 18, 2002
SCO UnixWare 7.1.3
december 4, 2002
SCO UnixWare 7.1.3
Update Pack 1
may 8, 2003
Unicos 10.0.1.2
may 2003
Linux 2.4.20
november 28, 2002
HP-UX 11.11/11iv1/0212
december 2002
HP-UX 11.11/11iv1/0303
march 2003
Linux 2.5.48
november 18, 2002
Linux 2.5.52
december 15, 2002
Linux 2.5.62
february 17, 2003
Linux 2.5.65
march 17, 2003
Linux 2.5.68
april 19, 2003
Linux 2.5.70
may 26, 2003
Linux 2.2.23
november 29, 2002
Linux 2.2.24
march 5, 2003
Linux 2.2.25
march 17, 2003
AIX 5L v5.2
Maintenance Level 1
may 2003
AIX 4.3.3 Maintenance Level 11
february 2003

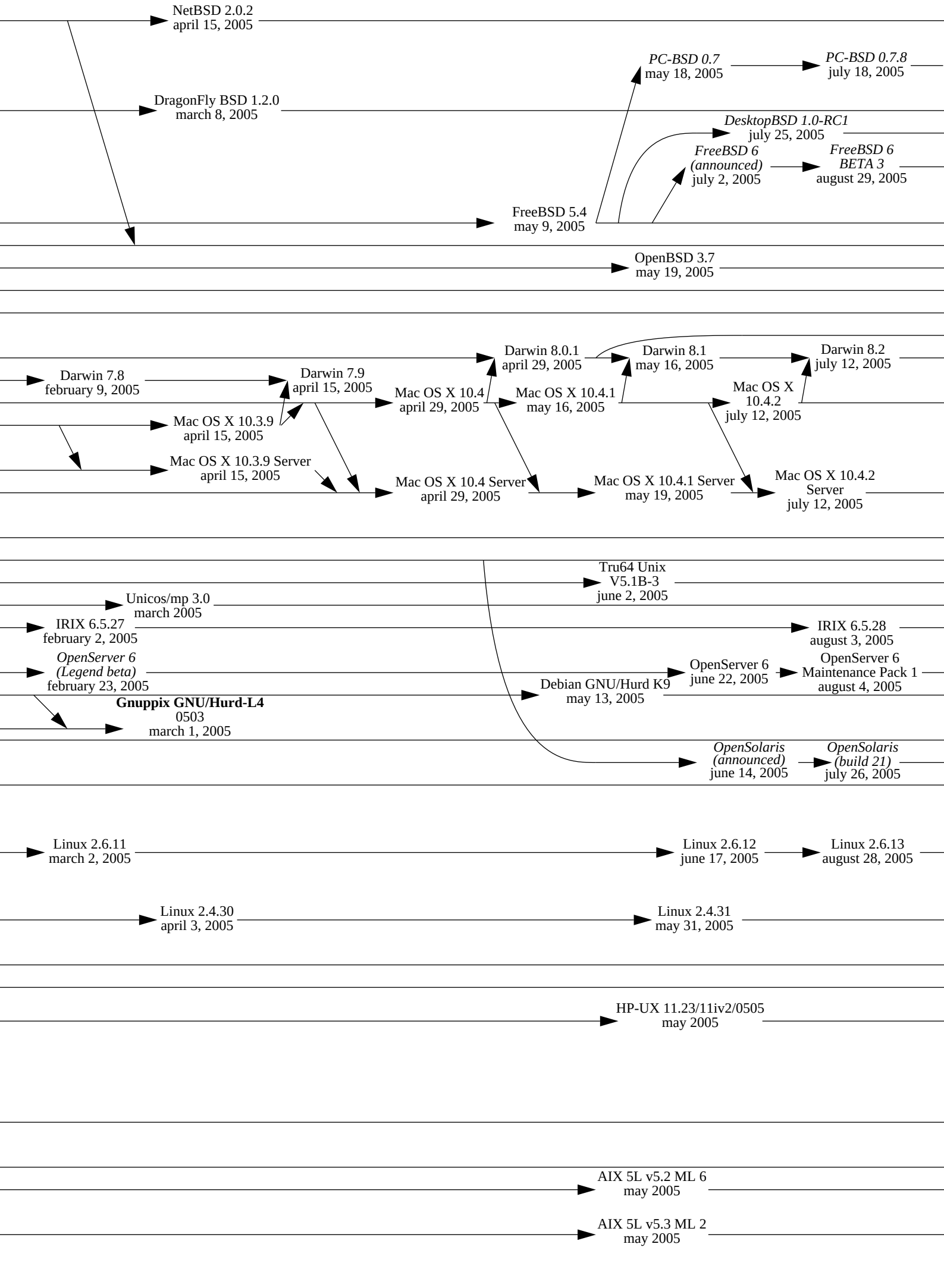

DragonFly BSD
july 16, 2003
FreeBSD 4.9
october 28, 2003
MicroBSD 0.7 beta
october 27, 2003
ekkoBSD 1.0 BETA1B
november 25, 2003
ekkoBSD
august 6, 2003
FreeBSD 5.1
june 9, 2003
FreeBSD 5.2-BETA
november 26, 2003
MirBSD #5
june 11, 2003
MirBSD #6
july 8, 2003
MirBSD #7semel
september 28, 2003
MirBSD #7bis
october 4, 2003
MirBSD #7ter
november 22, 2003
OpenBSD 3.4 beta
august 11, 2003
OpenBSD 3.4
november 1, 2003
GNU-Darwin 1.1
october 8, 2003
Darwin 6.7
sept. 22, 2003
Darwin 6.8
sept. 22, 2003
Darwin 7.0 Preview
june 25, 2003
Darwin 7.0
october 24, 2003
Darwin 7.0.1
november 14, 2003
Darwin 7.1
Mac OS X
10.2.7
august 18, 2003
Mac OS X
10.2.8
september 22, 2003
Mac OS X 10.3 beta (Panther)
june 23, 2003
Mac OS X 10.3
october 24, 2003
Mac OS X 10.3.1
november 10, 2003
Mac OS X Server 10.3 beta (Panther)
june 23, 2003
Mac OS X 10.3
Server
october 24, 2003
Mac OS X 10.3.1
Server
november 10, 2003
Mac OS X Server
10.2.8
september 22, 2003
Solaris 9 OE 8/03
july 29, 2003
Solaris 10 Preview
july 29, 2003
Tru64 Unix V5.1B-1
october 20, 2003
Unicos/mp 2.2
july 2003
Unicos/mp 2.3
october 2003
IRIX 6.5.21
august 6, 2003
IRIX 6.5.22
november 5, 2003
OpenServer 5.0.7 Update Pack 1
july 31, 2003
Debian GNU/Hurd K4
july 29, 2003
Debian GNU/Hurd K5
november 24, 2003
SCO UnixWare 7.1.3 /OKP
july 31, 2003
Linux 2.6.0-test1
july 13, 2003
Linux 2.6.0-test11
november 26, 2003
Linux 2.4.21
june 13, 2003
Linux 2.4.22
august 25, 2003
Linux 2.4.23
november 28, 2003
HP-UX 11.11/11iv1
0306
june 2003
HP-UX 11.11/11iv1/0309
septembre 2003
HP-UX 11.23 aka 11iv2 (IA)
september 2003
Linux 2.5.75
july 10, 2003
Minix 2.0.4
november 23, 2003
AIX 5L v5.2 ML 2
october 2003

2004

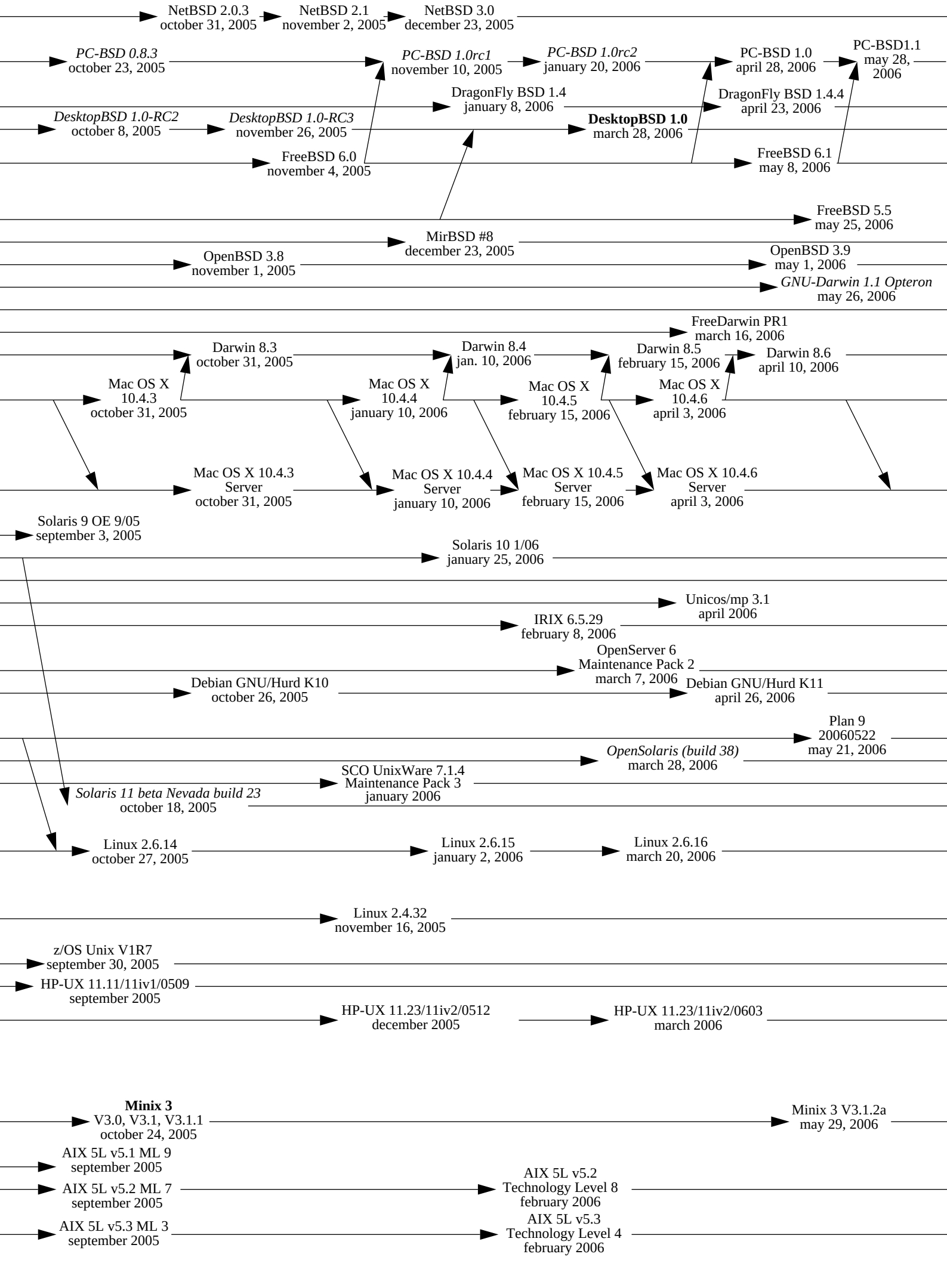

Silver OS
july 10, 2004
NetBSD 1.6.2
february 29, 2004
DragonFly BSD (beta)
march 5, 2004
DragonFly BSD 1.0-RC1
june 28, 2004
DragonFly BSD 1.0
july 12, 2004
DragonFly BSD 1.0A
july 15, 2004
FreeBSD 4.10
may 27, 2004
4.3BSD-Quasijarus0b
december 7, 2003
4.3BSD-Quasijarus0c
february 15, 2004
ekkoBSD BETA 2
february 18, 2004
ekkoBSD 1.0 BETA 2
july 7, 2004
FreeBSD 5.2-RC1
december 10, 2003
FreeBSD 5.2
january 12, 2004
FreeBSD 5.2.1
february 25, 2004
MirBSD #7quater
june 14, 2004
OpenBSD 3.5
may 1, 2004
QNX 6.3
june 3, 2004
OpenDarwin 7.2.1
july 16, 2004
Darwin 7.2
december 19, 2003
Darwin 7.3
march 15, 2004
Darwin 7.4
may 26, 2004
Mac OS X 10.4 (Tiger beta)
june 28, 2004
Mac OS X 10.3.2
december 17, 2003
Mac OS X 10.3.3
march 15, 2004
Mac OS X 10.3.4
may 26, 2004
Mac OS X 10.3.5
august 9, 2004
Mac OS X 10.3.2 Server
december 19, 2003
Mac OS X 10.3.3 Server
march 15, 2004
Mac OS X 10.3.4 Server
may 26, 2004
Mac OS X 10.3.5 Server
august 9, 2004
Mac OS X 10.4 Server (Tiger beta)
june 28, 2004
Solaris 9 OE 12/03
december 2003
Solaris 9 OE 4/04
april 1, 2004
Tru64 Unix V5.1B-2
may 2004
Unicos/mp 2.4
march 2004
IRIX 6.5.23
february 4, 2004
IRIX 6.5.24
may 5, 2004
IRIX 6.5.25
august 4, 2004
NonStop-UX C63
february 6, 2004
OpenServer 5.0.7 Update Pack 2
february 18, 2004
OpenServer 5.0.7 Update Pack 3
july 9, 2004
Debian GNU/Hurd K6
may 9, 2004
SCO UnixWare 7.1.4
june 15, 2004
SCO UnixWare 7.1.4 Maintenance Pack 1
july 2004
Linux 2.6.0
december 17, 2003
Linux 2.6.1
january 8, 2004
Linux 2.6.4
march 10, 2004
Linux 2.6.6
may 9, 2004
Linux 2.6.7
june 15, 2004
Linux 2.6.8
august 13, 2004
Diamond SVR6 (announced)
august 3, 2004
Linux 2.4.24
january 5, 2004
Linux 2.4.25
february 18, 2004
Linux 2.4.26
april 14, 2004
Linux 2.4.27
august 7, 2004
z/OS, z/OS.e Unix V1R5
march 26, 2004
HP-UX 11.11/11iv1/0312
december 2003
HP-UX 11.11/11iv1/0406
june 2004
HP-UX 11.23/11iv2/0403
march 2004
Linux 2.0.40
february 8, 2004
Linux 2.2.26
february 24, 2004
AIX 5L v5.2 ML 3
may 2004
AIX 5L v5.3 (announced)
july 13, 2004

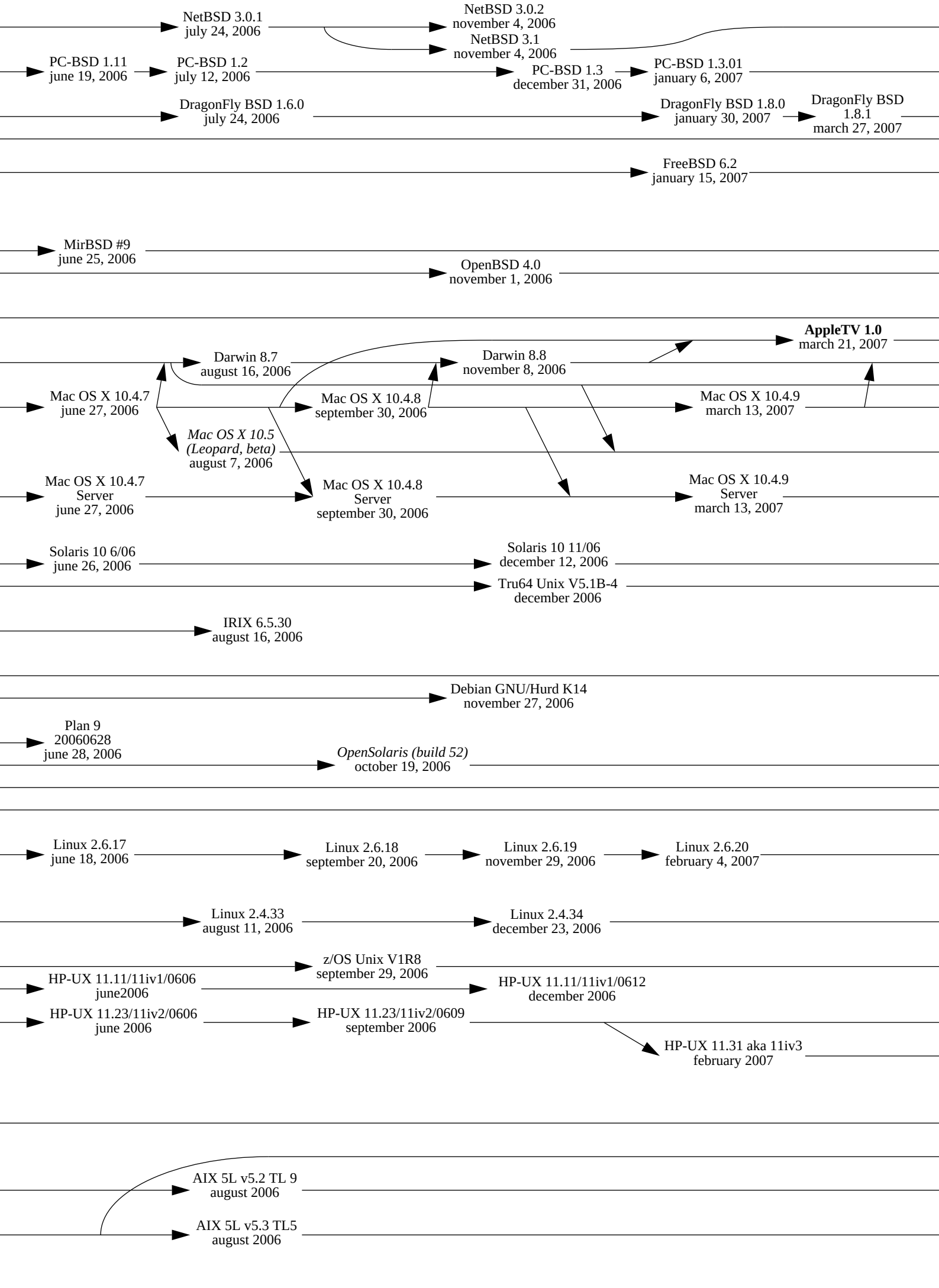

NetBSD 2.0 RC1
september 27, 2004
NetBSD 2.0 RC5
november 12, 2004
NetBSD 2.0
december 9, 2004
FireFly BSD 1.0
september 2004
FreeBSD 4.11
january 25, 2005
Triance OS 1.0-BETA
august 23, 2004
FreeBSD 5.3-BETA1
august 22, 2004
FreeBSD 5.3
november 6, 2004
MirBSD #8-beta
october 16, 2004
OpenBSD 3.6
october 29, 2004
GNU-Darwin 1.1 rc1
august 17, 2004
GNU-Darwin 1.1 rc2
september 29, 2004
Darwin 8.0b1
september 2004
Darwin 7.5
august 10, 2004
Darwin 7.6
november 6, 2004
Darwin 7.7
december 15, 2004
Mac OS X 10.4 (Tiger beta 2)
october 30, 2004
Mac OS X 10.3.6
november 5, 2004
Mac OS X 10.3.7
december 15, 2004
Mac OS X 10.3.8
february 9, 2005
Mac OS X 10.3.6 Server
november 5, 2004
Mac OS X 10.3.7 Server
december 15, 2004
Mac OS X 10.3.8 Server
february 9, 2005
Solaris 9 OE 9/04
august 16, 2004
Solaris 10 (announced)
november 15, 2004
Solaris 10
january 31, 2005
Unicos/mp 2.5
november 2004
IRIX 6.5.26
november 3, 2004
Debian GNU/Hurd K7
september 22, 2004
Debian GNU/Hurd K8
december 30, 2004
SCO UnixWare 7.1.4
Maintenance Pack 2
february 2005
Linux 2.6.8.1
august 14, 2004
Linux 2.6.9
october 18, 2004
Linux 2.6.10
december 24, 2004
Linux 2.4.28
november 17, 2004
Linux 2.4.29
january 19, 2005
z/OS Unix V1R6
september 24, 2004
HP-UX 11.11/11iv1/0412
december 2004
HP-UX 11.23/11iv2/0409 (IA/PA)
september 2004
AIX 5L v5.2 ML 4
december 2004
AIX 5L v5.2 ML 5
january 2005
AIX 5L v5.3.0
august 30, 2004
AIX 5L v5.3
Maintenance Level 1
january 2005

2005

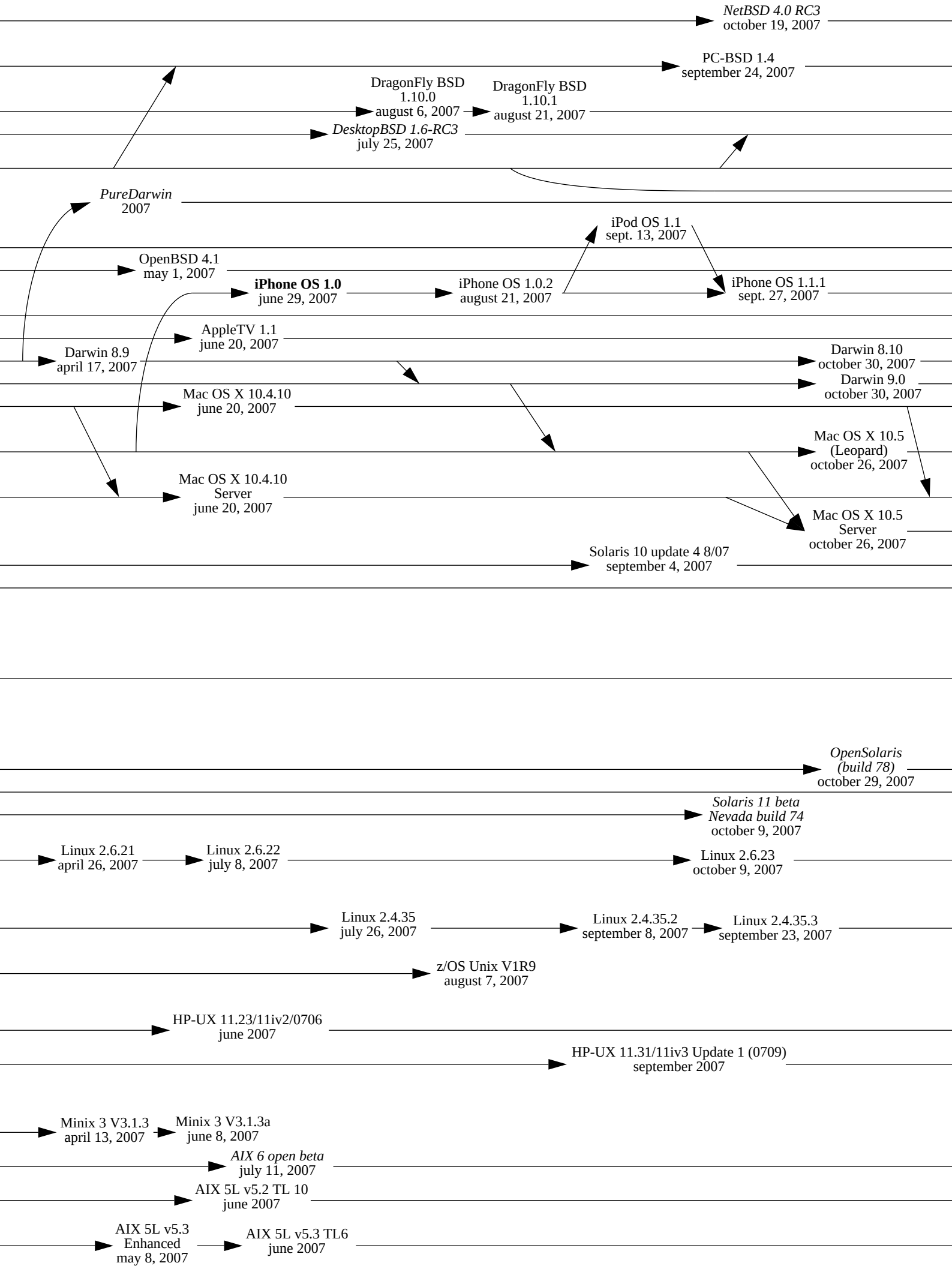

NetBSD 2.0.2
april 15, 2005
DragonFly BSD 1.2.0
march 8, 2005
PC-BSD 0.7
may 18, 2005
PC-BSD 0.7.8
july 18, 2005
DesktopBSD 1.0-RC1
july 25, 2005
FreeBSD 6
(announced)
july 2, 2005
FreeBSD 6
BETA 3
august 29, 2005
FreeBSD 5.4
may 9, 2005
OpenBSD 3.7
may 19, 2005
Darwin 8.0.1
april 29, 2005
Darwin 8.1
may 16, 2005
Darwin 8.2
july 12, 2005
Darwin 7.8
february 9, 2005
Darwin 7.9
april 15, 2005
Mac OS X 10.4
april 29, 2005
Mac OS X 10.4.1
may 16, 2005
Mac OS X
10.4.2
july 12, 2005
Mac OS X 10.3.9
april 15, 2005
Mac OS X 10.3.9 Server
april 15, 2005
Mac OS X 10.4 Server
april 29, 2005
Mac OS X 10.4.1 Server
may 19, 2005
Mac OS X 10.4.2
Server
july 12, 2005
Tru64 Unix
V5.1B-3
june 2, 2005
Unicos/mp 3.0
march 2005
IRIX 6.5.27
february 2, 2005
IRIX 6.5.28
august 3, 2005
OpenServer 6
(Legend beta)
february 23, 2005
OpenServer 6
june 22, 2005
OpenServer 6
Maintenance Pack 1
august 4, 2005
Debian GNU/Hurd K9
may 13, 2005
Gnuppix GNU/Hurd-L4
0503
march 1, 2005
OpenSolaris
(announced)
june 14, 2005
OpenSolaris
(build 21)
july 26, 2005
Linux 2.6.11
march 2, 2005
Linux 2.6.12
june 17, 2005
Linux 2.6.13
august 28, 2005
Linux 2.4.30
april 3, 2005
Linux 2.4.31
may 31, 2005
HP-UX 11.23/11iv2/0505
may 2005
AIX 5L v5.2 ML 6
may 2005
AIX 5L v5.3 ML 2
may 2005

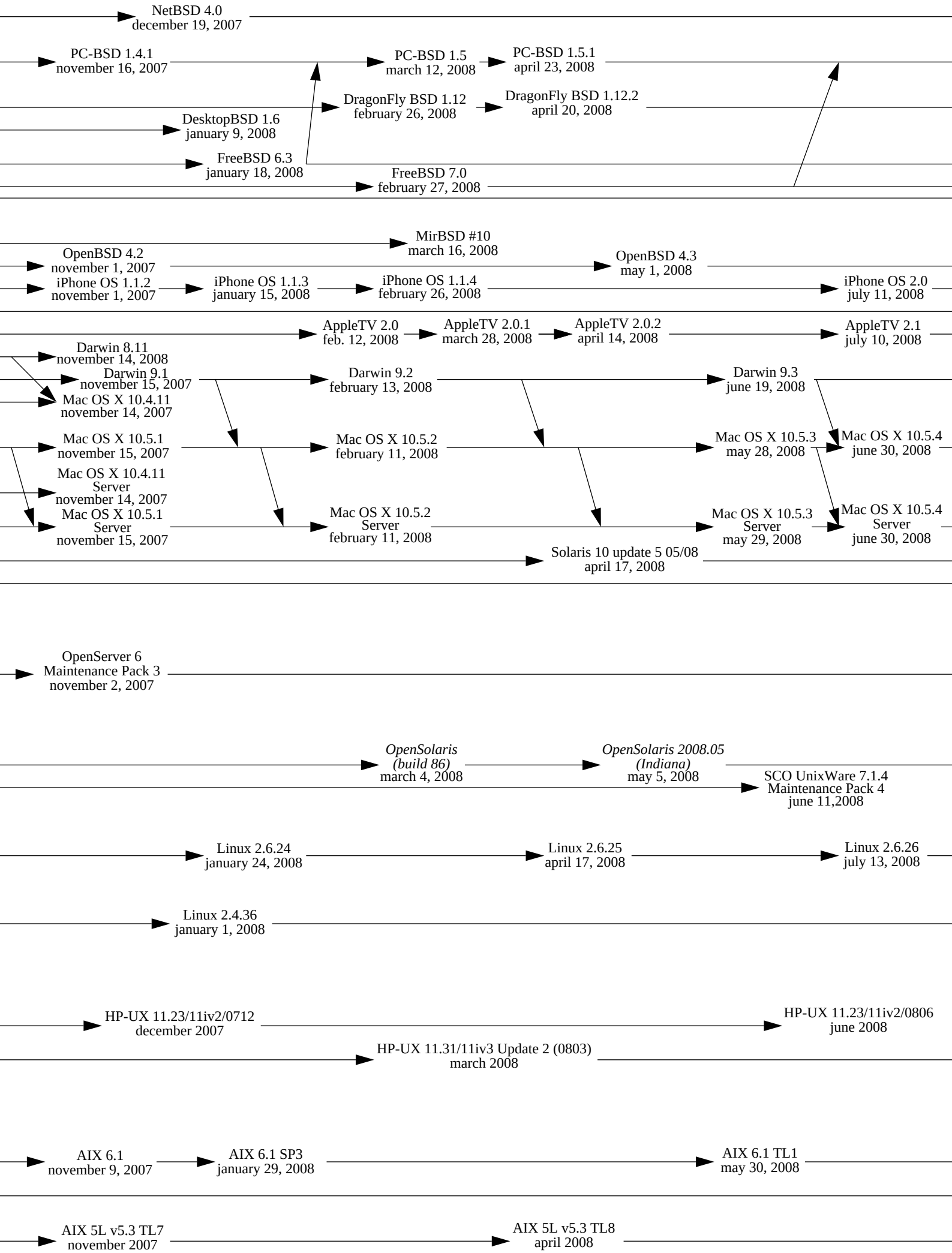
2006


NetBSD 2.0.3
october 31, 2005
NetBSD 2.1
november 2, 2005
NetBSD 3.0
december 23, 2005
PC-BSD 0.8.3
october 23, 2005
PC-BSD 1.0rc1
november 10, 2005
PC-BSD 1.0rc2
january 20, 2006
PC-BSD 1.0
april 28, 2006
PC-BSD1.1
may 28, 2006
DragonFly BSD 1.4
january 8, 2006
DragonFly BSD 1.4.4
april 23, 2006
DesktopBSD 1.0-RC2
october 8, 2005
DesktopBSD 1.0-RC3
november 26, 2005
DesktopBSD 1.0
march 28, 2006
FreeBSD 6.0
november 4, 2005
FreeBSD 6.1
may 8, 2006
FreeBSD 5.5
may 25, 2006
MirBSD #8
december 23, 2005
OpenBSD 3.8
november 1, 2005
OpenBSD 3.9
may 1, 2006
GNU-Darwin 1.1 Opteron
may 26, 2006
FreeDarwin PR1
march 16, 2006
Darwin 8.3
october 31, 2005
Darwin 8.4
jan. 10, 2006
Darwin 8.5
february 15, 2006
Darwin 8.6
april 10, 2006
Mac OS X
10.4.3
october 31, 2005
Mac OS X
10.4.4
january 10, 2006
Mac OS X
10.4.5
february 15, 2006
Mac OS X
10.4.6
april 3, 2006
Mac OS X 10.4.3
Server
october 31, 2005
Mac OS X 10.4.4
Server
january 10, 2006
Mac OS X 10.4.5
Server
february 15, 2006
Mac OS X 10.4.6
Server
april 3, 2006
Solaris 9 OE 9/05
september 3, 2005
Solaris 10 1/06
january 25, 2006
Unicos/mp 3.1
april 2006
IRIX 6.5.29
february 8, 2006
OpenServer 6
Maintenance Pack 2
march 7, 2006
Debian GNU/Hurd K10
october 26, 2005
Debian GNU/Hurd K11
april 26, 2006
Plan 9
20060522
may 21, 2006
OpenSolaris (build 38)
march 28, 2006
SCO UnixWare 7.1.4
Maintenance Pack 3
january 2006
Solaris 11 beta Nevada build 23
october 18, 2005
Linux 2.6.14
october 27, 2005
Linux 2.6.15
january 2, 2006
Linux 2.6.16
march 20, 2006
Linux 2.4.32
november 16, 2005
z/OS Unix V1R7
september 30, 2005
HP-UX 11.11/11iv1/0509
september 2005
HP-UX 11.23/11iv2/0512
december 2005
HP-UX 11.23/11iv2/0603
march 2006
Minix 3
V3.0, V3.1, V3.1.1
october 24, 2005
Minix 3 V3.1.2a
may 29, 2006
AIX 5L v5.1 ML 9
september 2005
AIX 5L v5.2 ML 7
september 2005
AIX 5L v5.2
Technology Level 8
february 2006
AIX 5L v5.3 ML 3
september 2005
AIX 5L v5.3
Technology Level 4
february 2006

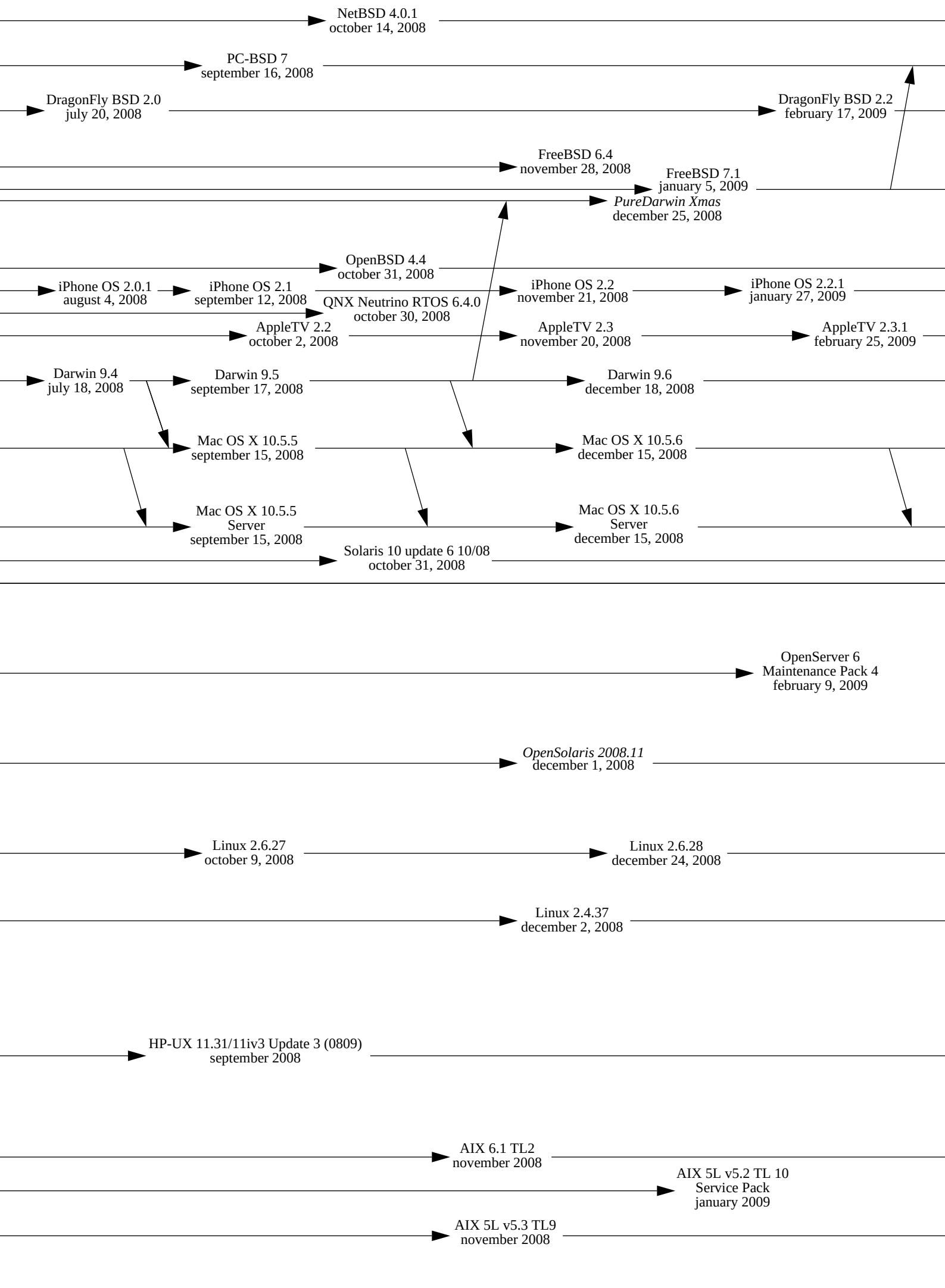

NetBSD 3.0.1
july 24, 2006
NetBSD 3.0.2
november 4, 2006
NetBSD 3.1
november 4, 2006
PC-BSD 1.11
june 19, 2006
PC-BSD 1.2
july 12, 2006
PC-BSD 1.3
december 31, 2006
PC-BSD 1.3.01
january 6, 2007
DragonFly BSD 1.6.0
july 24, 2006
DragonFly BSD 1.8.0
january 30, 2007
DragonFly BSD
1.8.1
march 27, 2007
FreeBSD 6.2
january 15, 2007
MirBSD #9
june 25, 2006
OpenBSD 4.0
november 1, 2006
AppleTV 1.0
march 21, 2007
Darwin 8.7
august 16, 2006
Darwin 8.8
november 8, 2006
Mac OS X 10.4.7
june 27, 2006
Mac OS X 10.4.8
september 30, 2006
Mac OS X 10.4.9
march 13, 2007
Mac OS X 10.5
(Leopard, beta)
august 7, 2006
Mac OS X 10.4.7
Server
june 27, 2006
Mac OS X 10.4.8
Server
september 30, 2006
Mac OS X 10.4.9
Server
march 13, 2007
Solaris 10 6/06
june 26, 2006
Solaris 10 11/06
december 12, 2006
Tru64 Unix V5.1B-4
december 2006
IRIX 6.5.30
august 16, 2006
Debian GNU/Hurd K14
november 27, 2006
Plan 9
20060628
june 28, 2006
OpenSolaris (build 52)
october 19, 2006
Linux 2.6.17
june 18, 2006
Linux 2.6.18
september 20, 2006
Linux 2.6.19
november 29, 2006
Linux 2.6.20
february 4, 2007
Linux 2.4.33
august 11, 2006
Linux 2.4.34
december 23, 2006
z/OS Unix V1R8
september 29, 2006
HP-UX 11.11/11iv1/0606
june2006
HP-UX 11.11/11iv1/0612
december 2006
HP-UX 11.23/11iv2/0606
june 2006
HP-UX 11.23/11iv2/0609
september 2006
HP-UX 11.31 aka 11iv3
february 2007
AIX 5L v5.2 TL 9
august 2006
AIX 5L v5.3 TL5
august 2006

2007

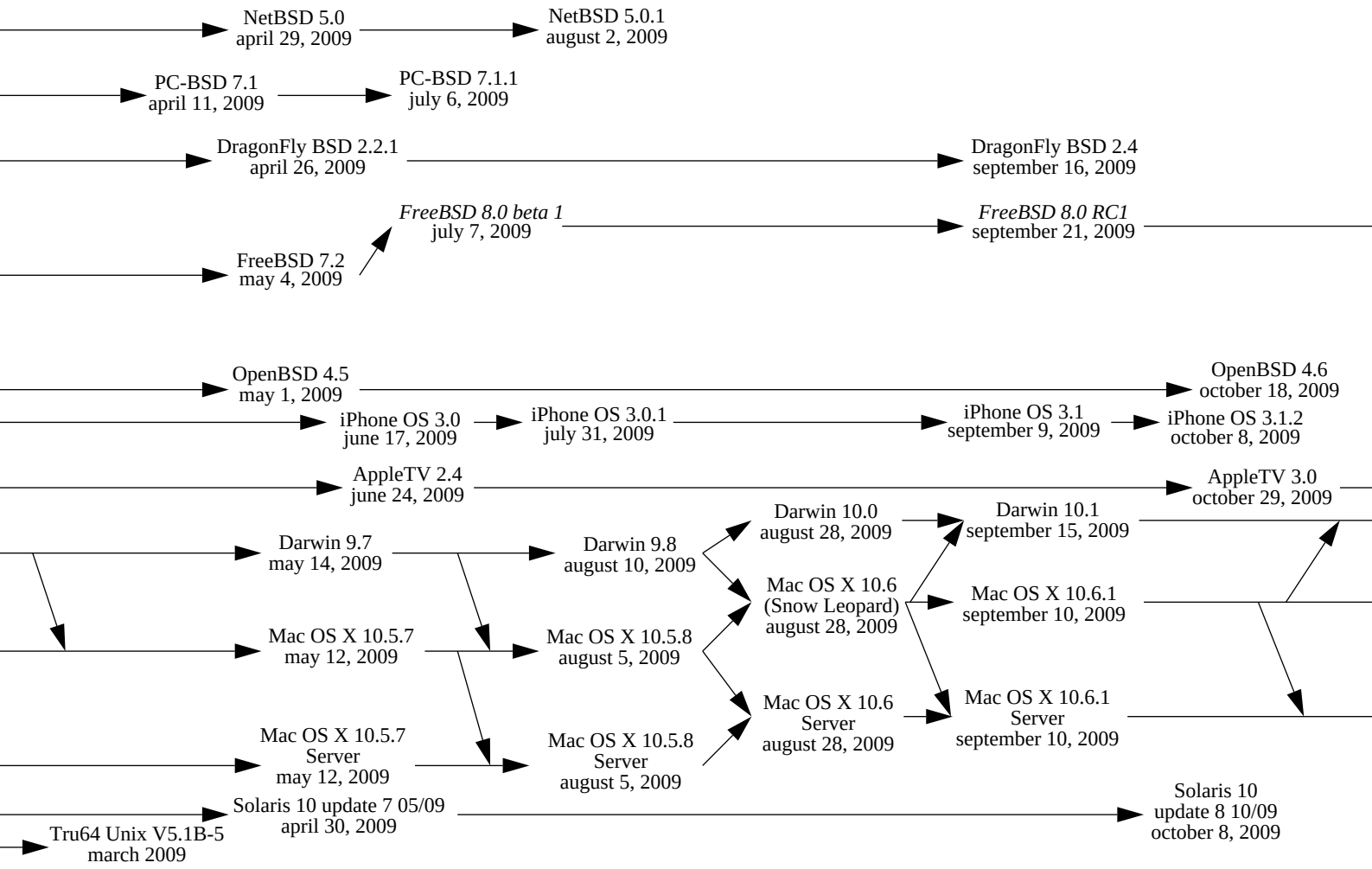

NetBSD 4.0 RC3
october 19, 2007
PC-BSD 1.4
september 24, 2007
DragonFly BSD
1.10.0
august 6, 2007
DragonFly BSD
1.10.1
august 21, 2007
DesktopBSD 1.6-RC3
july 25, 2007
PureDarwin
2007
iPod OS 1.1
sept. 13, 2007
OpenBSD 4.1
may 1, 2007
iPhone OS 1.0
june 29, 2007
iPhone OS 1.0.2
august 21, 2007
iPhone OS 1.1.1
sept. 27, 2007
AppleTV 1.1
june 20, 2007
Darwin 8.9
april 17, 2007
Darwin 8.10
october 30, 2007
Darwin 9.0
october 30, 2007
Mac OS X 10.4.10
june 20, 2007
Mac OS X 10.5
(Leopard)
october 26, 2007
Mac OS X 10.4.10
Server
june 20, 2007
Mac OS X 10.5
Server
october 26, 2007
Solaris 10 update 4 8/07
september 4, 2007
OpenSolaris
(build 78)
october 29, 2007
Solaris 11 beta
Nevada build 74
october 9, 2007
Linux 2.6.21
april 26, 2007
Linux 2.6.22
july 8, 2007
Linux 2.6.23
october 9, 2007
Linux 2.4.35
july 26, 2007
Linux 2.4.35.2
september 8, 2007
Linux 2.4.35.3
september 23, 2007
z/OS Unix V1R9
august 7, 2007
HP-UX 11.23/11iv2/0706
june 2007
HP-UX 11.31/11iv3 Update 1 (0709)
september 2007
Minix 3 V3.1.3
april 13, 2007
Minix 3 V3.1.3a
june 8, 2007
AIX 6 open beta
july 11, 2007
AIX 5L v5.2 TL 10
june 2007
AIX 5L v5.3
Enhanced
may 8, 2007
AIX 5L v5.3 TL6
june 2007

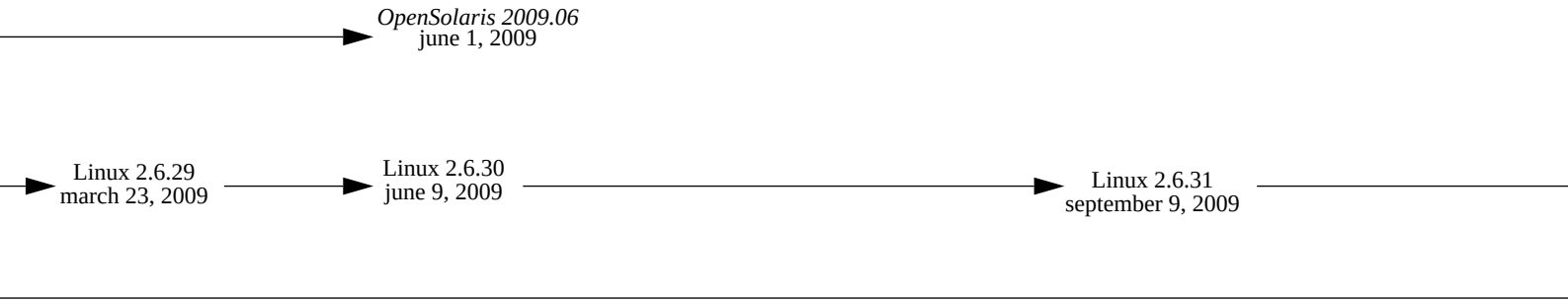

NetBSD 4.0
december 19, 2007
PC-BSD 1.4.1
november 16, 2007
PC-BSD 1.5
march 12, 2008
PC-BSD 1.5.1
april 23, 2008
DragonFly BSD 1.12
february 26, 2008
DragonFly BSD 1.12.2
april 20, 2008
DesktopBSD 1.6
january 9, 2008
FreeBSD 6.3
january 18, 2008
FreeBSD 7.0
february 27, 2008
MirBSD #10
march 16, 2008
OpenBSD 4.2
november 1, 2007
OpenBSD 4.3
may 1, 2008
iPhone OS 1.1.2
november 1, 2007
iPhone OS 1.1.3
january 15, 2008
iPhone OS 1.1.4
february 26, 2008
iPhone OS 2.0
july 11, 2008
AppleTV 2.0
feb. 12, 2008
AppleTV 2.0.1
march 28, 2008
AppleTV 2.0.2
april 14, 2008
AppleTV 2.1
july 10, 2008
Darwin 8.11
november 14, 2008
Darwin 9.1
november 15, 2007
Darwin 9.2
february 13, 2008
Darwin 9.3
june 19, 2008
Mac OS X 10.4.11
november 14, 2007
Mac OS X 10.5.1
november 15, 2007
Mac OS X 10.5.2
february 11, 2008
Mac OS X 10.5.3
may 28, 2008
Mac OS X 10.5.4
june 30, 2008
Mac OS X 10.4.11
Server
november 14, 2007
Mac OS X 10.5.1
Server
november 15, 2007
Mac OS X 10.5.2
Server
february 11, 2008
Mac OS X 10.5.3
Server
may 29, 2008
Mac OS X 10.5.4
Server
june 30, 2008
Solaris 10 update 5 05/08
april 17, 2008
OpenServer 6
Maintenance Pack 3
november 2, 2007
OpenSolaris
(build 86)
march 4, 2008
OpenSolaris 2008.05
(Indiana)
may 5, 2008
SCO UnixWare 7.1.4
Maintenance Pack 4
june 11,2008
Linux 2.6.24
january 24, 2008
Linux 2.6.25
april 17, 2008
Linux 2.6.26
july 13, 2008
Linux 2.4.36
january 1, 2008
HP-UX 11.23/11iv2/0712
december 2007
HP-UX 11.23/11iv2/0806
june 2008
HP-UX 11.31/11iv3 Update 2 (0803)
march 2008
AIX 6.1
november 9, 2007
AIX 6.1 SP3
january 29, 2008
AIX 6.1 TL1
may 30, 2008
AIX 5L v5.3 TL7
november 2007
AIX 5L v5.3 TL8
april 2008

2008


NetBSD 4.0.1
october 14, 2008
PC-BSD 7
september 16, 2008
DragonFly BSD 2.0
july 20, 2008
DragonFly BSD 2.2
february 17, 2009
FreeBSD 6.4
november 28, 2008
FreeBSD 7.1
january 5, 2009
PureDarwin Xmas
december 25, 2008
OpenBSD 4.4
october 31, 2008
iPhone OS 2.0.1
august 4, 2008
iPhone OS 2.1
september 12, 2008
iPhone OS 2.2
november 21, 2008
iPhone OS 2.2.1
january 27, 2009
QNX Neutrino RTOS 6.4.0
october 30, 2008
AppleTV 2.2
october 2, 2008
AppleTV 2.3
november 20, 2008
AppleTV 2.3.1
february 25, 2009
Darwin 9.4
july 18, 2008
Darwin 9.5
september 17, 2008
Darwin 9.6
december 18, 2008
Mac OS X 10.5.5
september 15, 2008
Mac OS X 10.5.6
december 15, 2008
Mac OS X 10.5.5
Server
september 15, 2008
Mac OS X 10.5.6
Server
december 15, 2008
Solaris 10 update 6 10/08
october 31, 2008
OpenServer 6
Maintenance Pack 4
february 9, 2009
OpenSolaris 2008.11
december 1, 2008
Linux 2.6.27
october 9, 2008
Linux 2.6.28
december 24, 2008
Linux 2.4.37
december 2, 2008
HP-UX 11.31/11iv3 Update 3 (0809)
september 2008
AIX 6.1 TL2
november 2008
AIX 5L v5.2 TL 10
Service Pack
january 2009
AIX 5L v5.3 TL9
november 2008

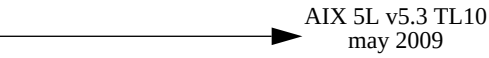
2009

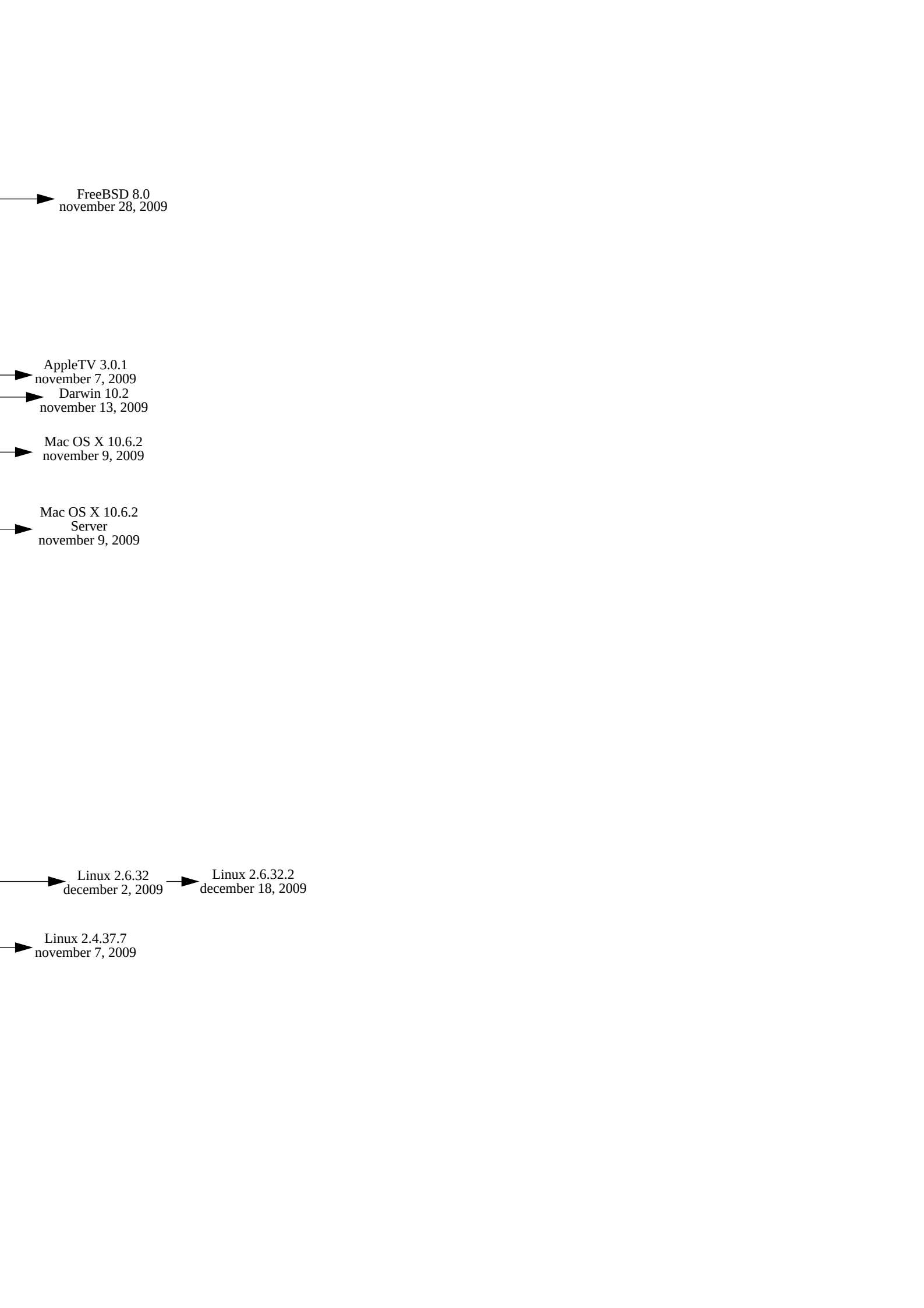

NetBSD 5.0
april 29, 2009
NetBSD 5.0.1
august 2, 2009
PC-BSD 7.1
april 11, 2009
PC-BSD 7.1.1
july 6, 2009
DragonFly BSD 2.2.1
april 26, 2009
DragonFly BSD 2.4
september 16, 2009
FreeBSD 7.2
may 4, 2009
FreeBSD 8.0 beta 1
july 7, 2009
FreeBSD 8.0 RC1
september 21, 2009
OpenBSD 4.5
may 1, 2009
OpenBSD 4.6
october 18, 2009
iPhone OS 3.0
june 17, 2009
iPhone OS 3.0.1
july 31, 2009
iPhone OS 3.1
september 9, 2009
iPhone OS 3.1.2
october 8, 2009
AppleTV 2.4
june 24, 2009
AppleTV 3.0
october 29, 2009
Darwin 9.7
may 14, 2009
Darwin 9.8
august 10, 2009
Darwin 10.0
august 28, 2009
Darwin 10.1
september 15, 2009
Mac OS X 10.5.7
may 12, 2009
Mac OS X 10.5.8
august 5, 2009
Mac OS X 10.6
(Snow Leopard)
august 28, 2009
Mac OS X 10.6.1
september 10, 2009
Mac OS X 10.5.7
Server
may 12, 2009
Mac OS X 10.5.8
Server
august 5, 2009
Mac OS X 10.6
Server
august 28, 2009
Mac OS X 10.6.1
Server
september 10, 2009
Solaris 10 update 7 05/09
april 30, 2009
Solaris 10
update 8 10/09
october 8, 2009
Tru64 Unix V5.1B-5
march 2009


OpenSolaris 2009.06
june 1, 2009
Linux 2.6.29
march 23, 2009
Linux 2.6.30
june 9, 2009
Linux 2.6.31
september 9, 2009


HP-UX 11.31/11iv3
Update 4 (0903)
april 9, 2009
HP-UX 11.31/11iv3
Update 5 (0909)
september 2009

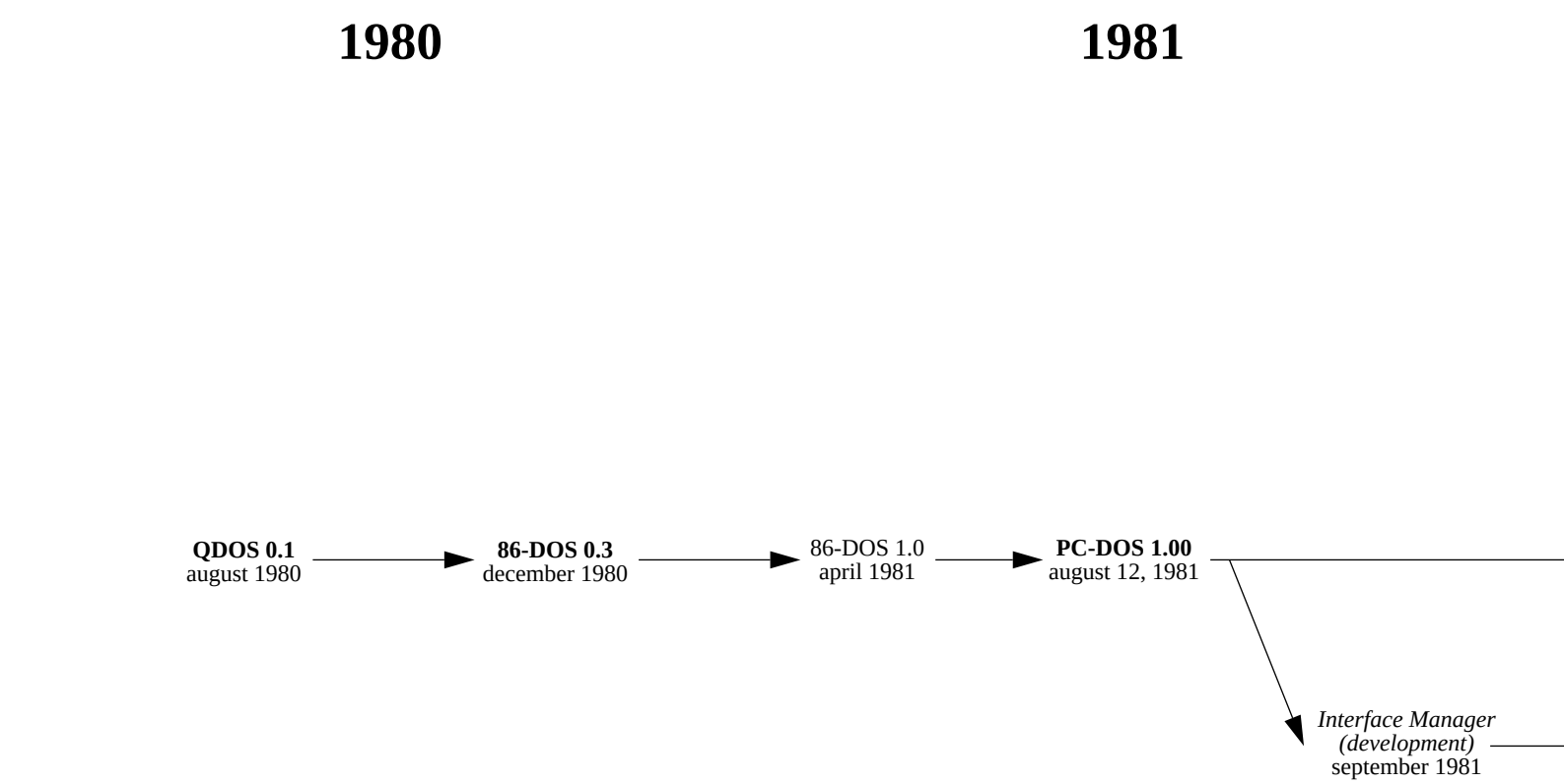

AIX 6.1 TL3
may 2009


AIX 5L v5.3 TL10
may 2009


FreeBSD 8.0
november 28, 2009
AppleTV 3.0.1
november 7, 2009
Darwin 10.2
november 13, 2009
Mac OS X 10.6.2
november 9, 2009
Mac OS X 10.6.2
Server
november 9, 2009
Linux 2.6.32
december 2, 2009
Linux 2.6.32.2
december 18, 2009
Linux 2.4.37.7
november 7, 2009

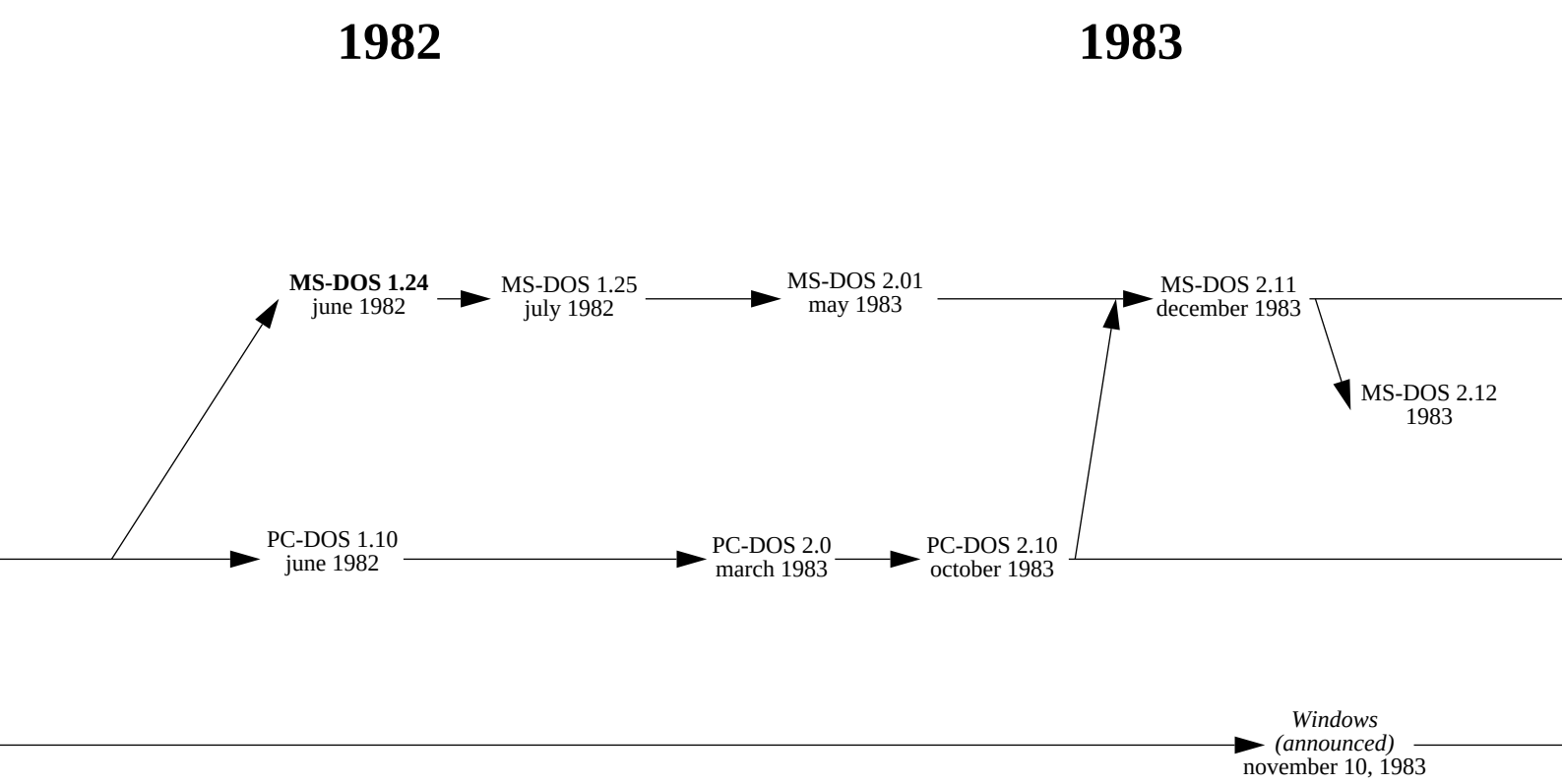
Ci joint dans la version imprimée de ce cours, un historique des différentes version de WINDOWS.

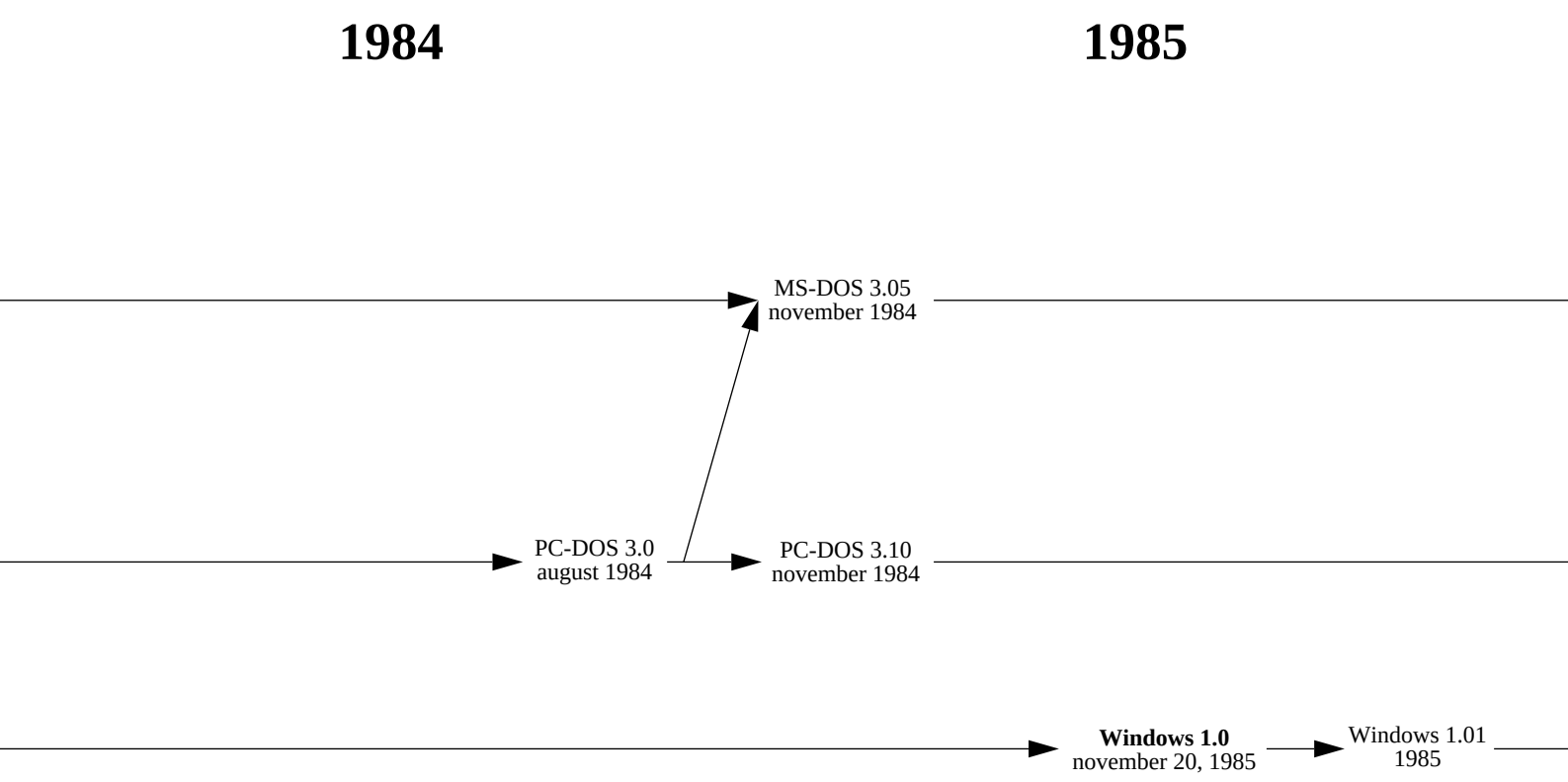
Adresse web du document fourni : « `http://www.levenez.com/unix/` »

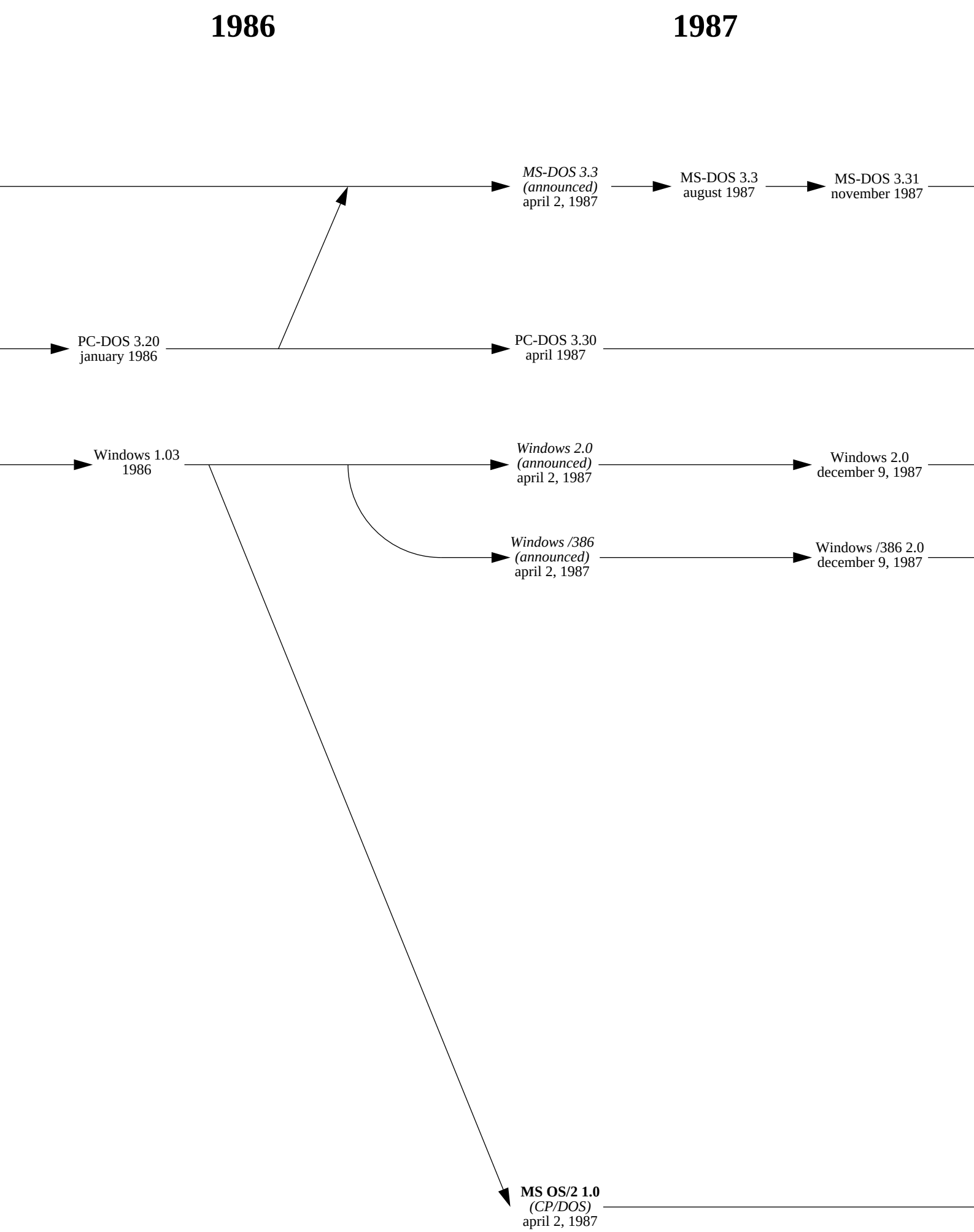


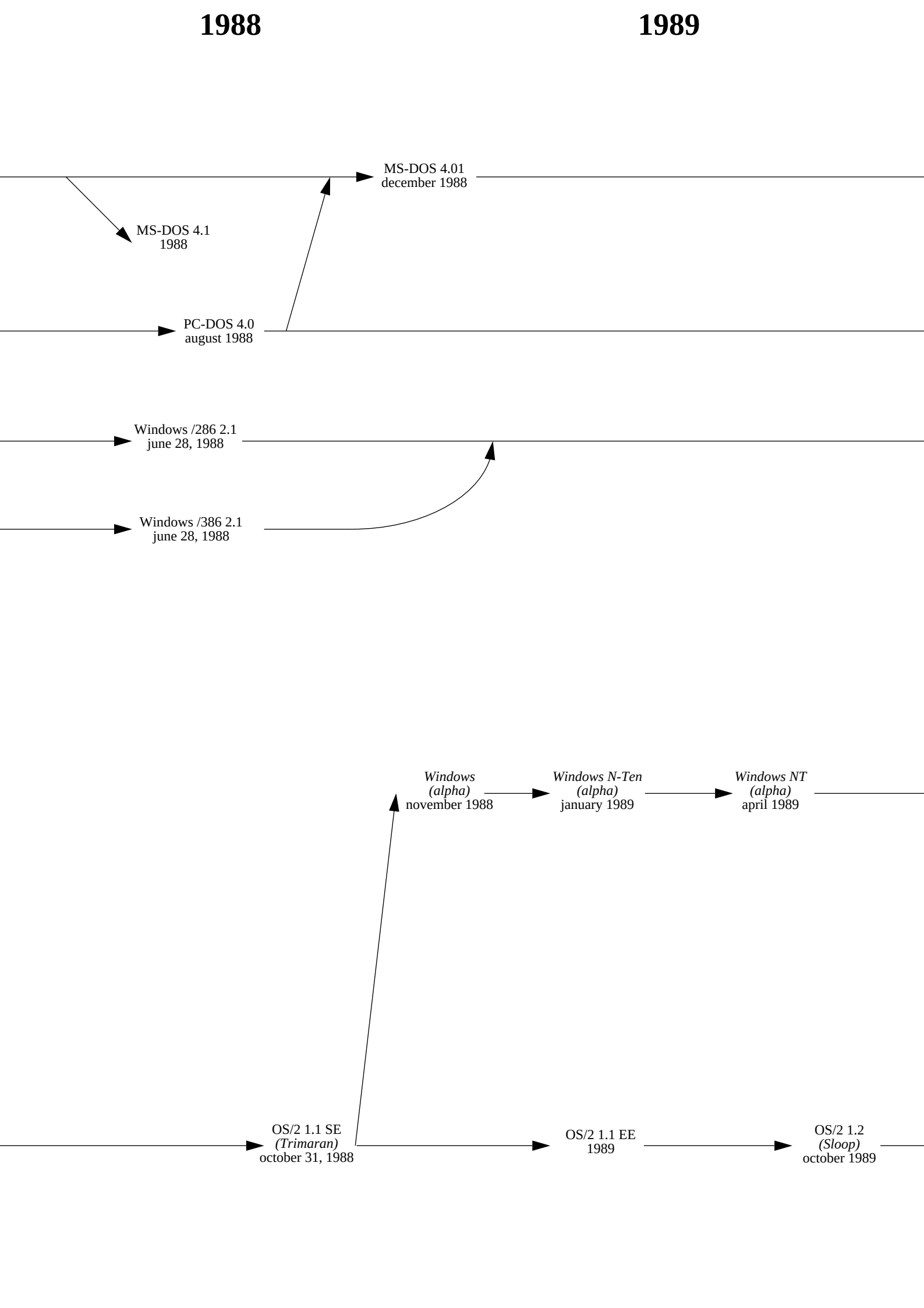
Closed Systems
october 22, 2009

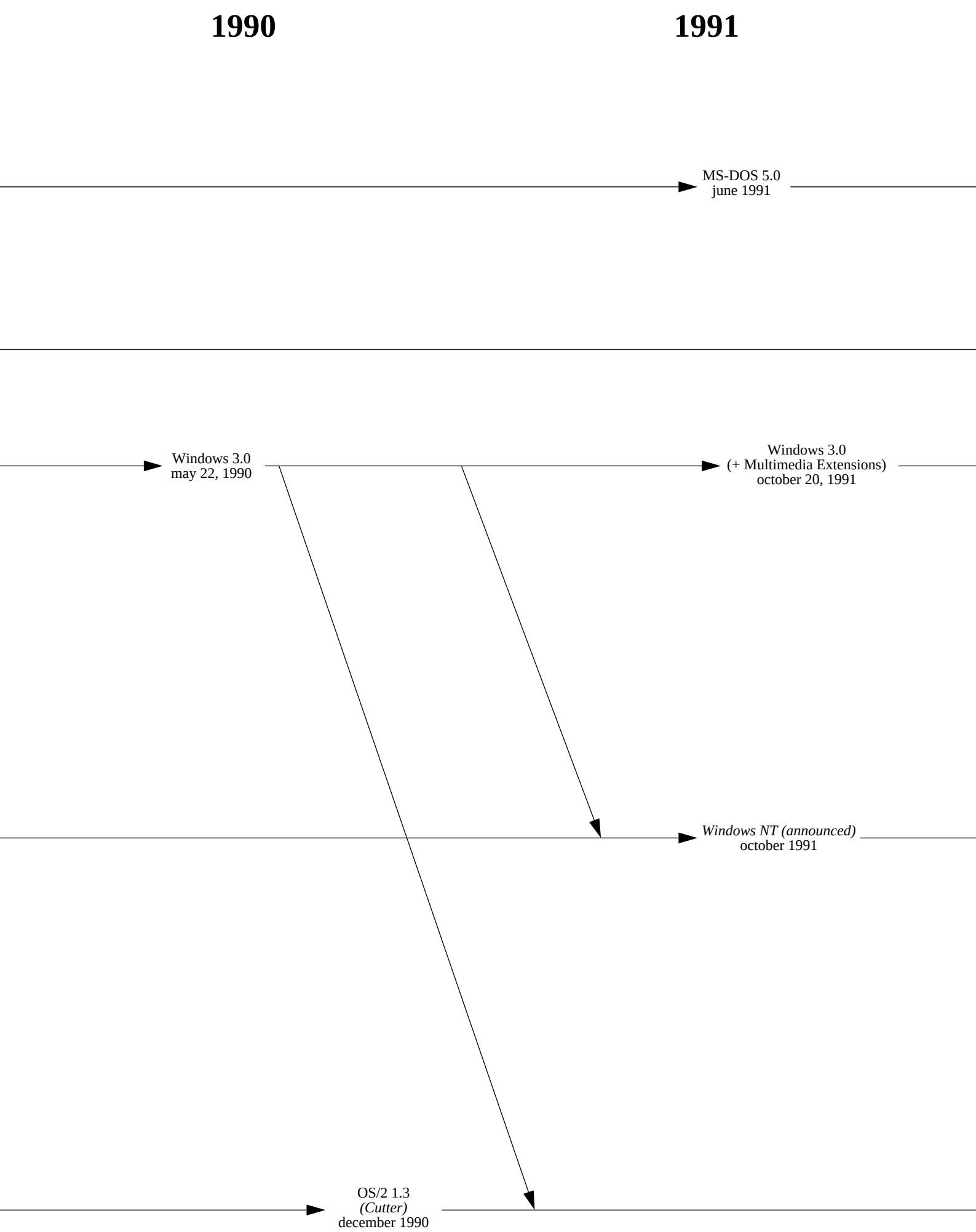
<http://www.levenez.com/windows/>

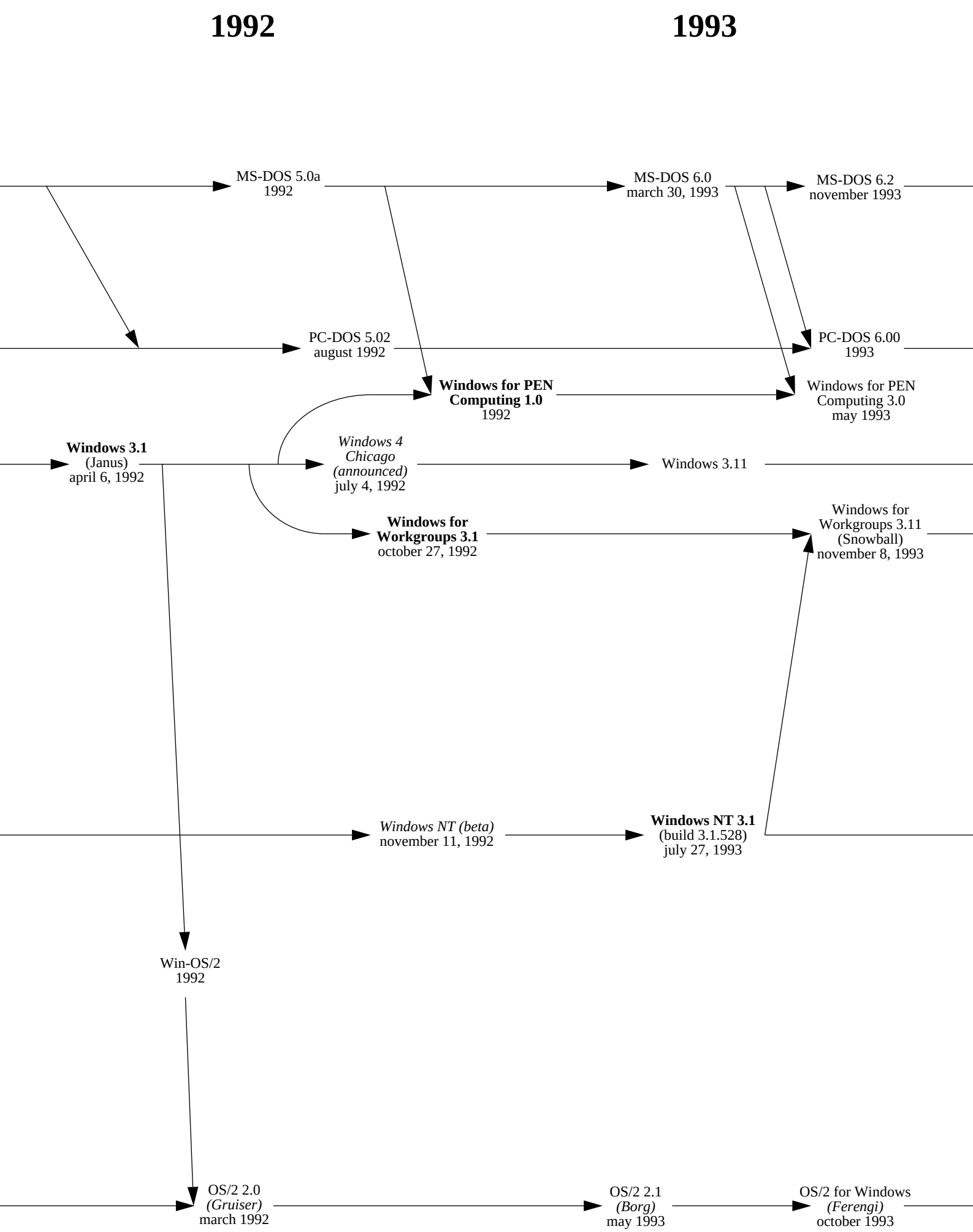

1982
1983
MS-DOS 1.24
june 1982
MS-DOS 1.25
july 1982
MS-DOS 2.01
may 1983
MS-DOS 2.11
december 1983
MS-DOS 2.12
1983
PC-DOS 1.10
june 1982
PC-DOS 2.0
march 1983
PC-DOS 2.10
october 1983
Windows
(announced)
november 10, 1983

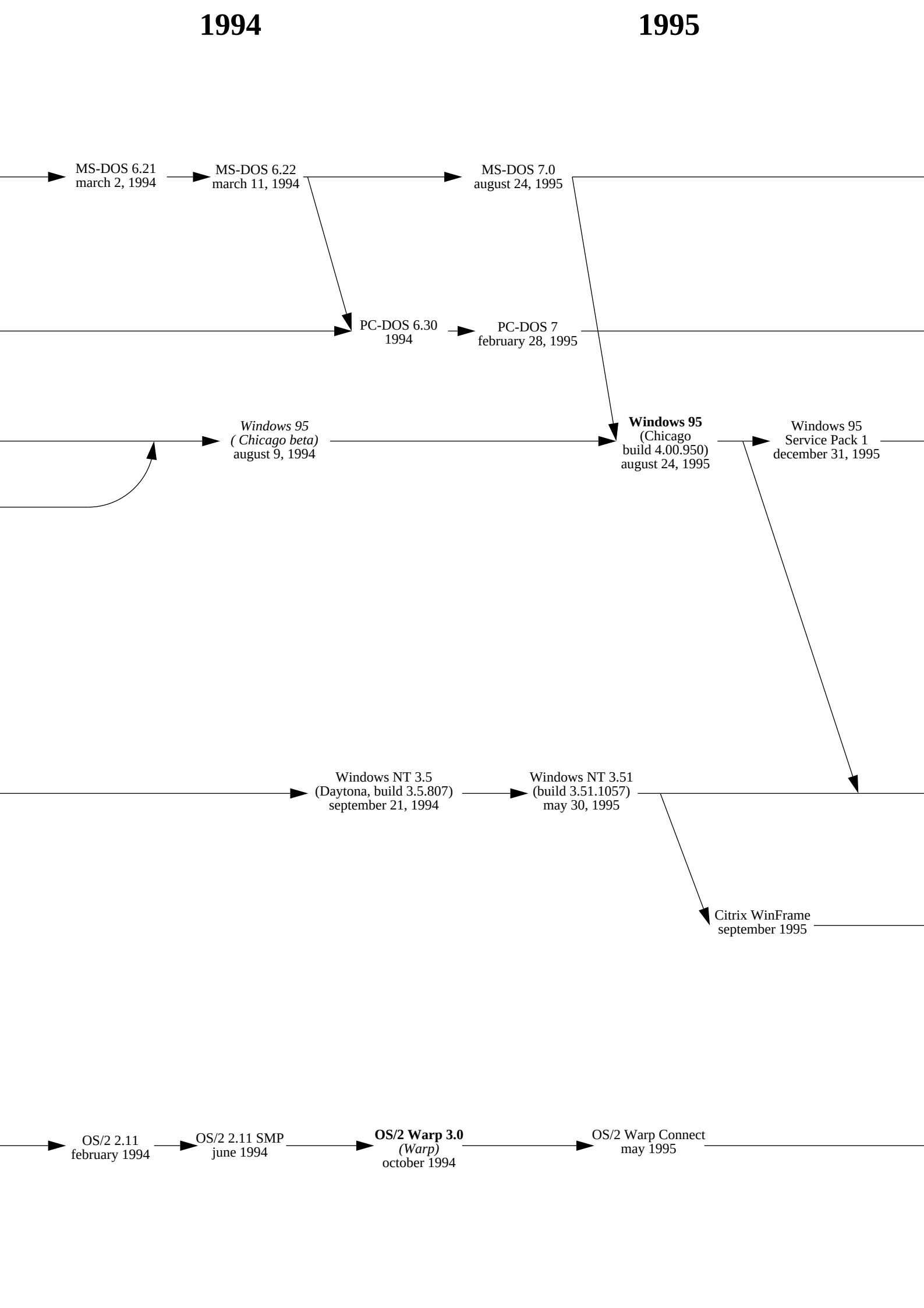

1984
1985
MS-DOS 3.05
november 1984
PC-DOS 3.0
august 1984
PC-DOS 3.10
november 1984
Windows 1.0
november 20, 1985
Windows 1.01
1985

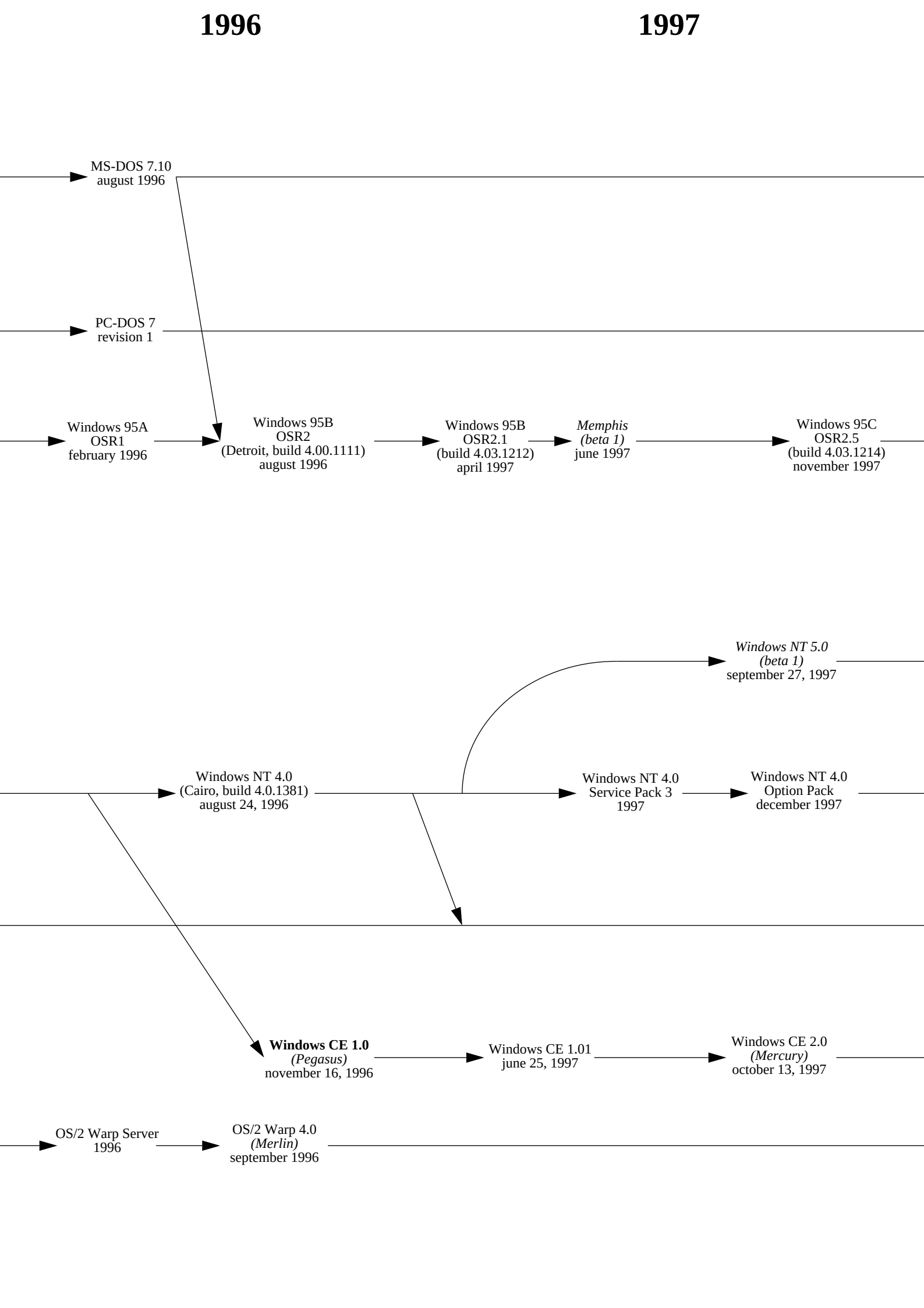

1986
1987
MS-DOS 3.3
(announced)
april 2, 1987
MS-DOS 3.3
august 1987
MS-DOS 3.31
november 1987
PC-DOS 3.20
january 1986
PC-DOS 3.30
april 1987
Windows 1.03
1986
Windows 2.0
(announced)
april 2, 1987
Windows 2.0
december 9, 1987
Windows /386
(announced)
april 2, 1987
Windows /386 2.0
december 9, 1987
MS OS/2 1.0
(CP/DOS)
april 2, 1987


1988
1989
MS-DOS 4.01
december 1988
MS-DOS 4.1
1988
PC-DOS 4.0
august 1988
Windows /286 2.1
june 28, 1988
Windows /386 2.1
june 28, 1988
Windows
(alpha)
november 1988
Windows N-Ten
(alpha)
january 1989
Windows NT
(alpha)
april 1989
OS/2 1.1 SE
(Trimaran)
october 31, 1988
OS/2 1.1 EE
1989
OS/2 1.2
(Sloop)
october 1989

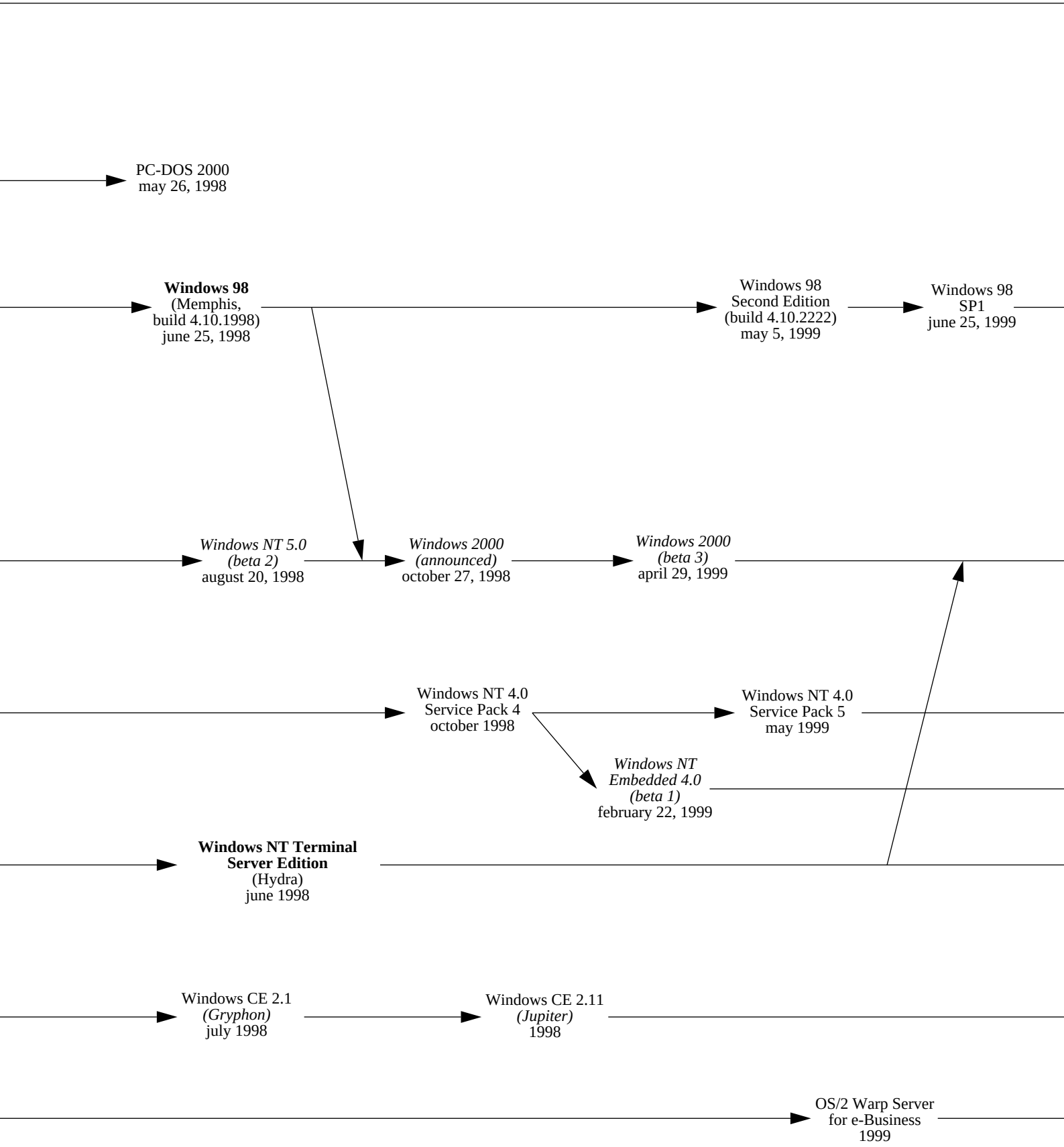

1990
1991
MS-DOS 5.0
june 1991
Windows 3.0
may 22, 1990
Windows 3.0
(+ Multimedia Extensions)
october 20, 1991
Windows NT (announced)
october 1991
OS/2 1.3
(Cutter)
december 1990


1992
1993
MS-DOS 5.0a
1992
MS-DOS 6.0
march 30, 1993
MS-DOS 6.2
november 1993
PC-DOS 5.02
august 1992
PC-DOS 6.00
1993
Windows for PEN
Computing 1.0
1992
Windows for PEN
Computing 3.0
may 1993
Windows 3.1
(Janus)
april 6, 1992
Windows 4
Chicago
(announced)
july 4, 1992
Windows 3.11
Windows for
Workgroups 3.1
october 27, 1992
Windows for
Workgroups 3.11
(Snowball)
november 8, 1993
Windows NT (beta)
november 11, 1992
Windows NT 3.1
(build 3.1.528)
july 27, 1993
Win-OS/2
1992
OS/2 2.0
(Gruiser)
march 1992
OS/2 2.1
(Borg)
may 1993
OS/2 for Windows
(Ferengi)
october 1993

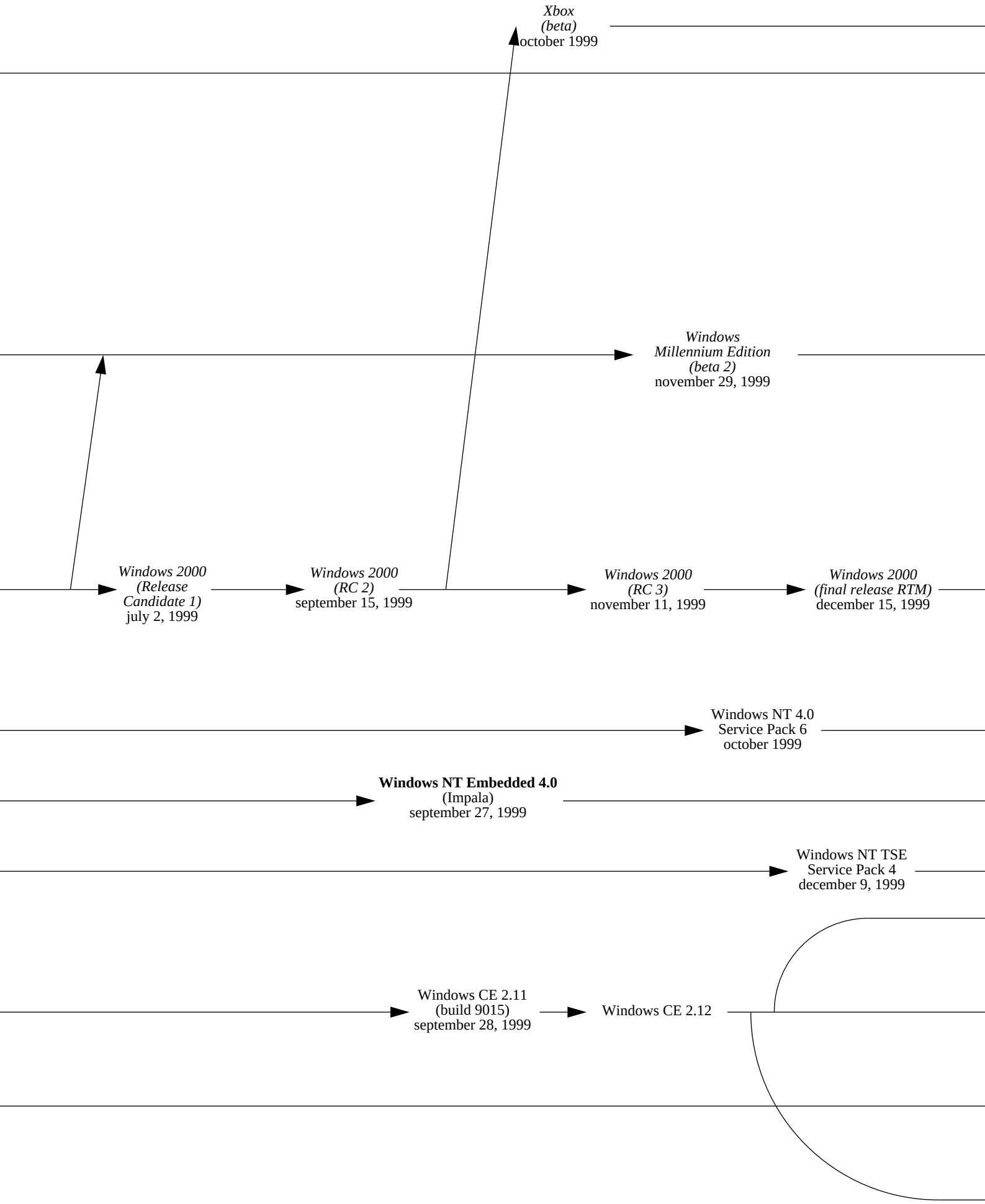

1994
1995
MS-DOS 6.21
march 2, 1994
MS-DOS 6.22
march 11, 1994
MS-DOS 7.0
august 24, 1995
PC-DOS 6.30
1994
PC-DOS 7
february 28, 1995
Windows 95
(Chicago beta)
august 9, 1994
Windows 95
(Chicago
build 4.00.950)
august 24, 1995
Windows 95
Service Pack 1
december 31, 1995
Windows NT 3.5
(Daytona, build 3.5.807)
september 21, 1994
Windows NT 3.51
(build 3.51.1057)
may 30, 1995
Citrix WinFrame
september 1995
OS/2 2.11
february 1994
OS/2 2.11 SMP
june 1994
OS/2 Warp 3.0
(Warp)
october 1994
OS/2 Warp Connect
may 1995


1996
1997
MS-DOS 7.10
august 1996
PC-DOS 7
revision 1
Windows 95A
OSR1
february 1996
Windows 95B
OSR2
(Detroit, build 4.00.1111)
august 1996
Windows 95B
OSR2.1
(build 4.03.1212)
april 1997
Memphis
(beta 1)
june 1997
Windows 95C
OSR2.5
(build 4.03.1214)
november 1997
Windows NT 5.0
(beta 1)
september 27, 1997
Windows NT 4.0
(Cairo, build 4.0.1381)
august 24, 1996
Windows NT 4.0
Service Pack 3
1997
Windows NT 4.0
Option Pack
december 1997
Windows CE 1.0
(Pegasus)
november 16, 1996
Windows CE 1.01
june 25, 1997
Windows CE 2.0
(Mercury)
october 13, 1997
OS/2 Warp Server
1996
OS/2 Warp 4.0
(Merlin)
september 1996

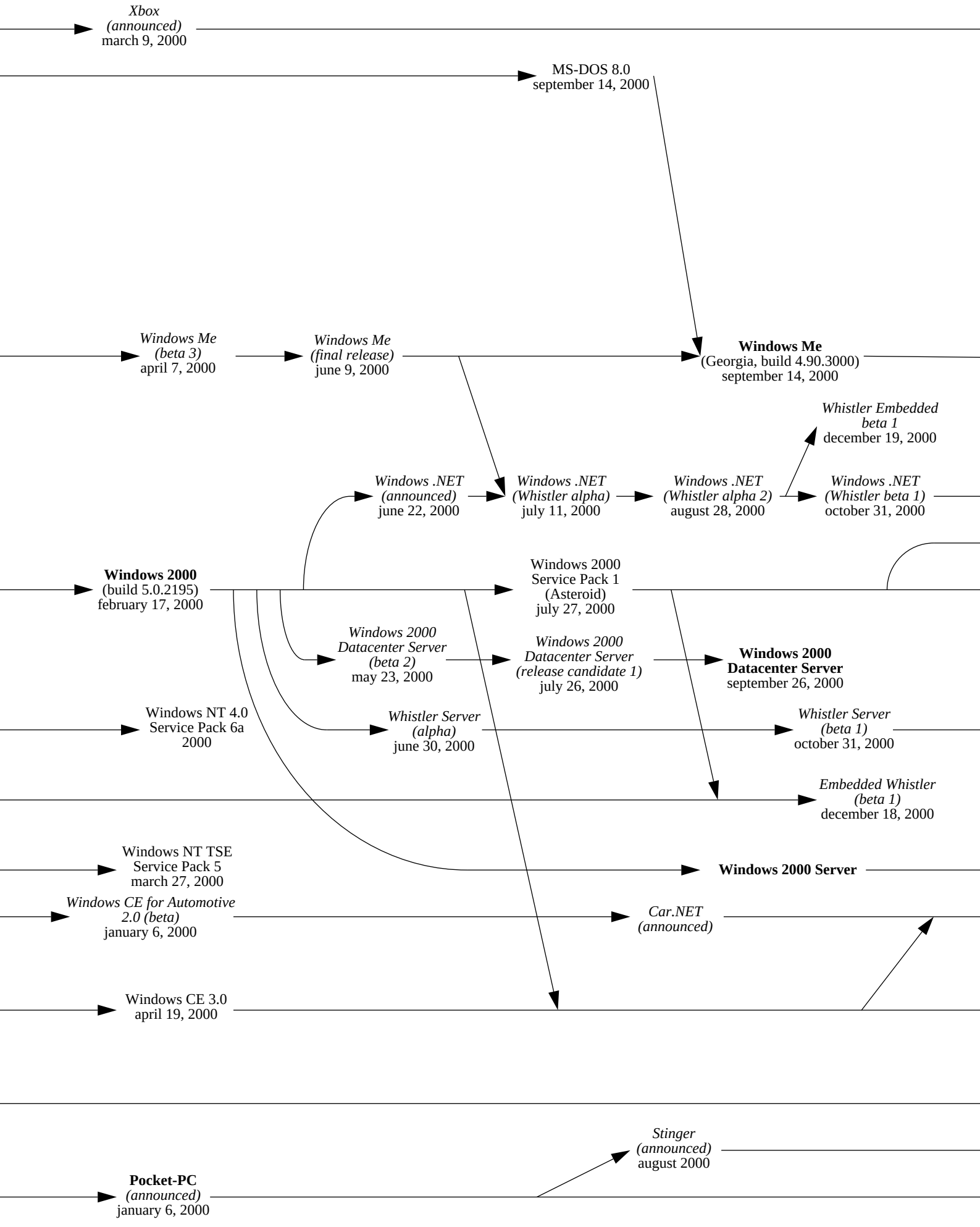
1998


PC-DOS 2000
may 26, 1998
Windows 98
(Memphis,
build 4.10.1998)
june 25, 1998
Windows 98
Second Edition
(build 4.10.2222)
may 5, 1999
Windows 98
SP1
june 25, 1999
Windows NT 5.0
(beta 2)
august 20, 1998
Windows 2000
(announced)
october 27, 1998
Windows 2000
(beta 3)
april 29, 1999
Windows NT 4.0
Service Pack 4
october 1998
Windows NT 4.0
Service Pack 5
may 1999
Windows NT
Embedded 4.0
(beta 1)
february 22, 1999
Windows NT Terminal
Server Edition
(Hydra)
june 1998
Windows CE 2.1
(Gryphon)
july 1998
Windows CE 2.11
(Jupiter)
1998
OS/2 Warp Server
for e-Business
1999

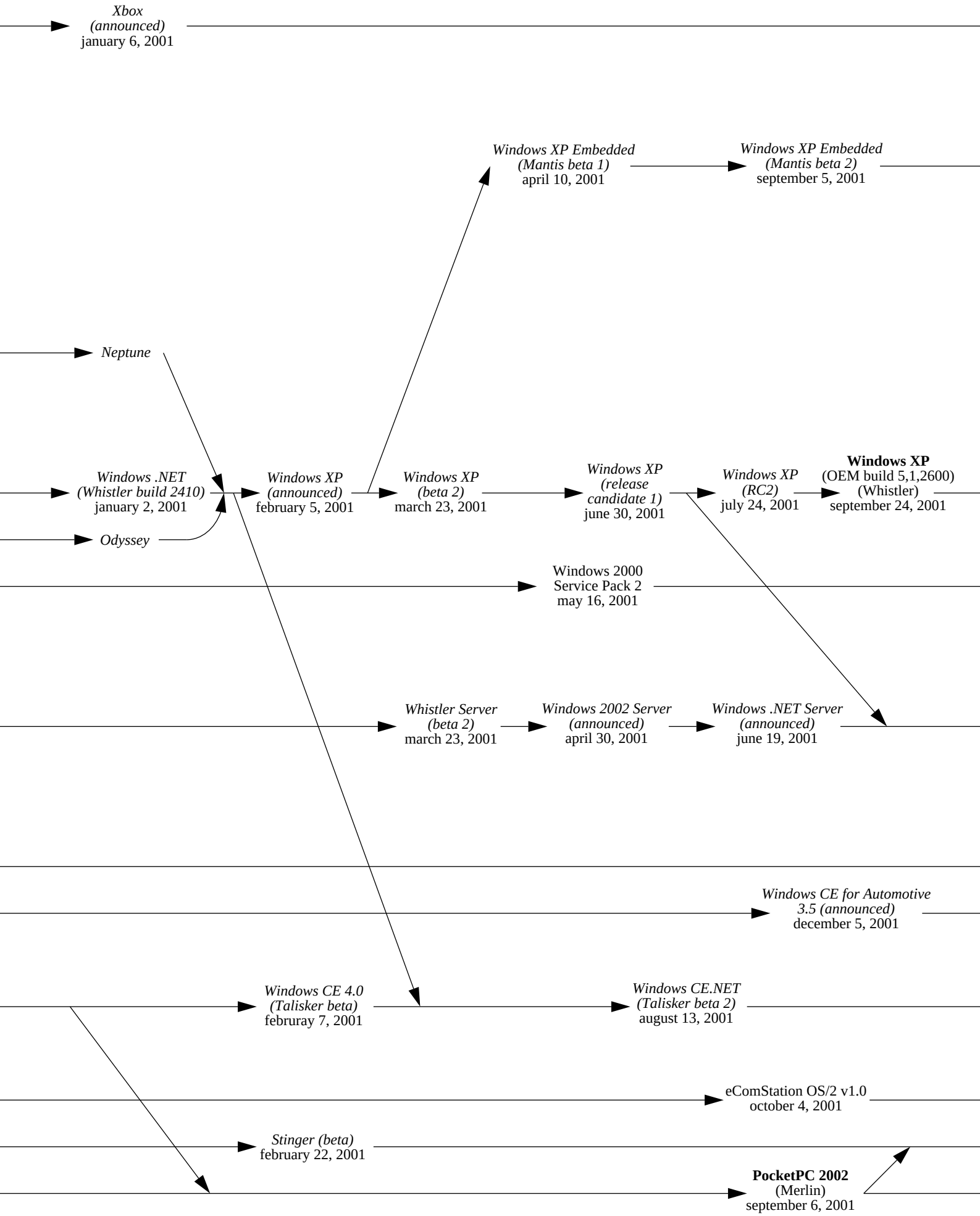
1999


Xbox
(beta)
october 1999
Windows
Millennium Edition
(beta 2)
november 29, 1999
Windows 2000
(Release
Candidate 1)
july 2, 1999
Windows 2000
(RC 2)
september 15, 1999
Windows 2000
(RC 3)
november 11, 1999
Windows 2000
(final release RTM)
december 15, 1999
Windows NT 4.0
Service Pack 6
october 1999
Windows NT Embedded 4.0
(Impala)
september 27, 1999
Windows NT TSE
Service Pack 4
december 9, 1999
Windows CE 2.11
(build 9015)
september 28, 1999
Windows CE 2.12

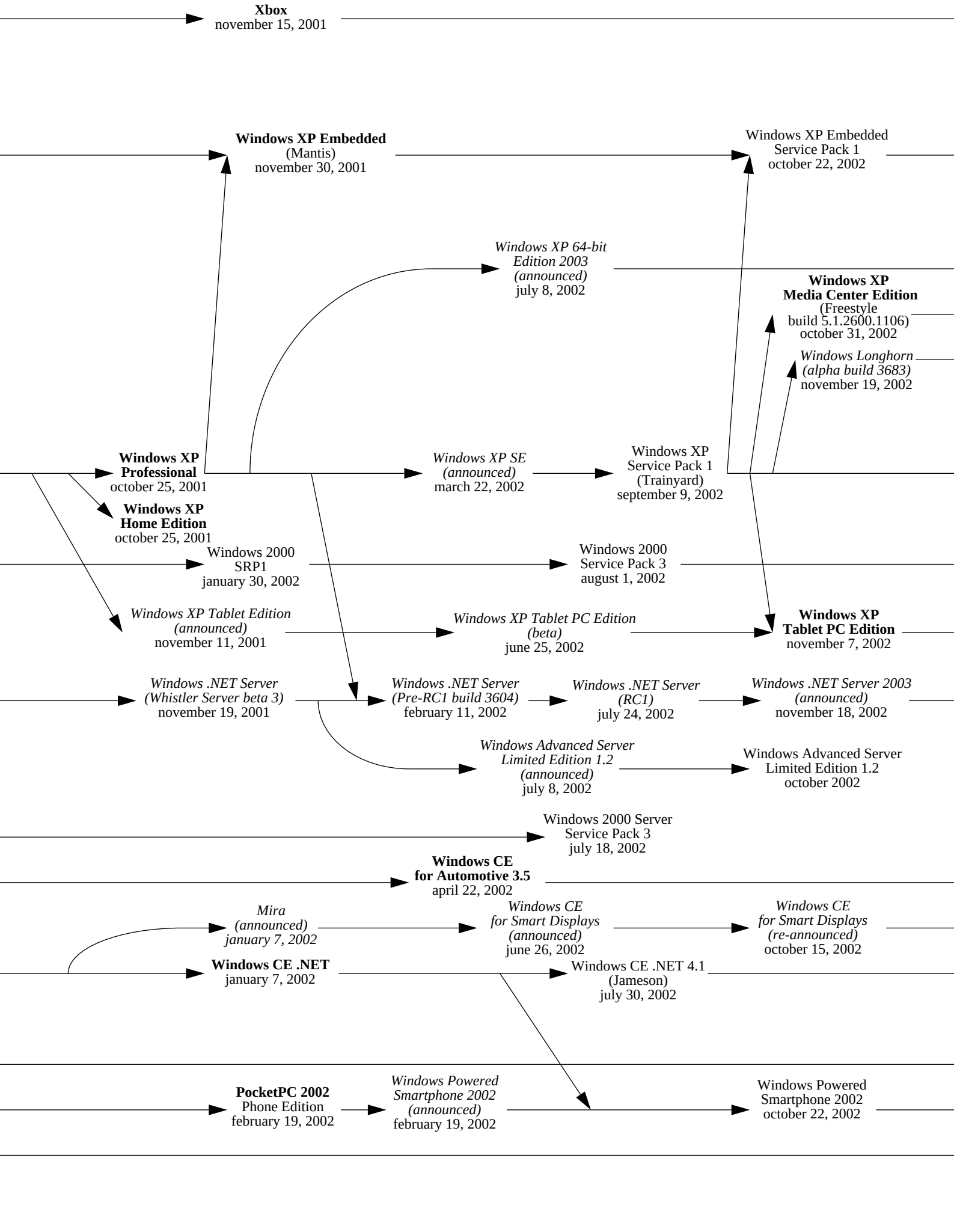
2000


Xbox
(announced)
march 9, 2000
MS-DOS 8.0
september 14, 2000
Windows Me
(beta 3)
april 7, 2000
Windows Me
(final release)
june 9, 2000
Windows Me
(Georgia, build 4.90.3000)
september 14, 2000
Whistler Embedded
beta 1
december 19, 2000
Windows .NET
(announced)
june 22, 2000
Windows .NET
(Whistler alpha)
july 11, 2000
Windows .NET
(Whistler alpha 2)
august 28, 2000
Windows .NET
(Whistler beta 1)
october 31, 2000
Windows 2000
(build 5.0.2195)
february 17, 2000
Windows 2000
Service Pack 1
(Asteroid)
july 27, 2000
Windows 2000
Datacenter Server
(beta 2)
may 23, 2000
Windows 2000
Datacenter Server
(release candidate 1)
july 26, 2000
Windows 2000
Datacenter Server
september 26, 2000
Windows NT 4.0
Service Pack 6a
2000
Whistler Server
(alpha)
june 30, 2000
Whistler Server
(beta 1)
october 31, 2000
Embedded Whistler
(beta 1)
december 18, 2000
Windows NT TSE
Service Pack 5
march 27, 2000
Windows 2000 Server
Windows CE for Automotive
2.0 (beta)
january 6, 2000
Car.NET
(announced)
Windows CE 3.0
april 19, 2000
Stinger
(announced)
august 2000
Pocket-PC
(announced)
january 6, 2000

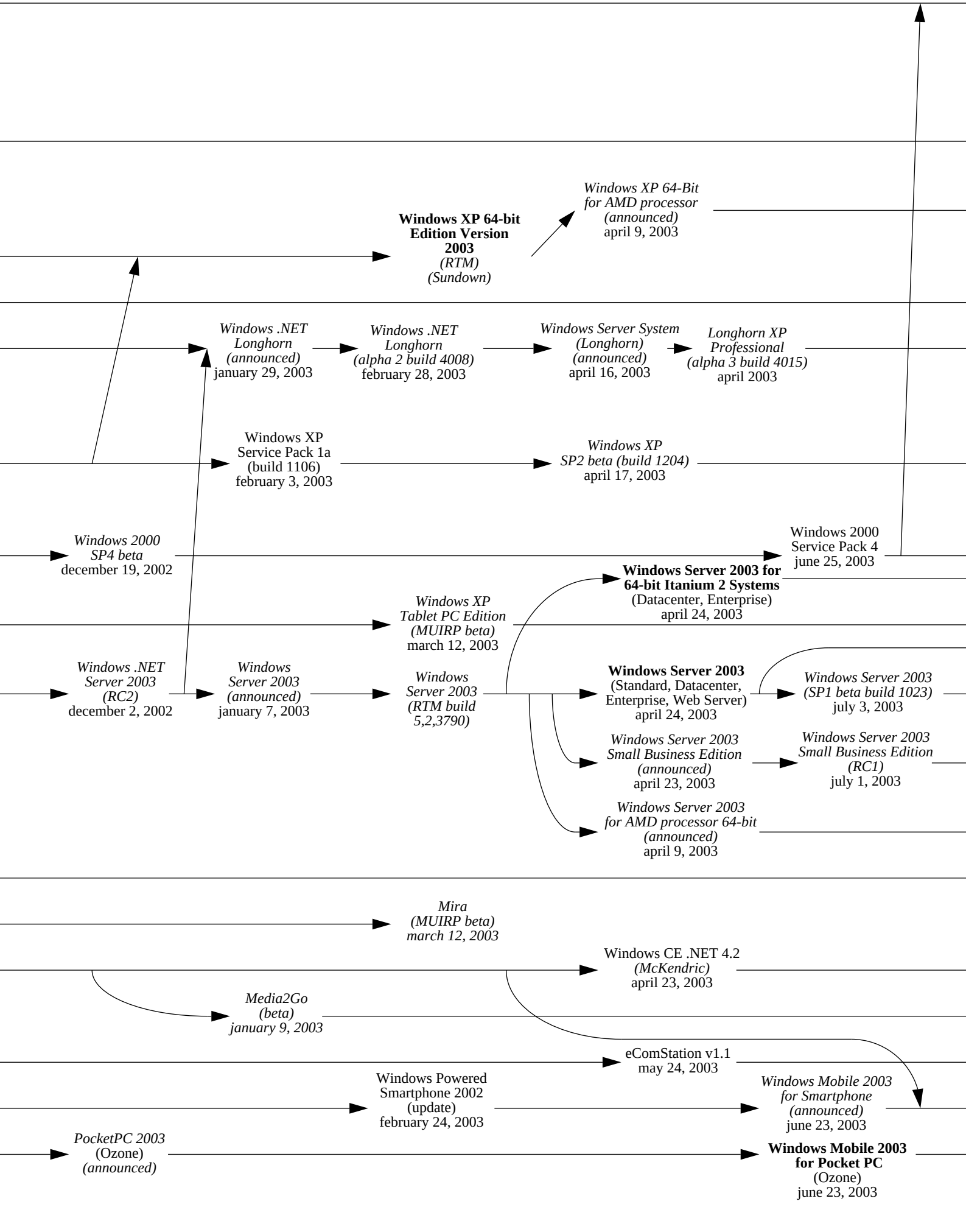
2001

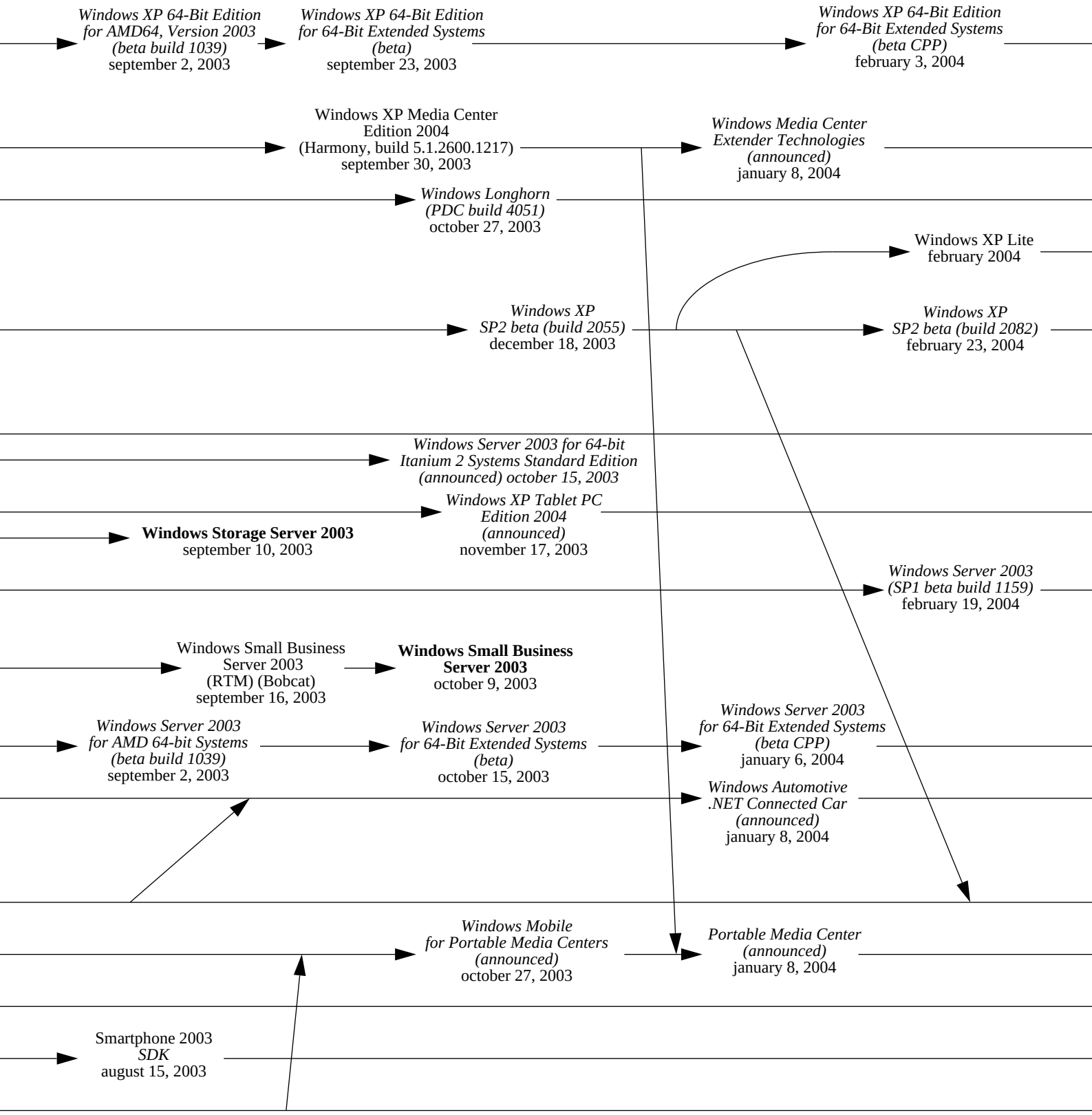

Xbox
(announced)
january 6, 2001
Windows XP Embedded
(Mantis beta 1)
april 10, 2001
Windows XP Embedded
(Mantis beta 2)
september 5, 2001
Neptune
Windows .NET
(Whistler build 2410)
january 2, 2001
Odyssey
Windows XP
(announced)
february 5, 2001
Windows XP
(beta 2)
march 23, 2001
Windows XP
(release
candidate 1)
june 30, 2001
Windows XP
(RC2)
july 24, 2001
Windows XP
(OEM build 5,1,2600)
(Whistler)
september 24, 2001
Windows 2000
Service Pack 2
may 16, 2001
Whistler Server
(beta 2)
march 23, 2001
Windows 2002 Server
(announced)
april 30, 2001
Windows .NET Server
(announced)
june 19, 2001
Windows CE for Automotive
3.5 (announced)
december 5, 2001
Windows CE 4.0
(Talisker beta)
februray 7, 2001
Windows CE.NET
(Talisker beta 2)
august 13, 2001
eComStation OS/2 v1.0
october 4, 2001
Stinger (beta)
february 22, 2001
PocketPC 2002
(Merlin)
september 6, 2001

2002

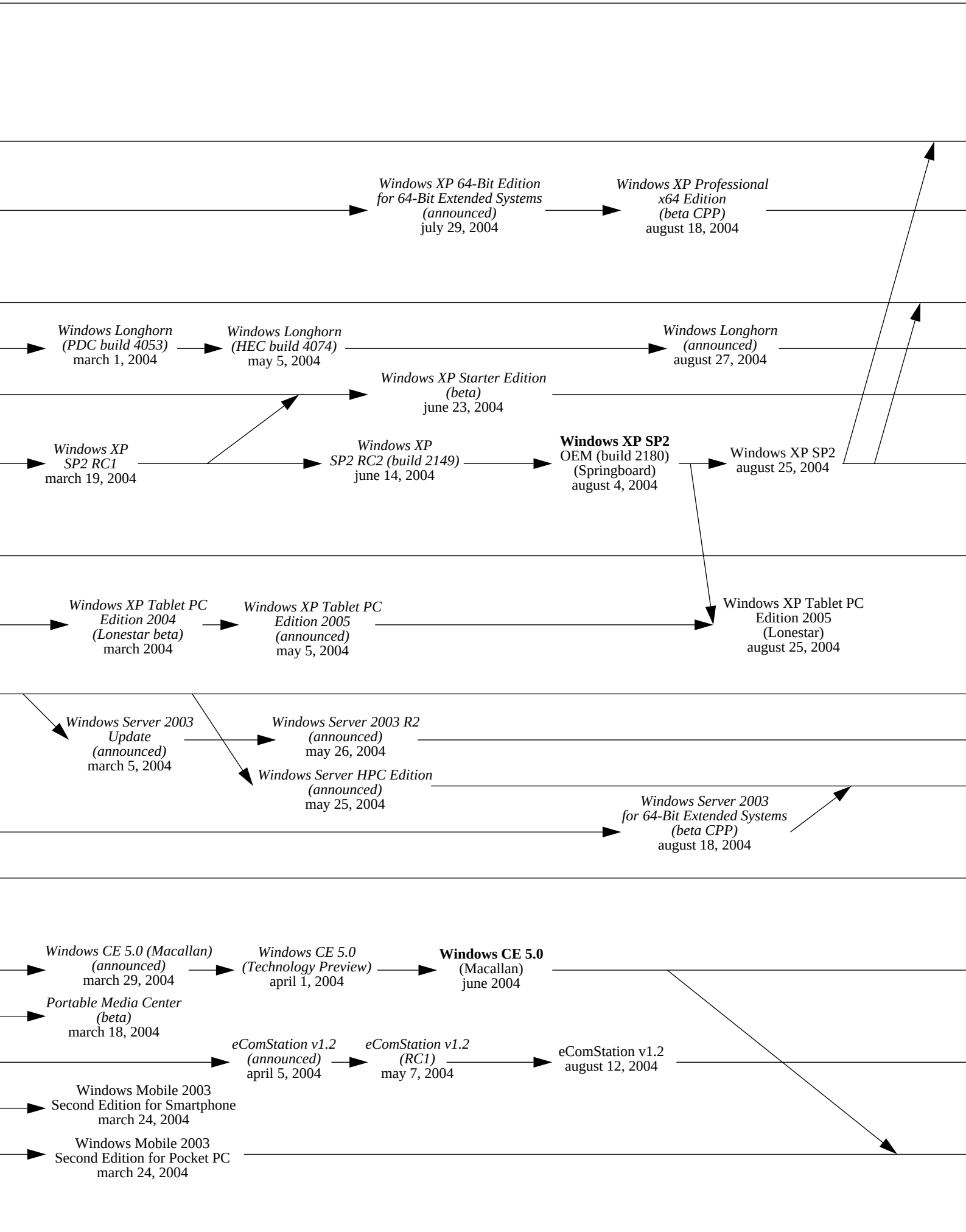

Xbox
november 15, 2001
Windows XP Embedded
(Mantis)
november 30, 2001
Windows XP Embedded
Service Pack 1
october 22, 2002
Windows XP 64-bit
Edition 2003
(announced)
july 8, 2002
Windows XP
Media Center Edition
(Freestyle
build 5.1.2600.1106)
october 31, 2002
Windows Longhorn
(alpha build 3683)
november 19, 2002
Windows XP
Professional
october 25, 2001
Windows XP SE
(announced)
march 22, 2002
Windows XP
Service Pack 1
(Trainyard)
september 9, 2002
Windows XP
Home Edition
october 25, 2001
Windows 2000
SRP1
january 30, 2002
Windows 2000
Service Pack 3
august 1, 2002
Windows XP Tablet Edition
(announced)
november 11, 2001
Windows XP Tablet PC Edition
(beta)
june 25, 2002
Windows XP
Tablet PC Edition
november 7, 2002
Windows .NET Server
(Whistler Server beta 3)
november 19, 2001
Windows .NET Server
(Pre-RC1 build 3604)
february 11, 2002
Windows .NET Server
(RC1)
july 24, 2002
Windows .NET Server 2003
(announced)
november 18, 2002
Windows Advanced Server
Limited Edition 1.2
(announced)
july 8, 2002
Windows Advanced Server
Limited Edition 1.2
october 2002
Windows 2000 Server
Service Pack 3
july 18, 2002
Windows CE
for Automotive 3.5
april 22, 2002
Mira
(announced)
january 7, 2002
Windows CE
for Smart Displays
(announced)
june 26, 2002
Windows CE
for Smart Displays
(re-announced)
october 15, 2002
Windows CE .NET
january 7, 2002
Windows CE .NET 4.1
(Jameson)
july 30, 2002
PocketPC 2002
Phone Edition
february 19, 2002
Windows Powered
Smartphone 2002
(announced)
february 19, 2002
Windows Powered
Smartphone 2002
october 22, 2002

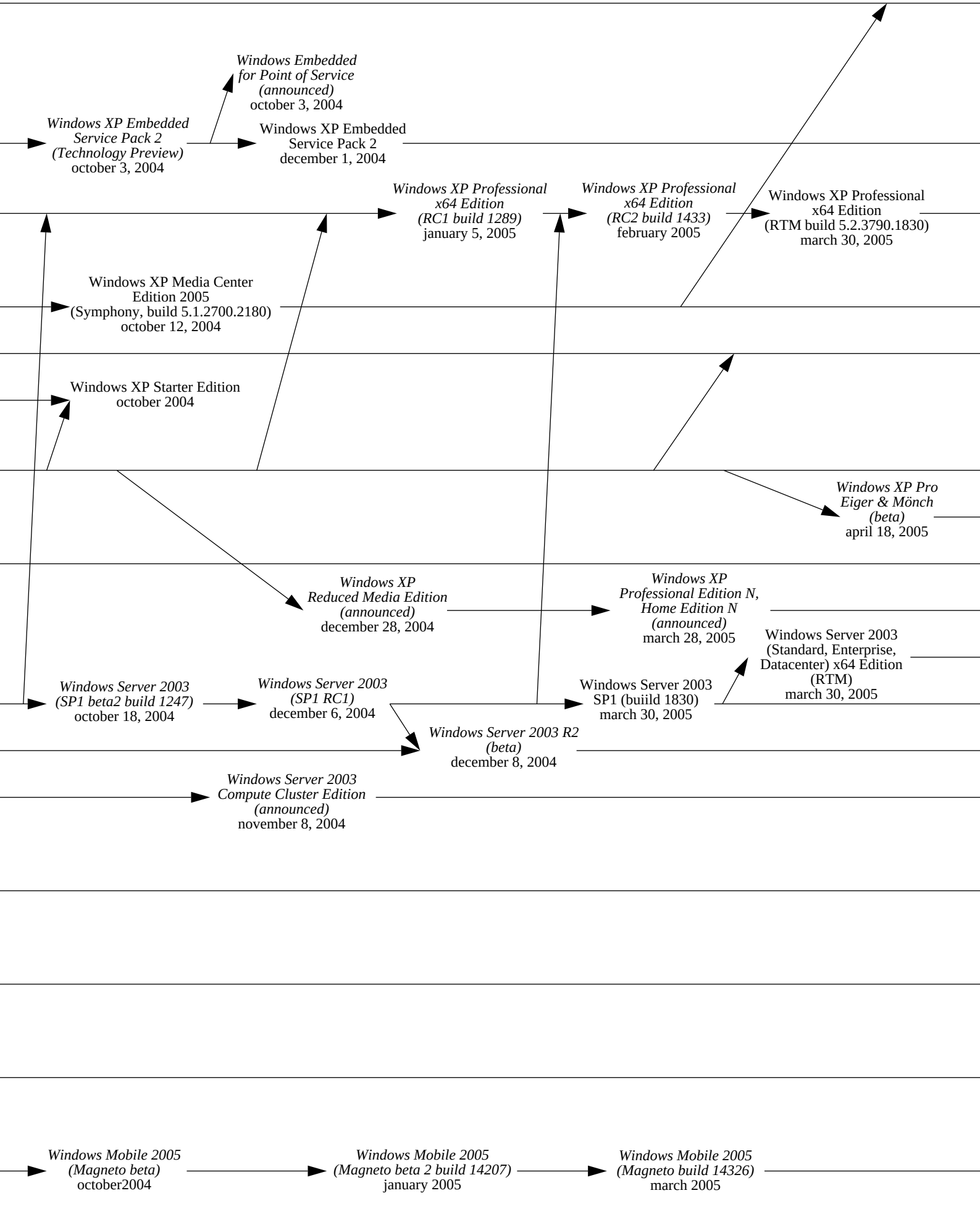
2003


Windows XP 64-bit Edition Version 2003
(RTM)
(Sundown)
Windows XP 64-Bit for AMD processor (announced)
april 9, 2003
Windows .NET Longhorn (announced)
january 29, 2003
Windows .NET Longhorn (alpha 2 build 4008)
february 28, 2003
Windows Server System (Longhorn) (announced)
april 16, 2003
Longhorn XP Professional (alpha 3 build 4015)
april 2003
Windows XP Service Pack 1a (build 1106)
february 3, 2003
Windows XP SP2 beta (build 1204)
april 17, 2003
Windows 2000 SP4 beta
december 19, 2002
Windows 2000 Service Pack 4
june 25, 2003
Windows Server 2003 for 64-bit Itanium 2 Systems (Datacenter, Enterprise)
april 24, 2003
Windows XP Tablet PC Edition (MUIRP beta)
march 12, 2003
Windows .NET Server 2003 (RC2)
december 2, 2002
Windows Server 2003 (announced)
january 7, 2003
Windows Server 2003 (RTM build 5,2,3790)
Windows Server 2003 (Standard, Datacenter, Enterprise, Web Server)
april 24, 2003
Windows Server 2003 (SP1 beta build 1023)
july 3, 2003
Windows Server 2003 Small Business Edition (announced)
april 23, 2003
Windows Server 2003 Small Business Edition (RC1)
july 1, 2003
Windows Server 2003 for AMD processor 64-bit (announced)
april 9, 2003
Mira (MUIRP beta)
march 12, 2003
Windows CE .NET 4.2 *(McKendric)*
april 23, 2003
Media2Go (beta)
january 9, 2003
eComStation v1.1
may 24, 2003
Windows Powered Smartphone 2002 (update)
february 24, 2003
Windows Mobile 2003 for Smartphone (announced)
june 23, 2003
PocketPC 2003 (Ozone) *(announced)*
Windows Mobile 2003 for Pocket PC (Ozone)
june 23, 2003

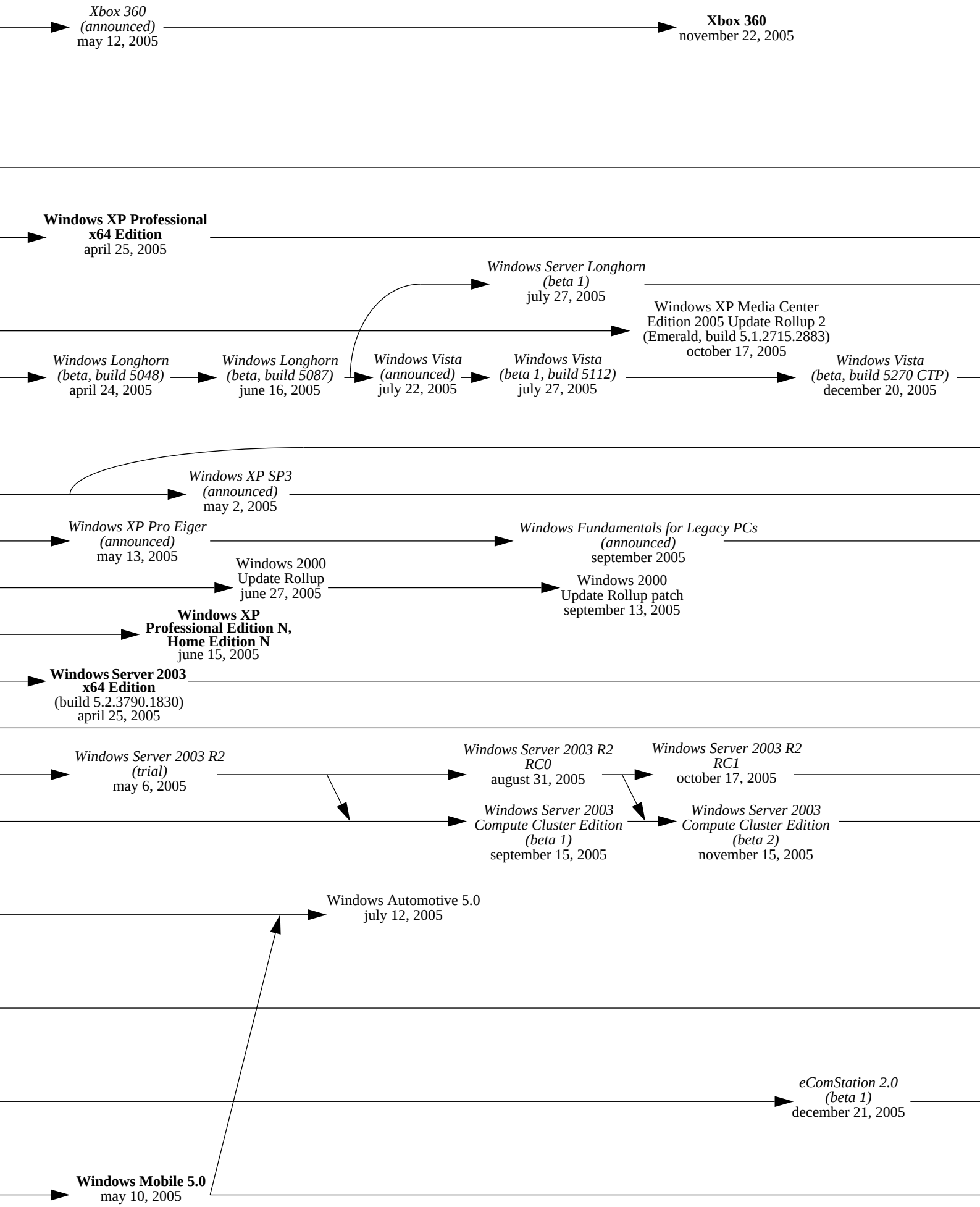

Windows XP 64-Bit Edition for AMD64, Version 2003 (beta build 1039) september 2, 2003
Windows XP 64-Bit Edition for 64-Bit Extended Systems (beta) september 23, 2003
Windows XP 64-Bit Edition for 64-Bit Extended Systems (beta CPP) february 3, 2004
Windows XP Media Center Edition 2004 (Harmony, build 5.1.2600.1217) september 30, 2003
Windows Media Center Extender Technologies (announced) january 8, 2004
Windows Longhorn (PDC build 4051) october 27, 2003
Windows XP Lite february 2004
Windows XP SP2 beta (build 2055) december 18, 2003
Windows XP SP2 beta (build 2082) february 23, 2004
Windows Server 2003 for 64-bit Itanium 2 Systems Standard Edition (announced) october 15, 2003
Windows XP Tablet PC Edition 2004 (announced) november 17, 2003
Windows Storage Server 2003 september 10, 2003
Windows Server 2003 (SP1 beta build 1159) february 19, 2004
Windows Small Business Server 2003 (RTM) (Bobcat) september 16, 2003
Windows Small Business Server 2003 october 9, 2003
Windows Server 2003 for AMD 64-bit Systems (beta build 1039) september 2, 2003
Windows Server 2003 for 64-Bit Extended Systems (beta) october 15, 2003
Windows Server 2003 for 64-Bit Extended Systems (beta CPP) january 6, 2004
Windows Automotive .NET Connected Car (announced) january 8, 2004
Windows Mobile for Portable Media Centers (announced) october 27, 2003
Portable Media Center (announced) january 8, 2004
Smartphone 2003 *SDK* august 15, 2003

2004

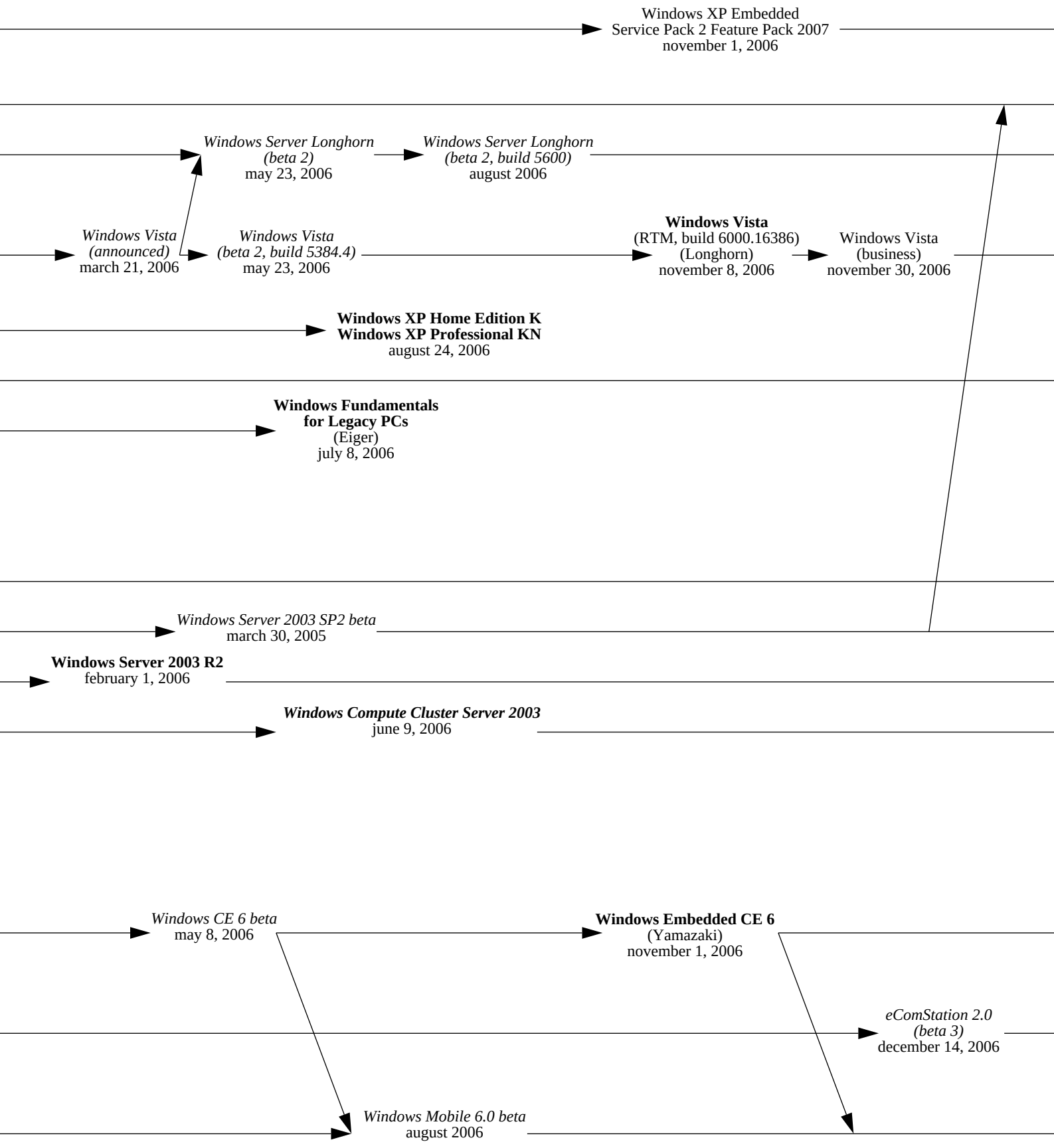

Windows XP 64-Bit Edition for 64-Bit Extended Systems (announced) july 29, 2004
Windows XP Professional x64 Edition (beta CPP) august 18, 2004
Windows Longhorn (PDC build 4053) march 1, 2004
Windows Longhorn (HEC build 4074) may 5, 2004
Windows Longhorn (announced) august 27, 2004
Windows XP Starter Edition (beta) june 23, 2004
Windows XP SP2 RC1 march 19, 2004
Windows XP SP2 RC2 (build 2149) june 14, 2004
Windows XP SP2 OEM (build 2180) (Springboard) august 4, 2004
Windows XP SP2 august 25, 2004
Windows XP Tablet PC Edition 2004 (Lonestar beta) march 2004
Windows XP Tablet PC Edition 2005 (announced) may 5, 2004
Windows XP Tablet PC Edition 2005 (Lonestar) august 25, 2004
Windows Server 2003 Update (announced) march 5, 2004
Windows Server 2003 R2 (announced) may 26, 2004
Windows Server HPC Edition (announced) may 25, 2004
Windows Server 2003 for 64-Bit Extended Systems (beta CPP) august 18, 2004
Windows CE 5.0 (Macallan) (announced) march 29, 2004
Windows CE 5.0 (Technology Preview) april 1, 2004
Windows CE 5.0 (Macallan) june 2004
Portable Media Center (beta) march 18, 2004
eComStation v1.2 (announced) april 5, 2004
eComStation v1.2 (RC1) may 7, 2004
eComStation v1.2 august 12, 2004
Windows Mobile 2003 Second Edition for Smartphone march 24, 2004
Windows Mobile 2003 Second Edition for Pocket PC march 24, 2004


Windows Embedded for Point of Service (announced) october 3, 2004
Windows XP Embedded Service Pack 2 (Technology Preview) october 3, 2004
Windows XP Embedded Service Pack 2 december 1, 2004
Windows XP Professional x64 Edition (RC1 build 1289) january 5, 2005
Windows XP Professional x64 Edition (RC2 build 1433) february 2005
Windows XP Professional x64 Edition (RTM build 5.2.3790.1830) march 30, 2005
Windows XP Media Center Edition 2005 (Symphony, build 5.1.2700.2180) october 12, 2004
Windows XP Starter Edition october 2004
Windows XP Pro Eiger & Mönch (beta) april 18, 2005
Windows XP Reduced Media Edition (announced) december 28, 2004
Windows XP Professional Edition N, Home Edition N (announced) march 28, 2005
Windows Server 2003 (Standard, Enterprise, Datacenter) x64 Edition (RTM) march 30, 2005
Windows Server 2003 (SP1 beta2 build 1247) october 18, 2004
Windows Server 2003 (SP1 RC1) december 6, 2004
Windows Server 2003 SP1 (buiild 1830) march 30, 2005
Windows Server 2003 R2 (beta) december 8, 2004
Windows Server 2003 Compute Cluster Edition (announced) november 8, 2004
Windows Mobile 2005 (Magneto beta) october2004
Windows Mobile 2005 (Magneto beta 2 build 14207) january 2005
Windows Mobile 2005 (Magneto build 14326) march 2005

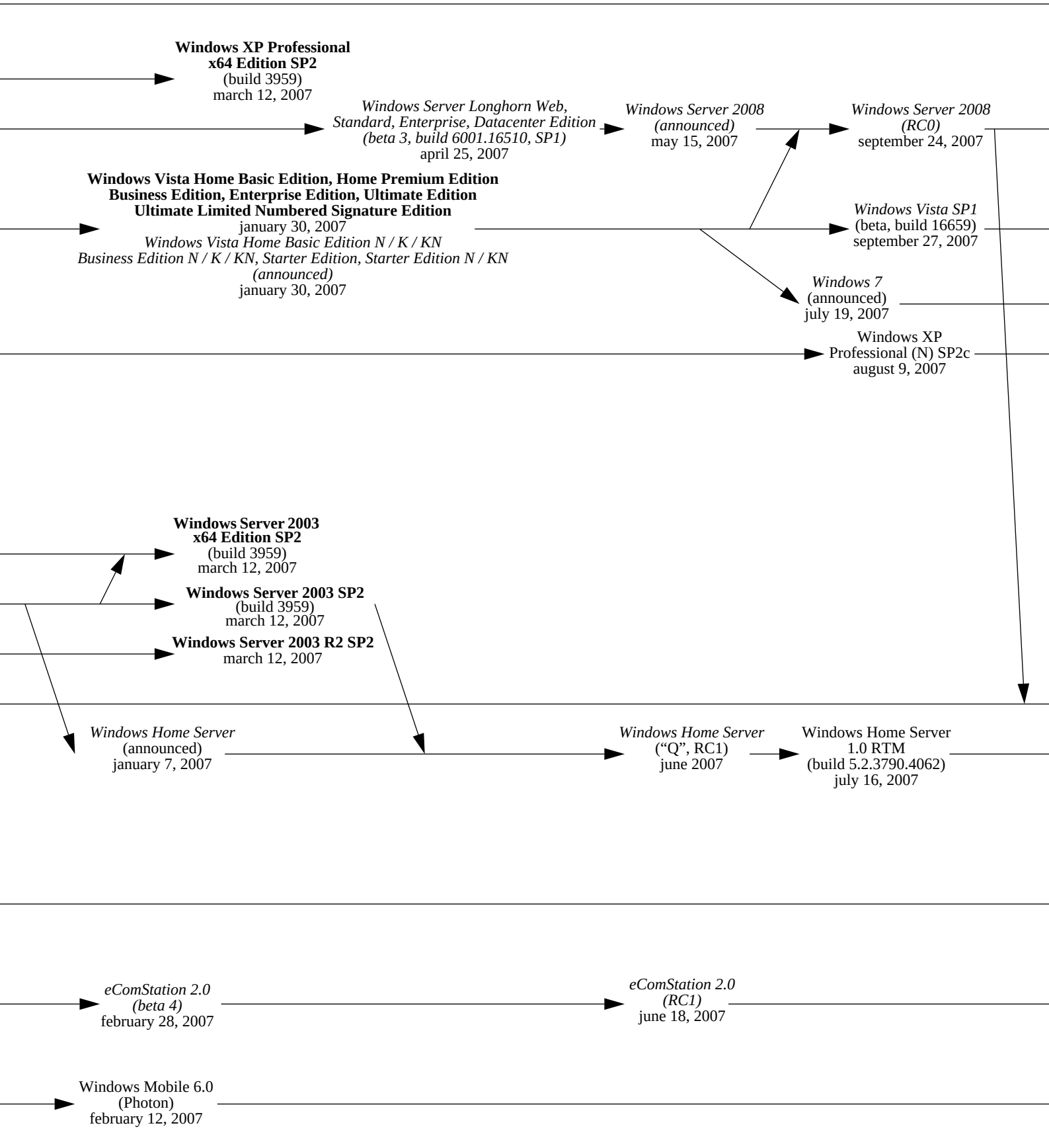
2005


Xbox 360 (announced)
may 12, 2005
Xbox 360
november 22, 2005
Windows XP Professional x64 Edition
april 25, 2005
Windows Server Longhorn (beta 1)
july 27, 2005
Windows XP Media Center Edition 2005 Update Rollup 2 (Emerald, build 5.1.2715.2883)
october 17, 2005
Windows Longhorn (beta, build 5048)
april 24, 2005
Windows Longhorn (beta, build 5087)
june 16, 2005
Windows Vista (announced)
july 22, 2005
Windows Vista (beta 1, build 5112)
july 27, 2005
Windows Vista (beta, build 5270 CTP)
december 20, 2005
Windows XP SP3 (announced)
may 2, 2005
Windows XP Pro Eiger (announced)
may 13, 2005
Windows Fundamentals for Legacy PCs (announced)
september 2005
Windows 2000 Update Rollup
june 27, 2005
Windows 2000 Update Rollup patch
september 13, 2005
Windows XP Professional Edition N, Home Edition N
june 15, 2005
Windows Server 2003 x64 Edition
(build 5.2.3790.1830)
april 25, 2005
Windows Server 2003 R2 (trial)
may 6, 2005
Windows Server 2003 R2 RC0
august 31, 2005
Windows Server 2003 R2 RC1
october 17, 2005
Windows Server 2003 Compute Cluster Edition (beta 1)
september 15, 2005
Windows Server 2003 Compute Cluster Edition (beta 2)
november 15, 2005
Windows Automotive 5.0
july 12, 2005
eComStation 2.0 (beta 1)
december 21, 2005
Windows Mobile 5.0
may 10, 2005

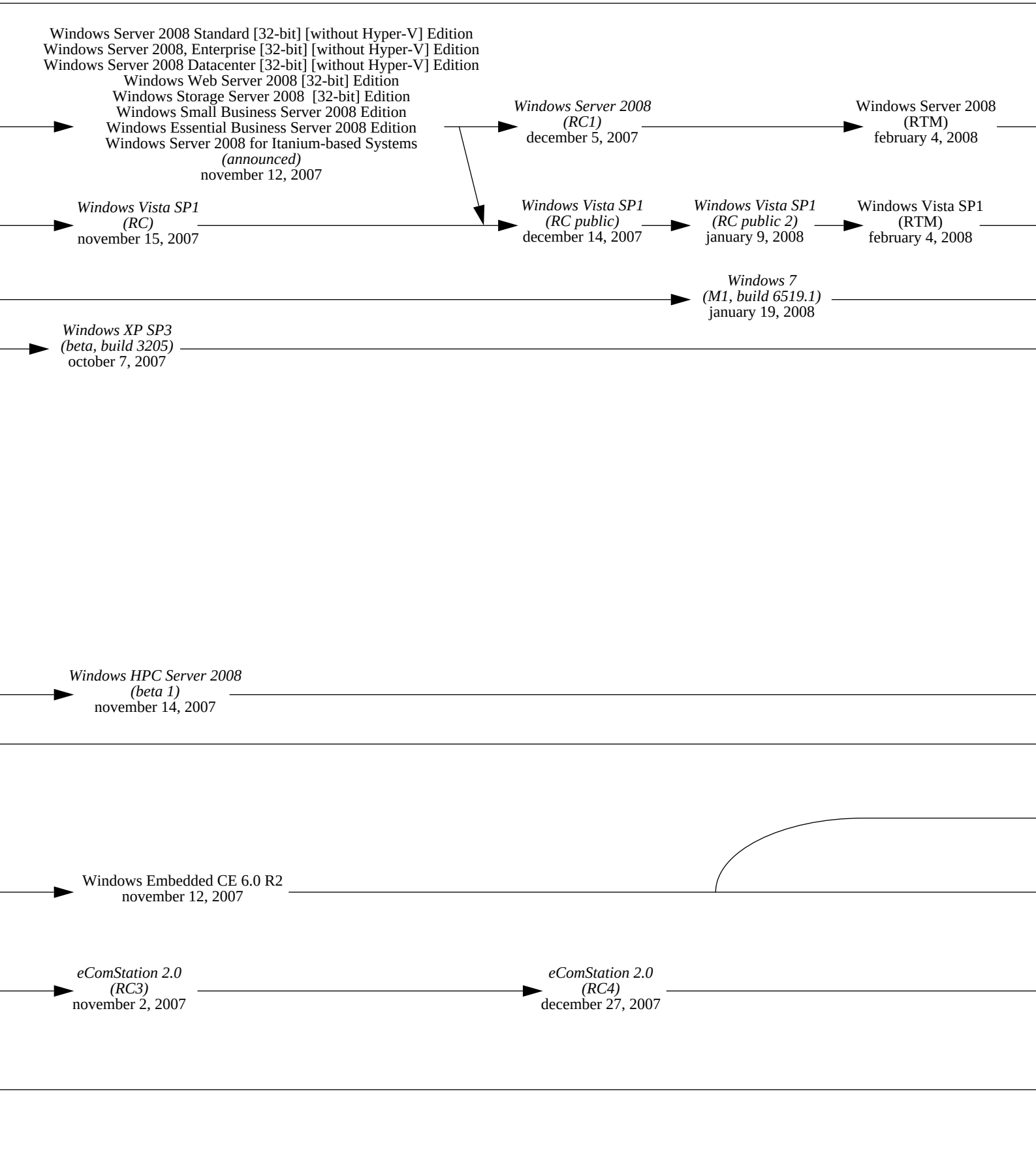
2006

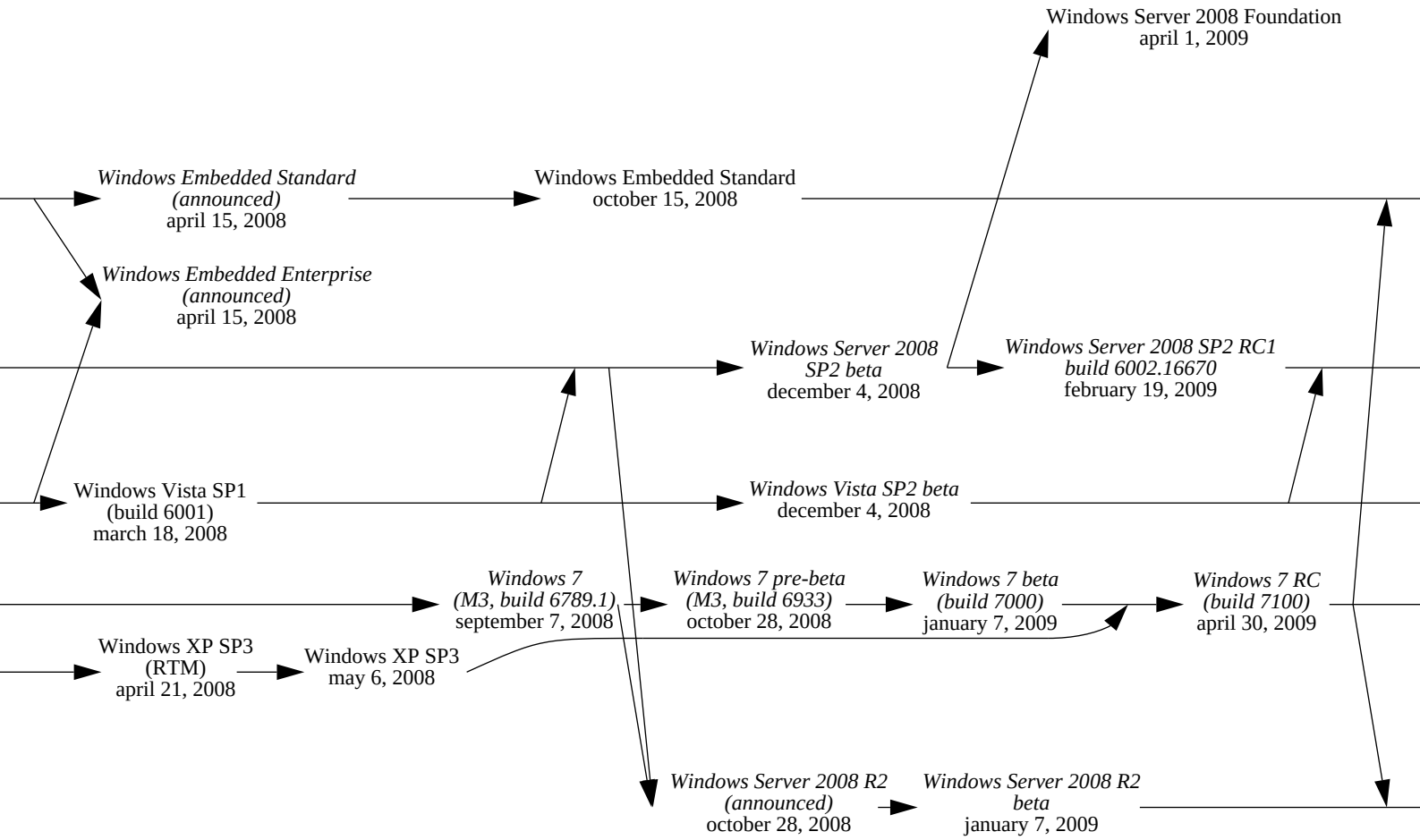

Windows XP Embedded
Service Pack 2 Feature Pack 2007
november 1, 2006
Windows Server Longhorn
(beta 2)
may 23, 2006
Windows Server Longhorn
(beta 2, build 5600)
august 2006
Windows Vista
(announced)
march 21, 2006
Windows Vista
(beta 2, build 5384.4)
may 23, 2006
Windows Vista
(RTM, build 6000.16386)
(Longhorn)
november 8, 2006
Windows Vista
(business)
november 30, 2006
Windows XP Home Edition K
Windows XP Professional KN
august 24, 2006
Windows Fundamentals
for Legacy PCs
(Eiger)
july 8, 2006
Windows Server 2003 SP2 beta
march 30, 2005
Windows Server 2003 R2
february 1, 2006
Windows Compute Cluster Server 2003
june 9, 2006
Windows CE 6 beta
may 8, 2006
Windows Embedded CE 6
(Yamazaki)
november 1, 2006
eComStation 2.0
(beta 3)
december 14, 2006
Windows Mobile 6.0 beta
august 2006

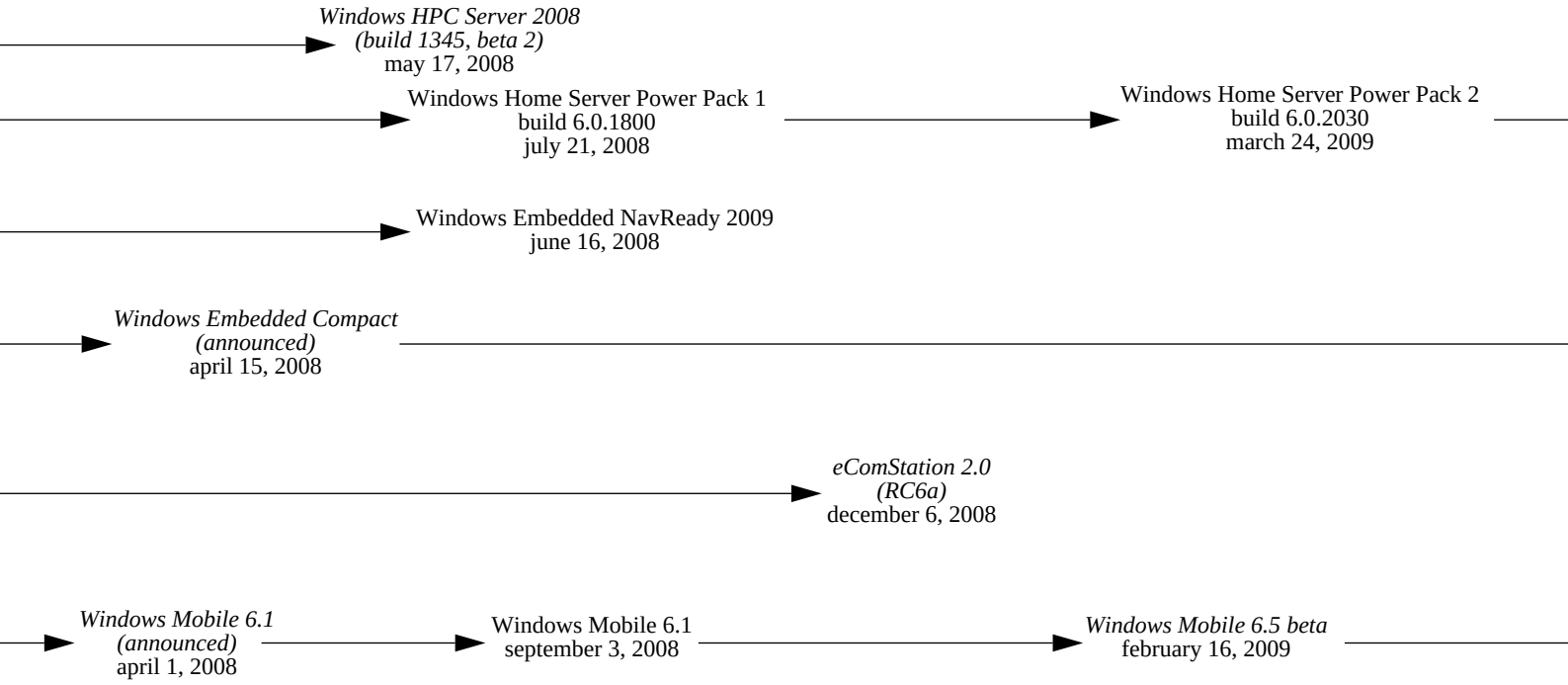
2007


Windows XP Professional x64 Edition SP2
(build 3959)
march 12, 2007
Windows Server Longhorn Web, Standard, Enterprise, Datacenter Edition (beta 3, build 6001.16510, SP1)
april 25, 2007
Windows Server 2008 (announced)
may 15, 2007
Windows Server 2008 (RC0)
september 24, 2007
Windows Vista Home Basic Edition, Home Premium Edition Business Edition, Enterprise Edition, Ultimate Edition Ultimate Limited Numbered Signature Edition
january 30, 2007
Windows Vista Home Basic Edition N / K / KN Business Edition N / K / KN, Starter Edition, Starter Edition N / KN (announced)
january 30, 2007
Windows Vista SP1
(beta, build 16659)
september 27, 2007
Windows 7
(announced)
july 19, 2007
Windows XP Professional (N) SP2c
august 9, 2007
Windows Server 2003 x64 Edition SP2
(build 3959)
march 12, 2007
Windows Server 2003 SP2
(build 3959)
march 12, 2007
Windows Server 2003 R2 SP2
march 12, 2007
Windows Home Server
(announced)
january 7, 2007
Windows Home Server
("Q", RC1)
june 2007
Windows Home Server 1.0 RTM
(build 5.2.3790.4062)
july 16, 2007
eComStation 2.0 (beta 4)
february 28, 2007
eComStation 2.0 (RC1)
june 18, 2007
Windows Mobile 6.0 (Photon)
february 12, 2007

2008


Windows Server 2008 Standard [32-bit] [without Hyper-V] Edition
Windows Server 2008, Enterprise [32-bit] [without Hyper-V] Edition
Windows Server 2008 Datacenter [32-bit] [without Hyper-V] Edition
Windows Web Server 2008 [32-bit] Edition
Windows Storage Server 2008 [32-bit] Edition
Windows Small Business Server 2008 Edition
Windows Essential Business Server 2008 Edition
Windows Server 2008 for Itanium-based Systems
(announced)
november 12, 2007
Windows Server 2008
(RC1)
december 5, 2007
Windows Server 2008
(RTM)
february 4, 2008
Windows Vista SP1
(RC)
november 15, 2007
Windows Vista SP1
(RC public)
december 14, 2007
Windows Vista SP1
(RC public 2)
january 9, 2008
Windows Vista SP1
(RTM)
february 4, 2008
Windows 7
(M1, build 6519.1)
january 19, 2008
Windows XP SP3
(beta, build 3205)
october 7, 2007
Windows HPC Server 2008
(beta 1)
november 14, 2007
Windows Embedded CE 6.0 R2
november 12, 2007
eComStation 2.0
(RC3)
november 2, 2007
eComStation 2.0
(RC4)
december 27, 2007


Windows Server 2008 Foundation
april 1, 2009
Windows Embedded Standard (announced) april 15, 2008
Windows Embedded Standard october 15, 2008
Windows Embedded Enterprise (announced) april 15, 2008
Windows Server 2008 SP2 beta december 4, 2008
Windows Server 2008 SP2 RC1 build 6002.16670 february 19, 2009
Windows Vista SP1 (build 6001) march 18, 2008
Windows Vista SP2 beta december 4, 2008
Windows 7 (M3, build 6789.1) september 7, 2008
Windows 7 pre-beta (M3, build 6933) october 28, 2008
Windows 7 beta (build 7000) january 7, 2009
Windows 7 RC (build 7100) april 30, 2009
Windows XP SP3 (RTM) april 21, 2008
Windows XP SP3 may 6, 2008
Windows Server 2008 R2 (announced) october 28, 2008
Windows Server 2008 R2 beta january 7, 2009


Windows HPC Server 2008 (build 1345, beta 2) may 17, 2008
Windows Home Server Power Pack 1 build 6.0.1800 july 21, 2008
Windows Home Server Power Pack 2 build 6.0.2030 march 24, 2009
Windows Embedded NavReady 2009 june 16, 2008
Windows Embedded Compact (announced) april 15, 2008
eComStation 2.0 (RC6a) december 6, 2008
Windows Mobile 6.1 (announced) april 1, 2008
Windows Mobile 6.1 september 3, 2008
Windows Mobile 6.5 beta february 16, 2009

2009

Windows Embedded Standard 2011
(CTP)
september 1, 2009

Windows Server 2008 SP2
may 26, 2009

Windows Vista SP2
may 26, 2009

Windows 7 Starter Edition, Home Basic Edition, Home Premium Edition, Professional Edition, Ultimate Edition, Enterprise Edition
(RTM build 7600.16385)
july 22, 2009

Windows 7
october 22, 2009

Windows Server 2008 R2
RTM
july 22, 2009

Windows Home Server
Power Pack 3
july 17, 2009

Windows Embedded CE 6.0 R3
september 22, 2009

Windows Mobile 6.5 RTM
may 25, 2009

Windows Mobile 6.5
october 6, 2009

Ci joint dans la version imprimée de ce cours, un article intitué « UNIX : l'Univers a 40 ans » du hors-série 46 de Février/Mars 2010 de LINUX Magazine France.

UNIX : l'Univers a 40 ans

Auteur

Denis Bodor

Ne vous êtes-vous jamais demandé pourquoi et par quel miracle un système de plus de 40 ans fait toujours parler de lui ? Pourquoi UNIX est toujours là, un système d'exploitation dont les premières implémentations fonctionnaient sur des ordinateurs surpassés aujourd'hui en puissance et en capacité par la première calculatrice scientifique, le premier PDA ou le premier smartphone qui vous tombe sous la main ? La réponse est simple. Ce merveilleux système repose sur une philosophie solide, adaptable et cohérente. Piqûre de rappel.

Ce qui va suivre ne doit être ignoré ou oublié d'aucun utilisateur. Plus important encore, ce qui va suivre devrait être su de tout nouvel utilisateur d'une Ubuntu 3D-compiz-splash-truc-tape-à-l'œil, de tout utilisateur de Mac OS X snow-lolcat et même de tout utilisateur d'applications en logiciel libre sous Windows 7. Ce qui va suivre est notre histoire et elle est peuplée de personnes de talent sinon de génie envers qui nous sommes redevables. Que nous allumions simplement nos machines sous GNU/Linux ou que nous passions un simple coup de fil, UNIX et sa philosophie sont avec nous et partout autour de nous.

1 UN PEU D'HISTOIRE

La grande histoire d'UNIX débute à la fin des années 60. C'est Ken Thompson, qui travaille alors pour les laboratoires Bell, qui va développer la première version d'un système d'exploitation mono-utilisateur sous le nom de New Ken's System sur DEC PDP-7. Le nom Unics fut proposé par Brian Kernighan sur la base d'un jeu de mots comparant le nouveau système de Thompson à Multics. Ce nom évolua par contraction en « Unix » pour être ensuite déposé sous le nom « UNIX » par AT&T.

Sur décision de justice, AT&T (et donc Bell Labs) ne pouvait commercialiser uniquement des équipements téléphoniques ou télégraphiques. La commercialisation d'UNIX était donc impossible et la décision fut prise, en 1973, de diffuser le système avec son code source dans les universités contre acquisition d'une licence bon marché.

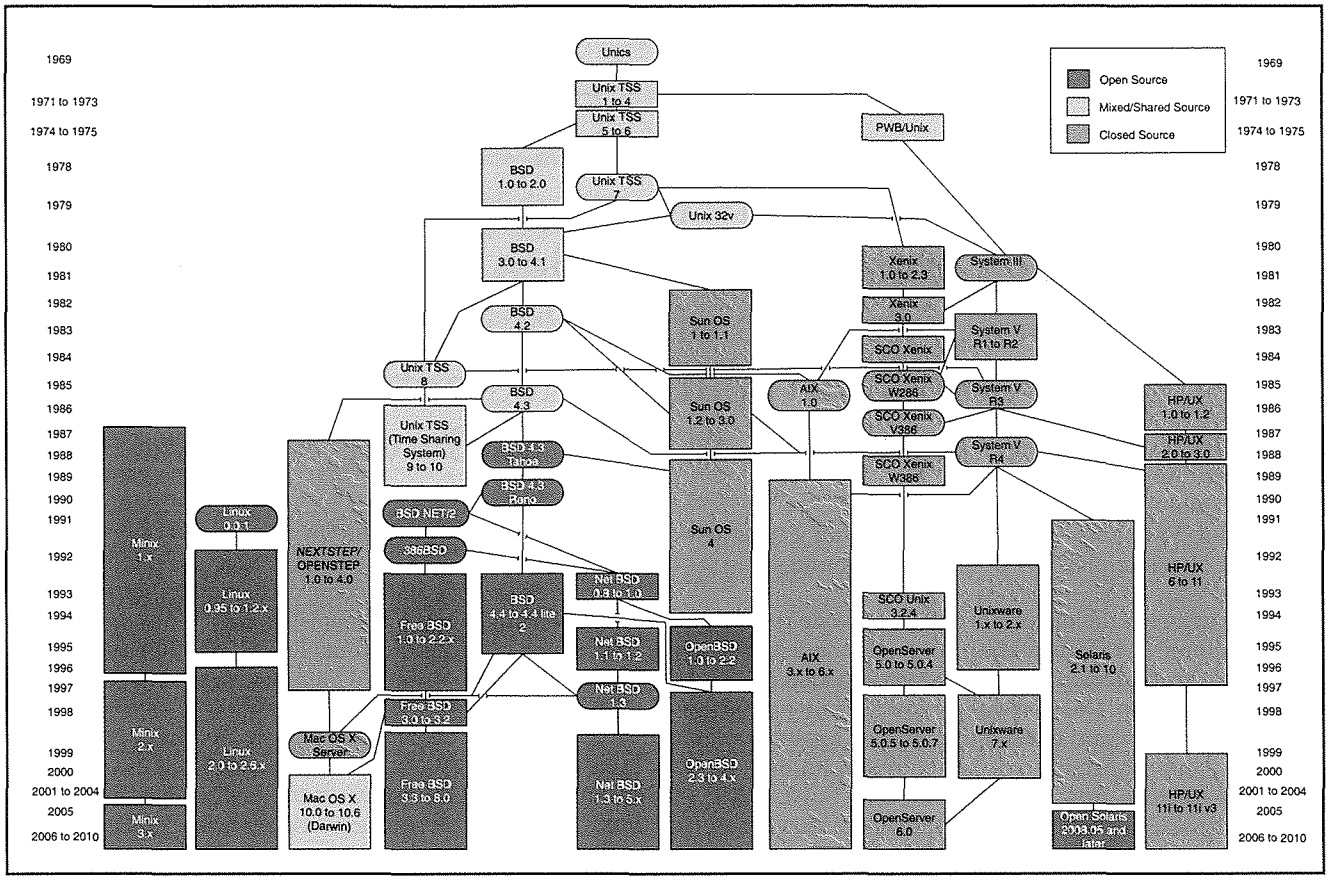
« Unix est simple. Il faut juste être un génie pour comprendre sa simplicité. » – Dennis Ritchie

En 1971, Ken Thompson conçut le langage B avec l'aide de Dennis Ritchie après avoir constaté que la maintenance d'un système d'exploitation écrit en assembleur n'était pas possible. Plus tard, Dennis Ritchie développa NB (New B), le langage successeur du B qui se transforma en C. Le langage majoritairement utilisé avec les systèmes UNIX, et celui utilisé encore aujourd'hui pour leur développement, vit le jour.

Par la suite, d'autres noms s'ajoutèrent à la liste des contributeurs majeurs à l'histoire d'UNIX : Bob Fabry, Keith Standiford, Jeff Schriebman, Bob Kridle, Bill Joy, Chuck Haley, Alan Snyder, Steven C. Johnson, Michael Lesk, ...

D'autres noms firent leur apparition, ceux d'outils encore largement utilisés aujourd'hui : Vi, Cron, Csh, ...

De cette genèse découle toute une famille de systèmes comprenant GNU/Linux, FreeBSD, OpenBSD, NetBSD ou encore Mac OS X. D'autres systèmes, qui ne sont pas des systèmes UNIX, profitèrent également de ces innovations, des idées et d'une partie de la philosophie sur laquelle repose le système. Ainsi, CP/M fut largement inspiré d'UNIX et lui-même servit de base pour MS/DOS dont on connaît sans doute tous l'histoire et l'évolution.



L'arbre généalogique et chronologique des systèmes UNIX (source Wikipédia, auteurs Eraserhead1, Infinity0).

UNIX a marqué à jamais l'histoire de l'informatique et continue à le faire, ceci pour une raison très simple : derrière cette famille de systèmes, il y a une idée ou plutôt un ensemble d'idées et de préceptes. Derrière UNIX, il y a une philosophie qui sert de ligne de conduite et de fil d'Ariane. Comprendre cette philosophie et la respecter le mieux possible assurent une stabilité et une pérennité sans précédent.

2 LA PHILOSOPHIE UNIX

Résumer la philosophie d'UNIX n'est pas chose évidente. Il s'agit d'un ensemble de principes. Nombreux sont ceux qui ont essayé de les résumer ou les lister. À notre tour maintenant de faire de même. Nous résumerons tout cela en quatre points principaux dont le reste découlera presque naturellement. Le lecteur curieux pourra faire quelques recherches sur les termes « philosophie UNIX » ou encore « less is more » pour obtenir plus de détails et des interprétations sensiblement différentes de la nôtre.

2.1 Écrivez des programmes qui effectuent une seule chose et qui le font bien

Voilà la base de toutes choses dans le monde UNIX (« dans le monde » tout court peut-être également). On retrouve cette idée dans bien des domaines mais, il est vrai, c'est en informatique qu'elle trouve le plus de sens : ne faire qu'une seule chose, et la faire bien.

Voilà qui n'est pas chose évidente dans l'évolution et le développement d'outils, d'utilitaires et de solutions logicielles. Un projet débute toujours avec une seule idée, un seul concept. Au fur et à mesure de sa vie, il gagne en maturité et il est tentant de lui ajouter de plus en plus de fonctionnalités. Il faut garder le précepte en tête.

Un bel exemple est sans doute Mozilla cumulant les fonctionnalités de navigateur, agrégateur RSS, client mail, etc. Mozilla a fini par exploser en Firefox, Thunderbird et d'autres éléments par la suite. La suite Mozilla était devenue ce qu'il est commun d'appeler en français une « usine à gaz ».

La concision est merveilleuse. Elle permet de se concentrer sur une tâche et de ne pas dépenser son énergie inutilement. En poursuivant plusieurs buts, les ressources et l'effort de développement s'évaporent, se répartissant plus ou moins également entre les objectifs à atteindre et ce, avec une déperdition inévitable.

Ceci implique également le fait de construire des noyaux de système d'exploitation petits et légers. Linux est un mauvais élève sur ce point, en particulier lorsqu'on regarde de près les distributions grand public actuelles. Le noyau intègre un support pour une masse énorme de périphériques. Son développement, bien entendu, est structuré et hiérarchisé. L'utilisation des modules permet de conserver une certaine simplicité. Quoi qu'il en soit, appréhender l'ensemble du code du noyau Linux est aujourd'hui chose impossible pour une personne seule.

« Unix n'a pas été conçu pour empêcher ses utilisateurs de commettre des actes stupides, car cela les empêcherait aussi de réaliser des actes ingénieux. » – Doug Gwyn

Un autre principe découlant de cette notion de concision peut se résumer par l'affirmation « Le silence est d'or ». En d'autres termes, lorsqu'un programme n'a rien à dire, il doit garder le silence. Ce n'est que lorsqu'il y a un problème qu'un outil doit devenir bavard et signaler explicitement une erreur. Un programme qui fait ce qu'on lui demande ne retourne rien, ne signale rien.

C'est le cas de la plupart des outils de base en ligne de commandes. Si nous compressons un fichier avec `gzip monfichier`, rien ne s'affiche. C'est à l'utilisateur de comprendre que s'il ne semble rien se passer, c'est tout simplement que tout a bien fonctionné et non qu'il ne s'est rien passé. Ceci évite les « avalanches » de messages pour des commandes standards qui, légitimement, n'auraient rien à dire. Certes, cela va à l'encontre de ce qu'aiment les utilisateurs. Tous, nous aimons voir et savoir que notre ordinateur travaille durement et fonctionne à merveille. Sans doute parce que de précédentes expériences, avec d'autres systèmes, nous ont fait perdre confiance dans les codes que nous utilisons.

« Unix ne dit jamais 's'il vous plaît'. » – Rob Pike

2.2 Écrivez des programmes qui collaborent

Si tous les programmes ne font, chacun, qu'une chose et qu'ils la font bien, ceci implique qu'ils doivent alors fonctionner de concert pour pouvoir achever des tâches plus importantes. Ces « briques » doivent alors collaborer les unes avec les autres du mieux possible.

Ce principe découle du précédent et implique le suivant. Le but est de former un système complet où la somme des parties est supérieure à l'ensemble. Lorsqu'un programme a accompli sa tâche correctement et qu'une autre est nécessaire, il passe le relais à un autre programme. La segmentation ainsi obtenue est totalement fonctionnelle.

Lorsqu'on dispose d'un ensemble de briques fiables, il est possible de construire un mur solide. Dans le cas des systèmes UNIX, le mortier, tel que perçu du point de vue de l'utilisateur de la ligne de commandes, est le shell. Il permet d'utiliser les briques via la construction de scripts. Le shell ne fait, lui aussi, qu'une seule chose : permettre d'utiliser les petites briques de concert.

Mais ce précepte s'étend à l'ensemble du système. On le retrouve avec la notion de couches telles qu'on peut les voir dans les interfaces graphiques. Au niveau du code, cette logique sera également appliquée, tout comme la concision. Inutile de construire une fonction `main()` qui fait tout. Il faut modulariser son code et penser les interfaces de manière à garantir la simplicité et la collaboration.

Pour en revenir au shell, rappelons que le meilleur exemple de collaboration entre les outils sont sans nul doute le « pipe » (|), les notions de redirection et d'Entrée/Sortie standards. Ceci nous conduit au troisième point de la philosophie.

2.3 Écrivez des programmes pour gérer des flux de texte

Les flux de texte représentent une interface universelle. Bien entendu, ceci est un peu moins vrai qu'il y a quelques dizaines d'années où ASCII-7 était vastement utilisé. L'encodage des caractères est un problème qui se réglera naturellement avec l'uniformisation UTF-8. Seule la période de transition est un problème et n'en est véritablement un qu'en cas d'interconnexion de systèmes aux « locales » différentes. Bien entendu, des configurations optimales permettent de contourner ce faux problème.

La notion de flux de texte est véritablement caractéristique des UNIX. Des choses comme `history | grep -v " h" | sed 's/[ \t]*$//' | sort -k 2 -r | uniq -f 1 | sort -n` n'ont de sens que pour un utilisateur de ce type de systèmes. Cependant, la magie et l'intelligence derrière une simple ligne de commandes comme celle-ci sont compréhensibles par tous. Elle résume, à elle seule,

les deux premiers préceptes et donc la philosophie d'un UNIX. Si je devais répondre à la question « qu'est-ce qu'un UNIX ? », je répondrai par ce type de commandes. Il s'agit de programmes (`grep`, `sed`, `sort`) qui font leur travail et collaborent via un flux de texte pour accomplir une tâche importante.

La notion de flux de texte ou de « texte » en général est présente partout dans un système UNIX. Ceci implique également le stockage des données et de la configuration. Préférez toujours les architectures où les données sont textuelles. Seule garantie de portabilité et de sécurité, le stockage de texte est universel. La clarté et l'intelligibilité ne dépendent pas du format mais de la structure des données.

« Unix est convivial. Cependant Unix ne précise pas vraiment avec qui. » – Steven King

Une configuration réseau stockée dans `/etc/network/interfaces` ou un `/etc/rc.conf` est structurée et donc claire. Bien plus claire que celle d'une interface graphique stockant tout cela dans une base de données ou une structure illisible via les simples outils mis à disposition par le reste du système.

Même les suites bureautiques peuvent bénéficier de ces avantages. Le format OpenDocument, par exemple, n'est rien d'autre qu'une archive contenant des données textuelles XML. Il est ainsi aisé de modifier manuellement ou via un programme, le contenu d'un tel document. L'expérience m'a ainsi appris que cet élément de la philosophie UNIX pouvait rendre de grands services, en particulier là où d'autres formats auraient induit un surcoût de travail et un retard très important.

2.4 Release soon, release often

Nous sortons un peu du cadre strict des systèmes UNIX pour arriver dans le développement OpenSource. Je pars du principe que l'OpenSource et les logiciels libres sont les conséquences directes de la philosophie sous-jacente de celle d'UNIX. Sans doute, car la génèse universitaire et l'histoire même d'UNIX sont fortement liées à la notion d'échange et de partage de connaissances également à la base du logiciel libre.

« Release early, release often » (« The Cathedral and the Bazaar », Eric S. Raymond) est une maxime concernant le développement de logiciels. Ceci rejoint un élément de la philosophie UNIX revenant à dire « Concevez un prototype dès que possible ».

Bien entendu ceci n'est applicable, raisonnablement, que dans le cadre de la diffusion de logiciels libres. « Releaser » un code tôt pour un logiciel propriétaire n'a pas vraiment de sens si l'on souhaite garder une certaine forme de crédibilité.

La philosophie UNIX est ancrée dans le matériel, dans les faits. On cherche un sens pratique à la structure du système et des éléments qui le composent.

Partant de cela, il est normal de chercher à rendre matures ses composants rapidement et surtout de faire en sorte qu'ils soient réellement utilisables. La diffusion de ses créations à intervalles rapprochés et le plus tôt possible, renforce le pragmatisme de la philosophie UNIX.

CONCLUSION

Arrêtons là cette dissertation pour l'instant. Vous l'aurez sans doute compris, ici, à la rédaction, nous aimons les systèmes UNIX (en logiciel libre) et nous pensons fermement que c'est cette philosophie qui permet de constituer un système stable, agréable et ergonomique.

« Ceux qui ne comprennent pas Unix sont condamnés à le ré-inventer, lamentablement. » – Henry Spencer

Certains diront qu'il s'agit presque d'une religion ou d'une forme d'intégrisme mais force est de constater que les quelques éléments que nous avons abordés ici sont une véritable force.

Alors même qu'UNIX s'impose chaque jour un peu plus dans de nombreux domaines et actuellement en téléphonie mobile, il est important de rappeler qu'il ne s'agit pas juste d'une masse de données mais d'une idée. Pour paraphraser un film que certains ne manqueront pas de reconnaître, un Unix c'est un peu comme un homme masqué, « Sous ce masque, il y a plus que de la chair. Sous ce masque, il y a une idée Creedy... Et les idées sont à l'épreuve des balles. ».

Auteur : Denis Bodor

Chapitre 2

Définition du rôle de l'administrateur

- gérer les comptes utilisateurs (tâche simple et automatisable)
- assister et éduquer les utilisateurs (réponses à leurs questions, documentation à jour)
- gérer les logiciels (installation, configuration, mise à jour)
- gérer le matériel (panne, remplacement, ajout)
- assurer la sécurité du système et des utilisateurs (sauvegardes fiables et régulières, contrôle d'accès, utilisations abusives de ressources)
- vérification de l'adéquation du matériel avec son utilisation (identifier les goulets d'étranglement)
- maintenance de premier niveau (diagnostiquer une panne, appel de la maintenance constructeur)
- gestion quotidienne (multiples tâches, petites ou grosses)

Autres facettes du métier :

- diplomatie, police
- aspects légaux (chiffrement, etc.)
- enquêtes judiciaires (vol, saccage, piratage informatique, articles pédophiles, etc.)
- relations commerciales
- politique d'utilisation des machines

Bien sûr, la charge de travail dépend de la taille du site.

L'administrateur est en première ligne lorsqu'un problème surgit. C'est lui qu'on incrimine naturellement lorsque quelque chose ne marche pas.

1. Votre pire ennemi, c'est vous : attention à ce que vous faites ! Exemple « `rm /tmp *` » (explications sur l'expansion des metacharacters du shell page 581)
2. Si vous êtes fatigué, ne faites rien.
3. Pas de modification importante un vendredi après-midi.
4. Soyez sûr de pouvoir revenir en arrière : sauvegarder tout fichier qui doit être modifié :

```
% mv config.ini config.ini.orig
% cp config.ini.orig config.ini
% vi config.ini
```

5. Documentez ce que vous faites.
6. Faites comme si vous ne pouviez pas venir demain.

Administrateur système == technicité + rigueur + bon sens

Administrateur d'UNIX : d'abord un **utilisateur expert d'UNIX**

- environnement utilisateur
- aide en ligne
- système de fichiers
- utilisation du shell
- utilisation d'un éditeur de texte
- commandes de base
- programmation shell

Ci joint dans la version imprimée de ce cours, un document humoristique sur la journée d'un administrateur systèmes.

Journée d'un "informaticien de centre"

Chacun à l'INRA, quel que soit son centre, quel que soit son corps d'appartenance, utilise couramment le reseau : messagerie, accès Internet, accès aux outils-scientifiques, accès Yole,. . .

Le réseau est désormais un point de passage obligé. Si pour l'utilisateur tout semble fonctionner "tout seul", il est certain que la condition de cette apparente facilité est un immense travail qui se déroule en coulisses, avec une pluralité d'acteurs, en général de la direction de l'informatique, garantie de la qualité et de la cohérence de l'ensemble.

"La journée d'un informaticien de centre", acteur local par excellence, conte une partie de la face cachée de la réalité exigée par la mise en place, la maintenance et l'évolution d'une telle infrastructure réseau/serveurs).

Eh bien oui ! Voilà dix-huit mois, à Carry-le-Rouet, dans l'ivresse des "journées de la direction de l'Informatique , promesse solennelle et inconsidérée était faite au rédacteur en chef de "l'Éditeur de Liens", de m'atteler à la rédaction de cet article de première importance !

Et puis, et puis, la gestion des priorités, tâche éminemment complexe et gourmande en temps, l'algorithme de gestion, totalement imparfait, ont repoussé de semaine en semaine, en queue de file d'attente, l'écriture de ces pages. . .

Soudain, tel un archange vengeur surplombant la tourmente, coup de poing sur la table dudit rédacteur ! "Bon ! dis-donc, c'est pour quand ta copie ?". . .Révision en urgence de l'algorithme de gestion ! Mise en chantier, toutes affaires cessantes, de ce qui commence à prendre des allures de pensum !. . .Voici donc le récit de la palpitante aventure quotidienne d'un "informaticien de centre" (IC sous la forme abrégée). Mais tout d'abord. il importe que vous sachiez comment se décline le profil d'un IC, métier de généraliste hautement qualifié en toutes spécialités informatiques.

PROFIL D'UN IC

Une solide formation en informatique ménagère

- un oeil attentif, des mains expertes : les casseroles sur le feu sont nombreuses !
- tous les jours, donner à chacun selon ses besoins
- tous les soirs. faire conserve de la production du jour
- ranger, nettoyer. remiser, installer, re-installer, il importe que l'espace soit dégagé pour permettre aux joueurs de s'ébattre à loisir
- une place pour chaque chose et chaque chose à sa place ! (règle d'or de l'administrateur).

Des etudes approfondies informaticien(ne) de plomberie moderne

Pour résoudre les problèmes de réseaux, il sera indispensable d'avoir une bonne experience dans les problèmes de robinet, les problèmes de debit. les problèmes de circulation. les problemes de conduit. les stations de distribution et les problèmes de régulation.

Un CAP sérieux de mecanique appliquée :

- savoir monter, démonter. remonter, configurer, optimiser un maximum de systèmes
- connaître l'art délicat de mettre de l'huile dans les rouages
- acquérir un flair des occurrences à problèmes et etre capable d'anticiper à l'aide de bouts de ficelles à nouer, de manière scientifique et acrobatique.

Une certification "LEGO"

Edifier des constructions complexes, agréger des briques supplémentaires et remplacer les éléments défectueux : ces opérations doivent pouvoir etre effectuées les yeux fermés, des sonneries de téléphone stridentes dans l'oreille droite, des bips en deux tonalités dans l'oreille gauche, trois personnes à la porte du bureau, sept fenetres sur trois

systèmes différents en attente sur le poste de travail.

De bonnes capacités relationnelles :
- fournir les cartes d'orientation, et les règles du jeu
- tenir la main des néophytes
- rassurer les angoissés, les frustrés, les timides, calmer les coléreux. ramener les égarés
- accompagner les aventureux dans les passages difficiles
- proposer de nouveaux jouets aux aguerris.

On l'aura compris, l'exercice de cette profession réclame, du souffle de l'endurance et un optimisme de bon aloi. Une longue pratique conduit à penser qu'un sens de l'humour à toute épreuve, peut faciliter considérablement la tâche...

Voici donc notre Icé, parti de bon matin, frais et dispos, guilleret, sautillant sur le chemin de son bureau, bâtissant dans sa tête, tel Perette et le pot au lait, le planning de sa journée ordinaire ...

LE PLANNING D'UN IC

Mes rendez-vous :
- 8h30-9h30 : déménagement de "`tartinette`", une des machines de centre. Les utilisateurs ont été prévenus la veille, l'inspection du nouveau logis n'a semblé recéler aucune surprise, les câbles sont prets, tout devrait baigner...
- 11h-12h30 : Audio-conférence de travail du groupe Baoutiltout
- 13h30 : RV avec un chercheur qui souhaite se séparer de son PC pour une station de travail...

Mon Pense-bête :
- appeler la Hot-line TLK pour renseignement technique sur la liaison onduleur/ordinateur
- re-recontacter GQ pour un code de licence
- appeler DonneStock pour le changement de plate-forme du produit Réseaulaborieux
- installer la version 5.07 de Navigateur-Moderne
- instruire les demandes d'extension réseau en attente.

Dans les intervalles de temps. réfléchit notre homme, j'installe sur le nouveau GQ, les imprimantes, les compilateurs Fortiche et Blaise, pendant que sous Solarimax je remets à niveau la dernière version de ZZ, je prépare mes transparents pour ma prochaine présentation de Magic Ware, et le tour est joué se dit-il, grimpant quatre à quatre les marches...

MA VRAIE JOURNEE

A présent. reprenez de l'endurance et que la vraie réalité vécue les deux pieds sur terre vous emporte plus sûrement "aux frontières du réel" que la moins mauvaise télévisuelle série B.

Pom, pom, pom, pom ! ! !...8h28. Un message d'avertissement, je vais stopper `tartinette`...

3 mn après : plus personne ne travaille, sauf Dugenou...*Allo, Gaston Dugenou ? oui, votre appli..., ca tourne..., ah oui..., le message..., non, j'ai pas vu...oui, j'arrête tout de suite...*

Le champ est libre...`init 0`, et tout et tout... Se profilent à la porte, trois utilisateurs, des Macs plein les bras ! *Bonjour, la formation réseau Mac, c'est ici ?* Le temps de détromper ces valeureux apprentis... `tartinette` attend... A peine ont-ils tourné les talons... Téléphone ! *Allo, ici Hiro-Hito, ... sur ma machine* `bluejean`*, j'ai un problème d'impression, tu pourrais pas regarder, parce que tu comprends, je dois fournir mon papier pour hier soir et j'ai pas terminé... et puis au fait, tu ne pourrais pas me donner la nouvelle adresse de Packgenie qui distribue Teletoc...*

Bien ! `bluejean` remis sur pieds, notre Icé vole vers `tartinette` : débranchements, transport, rebranchements. 9h40 : l'opération déménagement se solde par un succès. Tout est reparti.

Mais pendant ce temps, la boite aux lettres de notre ami s'est meublée. Pas le temps de lire... Re-téléphone : *Allo, ... ici Dupied. Sté Thermoclim... je suis dans le coin... je passe vous voir à 11h15... on pourra voir ensemble les derniers détails de votre installation... on gagnera quelques jours... ?*

Audioconf ou climatisation ? Entre les deux, pas d'hésitation : le jugement de Salomon ! une demi-heure à Thermoclim, une heure à Baoutiltout. Le dépeçage commence ! Revenons à notre boîte aux lettres pleine de moutons ! douze messages...

- La base Oracle n est pas lancée : tiens ! ben oui, hier grosse katastrophe, un disque menacé d'asphyxie par les bases d'index des sauvegardes. En plein vol, la base Oracle a été déménagée. Depuis, `patinette` (serveur de centre) a rebooté (ou comment repartir à zéro...) et les liens symboliques... ça a ses limites... bon. on revoit la copie... c'est réparé.
- Demande d'ouverture de compte : du travail pour Mikado (le technicien réseau)
- Ah. la réponse de la hot-line Applox. à voir plus tard...
- Les nouveaux tarifs Pleinsoleil : à engranger pour consultation.
- Courtepaille, station isolée, réclame à cors et à cris, sa cascade Renater en chantier. Et que le Megapac a besoin d'un upgrade pour la numérotation à dix chiffres (dans six jours), et que sinon ça va plus marcher, et que pour que la cascade marche, il faudrait que les routeurs soient opérationnels, et que pour ça il faudrait avoir reçu les upgrades mémoire d'icelui, et que renseignements pris. la numérotaticn Transpac n'est pas concernée par la réforme, et donc on revient à la case départ, où il urge d'attendre...

Dringgg ! ! ... le téléphone vous épargne les suivants...

10h45 : *Allo, ... ici Dukoud, ... bonjour, ... mon réseau ne marche plus ! ?* Restons calme... Interrogeons la boule de cristal, pardon, l'utilisateur... *Pouvez-vous me dire ce qui est affiché sur votre écran ? ... tchzoewrfndbdgr... ! Si vous appuyez sur la touche x, ca fait quoi ? ... brredrrebrredrre.. ! et si vous faites Ctrl Z ? ... kiuyyuytgfre... !* Après 5 minutes de dialogue entre une tête et des yeux bras déportés, le diagnostic est posé et confirmé, la messagerie a des soucis... Trifouillage dans les entrailles (virtuelles) du monstre : mais bon sang, c'est bien sûr, c'est le démon de messagerie qui a perdu le DNS ! Un geste chirurgical s'impose, meurtre dudit démon, suivi de sa résurrection ! Les phoenix ont toujours fait de belles carrières informatiques.

... 11h05, j'entre en téléconférence... et il ne me reste plus qu'une page... il va falloir faire court !

Eh bien. vous me croirez si vous voulez, mais pendant que l'audioconférence se poursuivait sans notre ami réclamé par Dukoud aux alentours de 11h26, `patinette` a planté de la plus belle manière qui soit. Mais le Dieu des Informaticiens veillait ! Mikado, de retour d'une expédition périlleuse (connexion de PC au réseau. c'est jamais la même chose que les autres fois), tient la main de `patinette` pendant son redémarrage laborieux, ... trois disques stoppes en pleine course, il faut effacer les traces...

12h35 : L'audioconférence terminée, Icé range ses notes et se prend à une profonde réflexion métaphysique, sans doute favorisée par le creux qui commence à s'installer au fond de son estomac : l'homme aurait-il créé l'ordinateur à son image ? (tel Dieu... cf. La Bible) ? Plusieurs raisonnements conduits de conserve, c'est monnaie courante (multiprocesseur), le cerveau se débrouille pour que ça marche au mieux (multithread), en utilisant la mémoire à court terme (mémoire à accès rapide), la mémoire à long terme (mémoire à accès lent), en travaillant hors du champ de conscience (background), une idée en faisant jaillir dix autres (fork, processus engendrant un autre processus), avec des niveaux d'interruption et des procédures de reprise sur pointeur dans les piles, sans parler des routines, qui me permettent de préparer mon café du matin sans y penser intensément ! ! La question n'est-elle pas des plus angoissantes ?

Las, le rappel à l'ordre, physiologique et brutal, des fonctions de nutrition brise net toute tentative d'approfondissement de cette nouvelle voie ouverte à la recherche.

13h30 : Contretemps : Albert Dubras reporte son rendez-vous à 14h30. Profitons de ce petit trou pour appeler GQ

et peut-être enfin joindre le préposé à la délivrance des mots de passe. Manque de chance, une fois de plus il est en réunion, une fois de plus le message lui sera transmis, Blaise n'est pas prêt d'être installé... Chez TVO, la réponse arrive dans la demi-journée. Autre entreprise, autres moeurs...

15h : Dubras est reparti, ravi des merveilles promises par sa future acquisition. Plongeon dans la boîte aux lettres. Cinq nouveaux messages... pas le temps. re-telephone : cette fois ca y est, ca devait arriver... tout réseau qui se respecte a au moins une fois dans son existence, subi l'inénarrable coup de pelleteuse, mais cette fois c'en est trop, on a fait original, un coup de perceuse a troué net le cable reliant nos chers connectés... Je ne vous raconte pas la suite !

De fil en aiguille, de coup de fil en message, de consultation en interruption...

- *Excuse-moi, mais tu pourrais pas venir voir, je suis bloquée dans Applox ?...*
- *Excuse-moi, mais javais débranché le clavier de ma station, elle ne repart plus...*
- *Excuse-moi, tu pourrais pas regarder mon terminal, c'est tout bizarre quand je clique sur le bouton de gauche... ?*

17h30 : La fin de la journée est proche, et notre Icé se trouve au niveau 9 d'interruption ! Alors plein de courage, il va fermer les niveaux qui peuvent l'être, noter ceux restés ouverts, faire le point sur sa folle journée, rédiger trois transparents de sa présentation Magic (la date se rapproche implacablement), compléter son pense-bête, préparer les demandes d'extension réseau, répondre aux derniers messages arrivés. et passer en revue l'état des machines avant de les laisser à leur labeur nocturne et aux utilisateurs insomniaques...

Vous l'avez compris. ce récit est une pure fiction sortie de l'imagination romanesque d'un informaticien surmené. Il est bien évident que toute ressemblance avec des situations ou des personnes existantes ou ayant existé, ne peut relever que d'une coïncidence fortuite et n'engage que son auteur.

Simone Desjeunes.
UIC Nancy.
(paru dans le numéro 92 de mars-mai 1997 du magazine de l'INRA)

Ci joint dans la version imprimée de ce cours, un document américain de l'association SAGE sur les niveaux du métier d'administrateur systèmes.

Adresse web du document fourni : « `http://www.sage.org/` »

SAGE Job Description for System Administrators

Organizations that rely on computing resources to carry out their mission have always depended on systems administration and systems administrators. The dramatic increase in the number and size of distributed networks of workstations in recent years has created a tremendous demand for more, and better trained, systems administrators. Understanding of the profession of systems administration on the part of employers, however, has not kept pace with the growth in the number of systems administrators or with the growth in complexity of system administration tasks. Both at sites with a long history of using computing resources and at sites into which computers have only recently been introduced, systems administrators face perception problems that present serious obstacles to their successfully carrying out their duties.

Systems administration is a widely varied task. The best systems administrators are generalists : they can wire and repair cables, install new software, repair bugs, train users, offer tips for increased productivity across areas from word processing to CAD tools, evaluate new hardware and software, automate a myriad of mundane tasks, and increase work flow at their site. In general, systems administrators enable people to exploit computers at a level which gains leverage for the entire organization.

Employers frequently fail to understand the background that systems administrators bring to their task. Because systems administration draws on knowledge from many fields, and because it has only recently begun to be taught at a few institutions of higher learning, systems administrators may come from a wide range of academic backgrounds. Most get their skills through on-the-job training by apprenticing themselves to a more experienced mentor. Although the system of informal education by apprenticeship has been extremely effective in producing skilled systems administrators, it is poorly understood by employers and hiring managers, who tend to focus on credentials to the exclusion of other factors when making personnel decisions.

Understanding system administrators' background, training, and the kind of job performance to be expected is challenging ; too often, employers fall back into (mis)using the job classifications with which they are familiar. These job classification problems are exacerbated by the scarcity of job descriptions for systems administrators. One frequently used misclassification is that of programmer or software engineer. Although the primary responsibility of the systems administrator is not to produce code, that is the metric by which programmers are evaluated, and systems administrators thus classified often receive poor evaluations for not being "productive" enough. Another common misclassification is the confusion of systems administrators with operators. Especially at smaller sites, where systems administrators themselves have to perform many of the functions normally assigned (at larger sites) to operators, systems administrators are forced to contend with the false assumption they are non-professional technicians. This, in turn, makes it very difficult for systems administrators to be compensated commensurate with their skill and experience.

SAGE, as the professional organization for systems administrators, formed the 'sage-jobs' working group to address these problems. Its goals include the creation of a set of appropriate job descriptions for systems administrators and promotion of their adoption by organizations that employ systems administrators.

Below are the current job description templates that the working group has produced. We have created an additional list of check-off items. The templates are intended to describe the core attributes of systems administrators at various levels of job performance, while the check-off list is intended to augment the core descriptions. In particular the check-off list is intended to address site-specific needs, or special areas of expertise that a systems administrator may have. Job descriptions for more experienced systems administrators or more senior positions will typically include more items from the check-off list.

As a SAGE member, we'd like to encourage your comments on the work to date. Please send your input to the sage-jobs working group, `sage-jobs@usenix.org`, or to the Chair, Tina Darmohray, `tmd@eticket.llnl.gov`. Feel free to join the working group as well by sending email to `majordomo@usenix.org`, with the body of the

message `subscribe sage-jobs`.

Tina Darmohray SAGE Jobs Working Group Chair `tmd@eticket.llnl.gov`

Definitions

A "small site" has 1-10 computers, all running the same operating system, and 20 or fewer users. (A computer used by only the administrator does not qualify as a site.)

A "midsized site" has up to 100 systems, running no more than 3 different operating systems, and up to 100 users.

A "large site" has 100 or more computers, potentially running more than one operating system, and 100 or more users.

The following are the core templates :

Novice

Required skills :

Has strong inter-personal and communication skills ; is capable of explaining simple procedures in writing or verbally, has good phone skills.

Is familiar with UNIX and its commands/utilities at a user level ; can edit files, use a a shell, find users' home directories, navigate through the file system, and use i/o redirection.

Is able to follow instructions well.

Required background :

2 years of college or equivalent post-high-school education or experience.

Desirable :

A degree or certificate in computer science or a related field.

Previous experience in customer support, computer operations, system administration or another related area. Motivated to advance in the profession.

Appropriate responsibilities :

Performs routine tasks under the direct supervision of a more experienced system administrator.

Acts as a front-line interface to users, accepting trouble reports and dispatching them to appropriate system administrators.

Junior

Required skills :

Strong inter-personal and communication skills ; capable of training users in applications and UNIX fundamentals, and writing basic documentation.

High skill with of most UNIX commands/utilities. Familiarity with most basic system administration tools and processes ; for example, can boot/shutdown a machine, add and remove user accounts, use backup programs and fsck, maintain system database files (groups, hosts, aliases).

Fundamental understanding of a UNIX-based operating system ; for example, understands job control, soft and hard links, distinctions between the kernel and the shell.

Required background :

One to three years of system administration experience.

Desirable :

A degree in computer science or a related field.

Familiarity with networked/distributed computing environment concepts ; for example, can use the route command, add a workstation to a network, and mount remote filesystems.

Ability to write scripts in some administrative language (Tk, Perl, a shell).

Programming experience in any applicable language.

Appropriate responsibilities :

Administers a small site alone or assists in the administration of a larger system. Works under the general supervision of a system administrator or computer systems manager.

Intermediate/Advanced :

Required skills :

Strong inter-personal and communication skills ; capable of writing purchase justifications, training users in complex topics, making presentations to an internal audience, and interacting positively with upper management. Independent problem solving ; self-direction.

Is comfortable with most aspects of UNIX systems administration ; for example, configuration of mail systems, system installation and configuration, printing systems, fundamentals of security, installing third-party software.

A solid understanding of a UNIX-based operating system ; understands paging and swapping, inter-process communication, devices and what device drivers do, file system concepts ("inode", "superblock").

Familiarity with fundamental networking/distributed computing environment concepts ; can configure NFS and NIS, can use nslookup or dig to check information in the DNS, understands basic routing concepts.

Ability to write scripts in some administrative language (Tk, Perl, a shell).

Ability to do minimal debugging and modification of C programs.

Required background :

Three to five years systems administration experience.

Desirable :

A degree in computer science or a related field.

Significant programming background in any applicable language.

Appropriate responsibilities :

Receives general instructions for new responsibilities from supervisor.

Administers a mid-sized site alone or assists in the administration of a larger site.

Initiates some new responsibilities and helps to plan for the future of the site/network.

Manages novice system administrators or operators. Evaluates and/or recommends purchases ; has strong influence on purchasing process.

Senior :

Required skills :

Strong inter-personal and communication skills ; capable of writing proposals or papers, acting as a vendor liaison, making presentations to customer or client audiences or professional peers, and working closely with upper management.

Ability to solve problems quickly and completely.

Ability to identify tasks which require automation and automate them.

A solid understanding of a UNIX-based operating system ; understands paging and swapping, inter-process communication, devices and what device drivers do, file system concepts ("inode", "superblock"), can use perfomance analysis to tune systems.

A solid understanding of networking/distributed computing environment concepts ; understands principles of routing, client/server programming, the design of consistent network-wide filesystem layouts.

Ability to program in an administrative language (Tk, Perl, a shell), to port C programs from one platform to another, and to write small C programs.

Required background :

More than five years previous systems administration experience.

Desirable :

A degree in computer science or a related field.

Extensive programming background in any applicable language.

Publications within the field of system administration.

Appropriate responsibilities :

Designs/implements complex local and wide-area networks of machines.

Manages a large site or network.

Works under general direction from senior management.

Establishes/recommends policies on system use and services.

Provides technical lead and/or supervises system administrators, system programmers, or others of equivalent seniority.

Has purchasing authority and responsibility for purchase justification.

These are things you might want to add to the base job descriptions as either required or desirable.

Local Environment Experience

Experience with the specific operating systems, applications, or programming languages in use at the site (for example SunOS, AIX, CAE/CAD software, FrameMaker, Mathematica, Fortran, Ada). Experience with the work done by the users at the site.

Heterogeneity Experience

Experience with more than one UNIX-based operating system. Experience with sites running more than one UNIX-based operating system. Familiarity with both System V and BSD-based UNIX operating systems. Experience with non-UNIX operating systems (for example, MS-DOS, Macintosh OS, or VMS). Experience with internetworking UNIX and other operating systems (MS-DOS, Macintosh OS, VMS).

Programming Skills

Extensive programming experience in an administrative language (Tk, Perl, a shell). Extensive programming experience in any applicable language.

Networking Skills

Experience configuring network file systems (for example, NFS, RFS, or AFS). Experience with network file synchronization schemes (for example, rdist and track). Experience configuring automounters. Experience configuring license managers. Experience configuring NIS/NIS+. Experience with TCP/IP networking protocols (ability to debug and program at the network level). Experience with non-TCP/IP networking protocols (for example, OSI, Chaosnet, DECnet, Appletalk, Novell Netware, Banyan Vines). Experience with high-speed networking (for example, FDDI, ATM, or SONET). Experience with complex TCP/IP networks (networks that contain routers). Experience with highly complex TCP/IP networks (networks that contain multiple routers and multiple media). Experience configuring and maintaining routers. Experience maintaining a site-wide modem pool/terminal servers. Experience with X/X terminals. Experience with dial-up networking (for example, SLIP, PPP, or UUCP). Experience at a site that is connected to the Internet. Experience installing/configuring DNS/BIND. Experience installing/administering Usenet news. Experience as postmaster of a site with external connections.

Security

Experience with network security (for example, building firewalls, deploying authentication systems, or applying cryptography to network applications). Experience with classified computing. Experience with multi-level classified environments. Experience with host security (for example, passwords, uids/gids, file permissions, file system integrity, use of security packages).

Site Specialities

Experience at sites with over 1,000 computers, over 1,000 users, or over a terabyte of disk space. Experience with supercomputers. Experience coordinating multiple independent computer facilities (for example, working for the central group at a large company or university). Experience with a site with 100@developing/implementing a site disaster recovery plan. Experience with a site requiring charge-back accounting.

Documentation

Background in technical publications, documentation, or desktop publishing.

Databases

Experience using relational databases. Experience using a database query language. Experience programming in a database query language. Previous experience as a database administrator.

Hardware

Experience installing and maintaining the network cabling in use at the site. Experience installing boards and

memory into systems. Experience with SCSI device setup and installation. Experience installing/configuring peripherals (for example, disks, modems, printers, or data acquisition devices). Experience with board-level diagnosis and repair of computer systems. Experience with component-level diagnosis and repair of computer system.

Management

Budget responsibility. Experience in writing personnel reviews, and ranking processes. Experience in interviewing/hiring.

Chapitre 3

Premiers contacts avec UNIX

Avant de commencer : n'ayez pas peur d'expérimenter. Le sytème ne vous fera pas de mal.

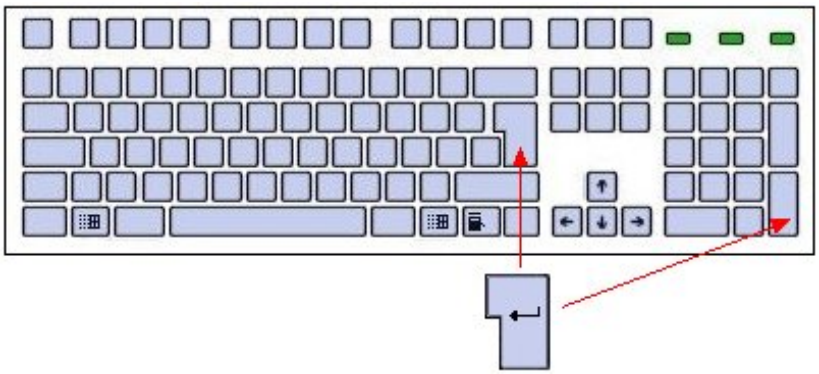
En mode utilisateur, vous ne pouvez rien abîmer en utilisant le système. UNIX, par conception, possède des notions de sécurité, afin d'éviter aux utilisateurs «normaux» de le déconfigurer.
En mode administrateur, bien sûr, faites attention. **On limitera tout travail en mode administrateur au minimum.**

Il faut savoir se servir d'un clavier !

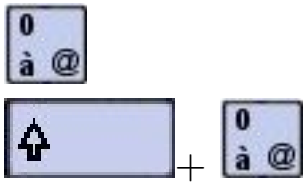
Il existe des claviers : américain, français, etc. déclinés en autant de modèles qu'imaginables.

(images de claviers trouvées sur le site
`http://www.freinet.org/creactif/bruyeres/labo111.html`)

Rappel : touche enter, entrée, valider



Rappel : touche shift



Appuyer sur la touche donne « à »

Appuyer sur ces 2 touches donne « 0 »

Rappel : touche shift lock, caps lock, majuscule

On n'utilise pas la touche caps lock. Non !

« `http://capsoff.blogsport.com/` »

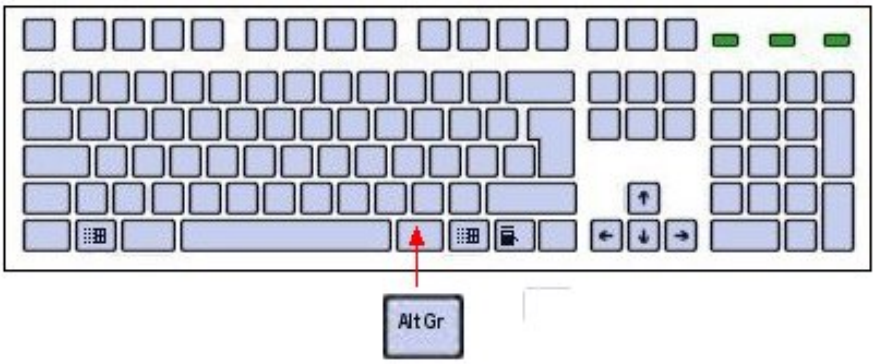
Hack windows : `Ctrl2cap` d'URL
`http://www.sysinternals.com/files/ctrl2cap.zip`

Rappel : touche tab, tabulation

Rappel : touche escape, esc, Échap, échappement

Equivalent : `Ctrl + [`

Rappel : touche Alt Gr (absente sur clavier QWERTY)



Appuyer sur la touche donne « à »

Appuyer sur ces 2 touches donne « 0 »

Appuyer sur ces 2 touches donne « @ »

Rappel : pavé numérique

On n'utilise pas le pavé numérique. Non !

Rappel : num lock

On n'utilise pas num lock. Non !

Un utilisateur UNIX est equivalent à :

- un identificateur (sur 8 lettres en général), son «nom» au sens informatique ; appelé **login** ;
- un mot de passe **confidentiel** ;

Gare aux sanctions en cas d'«amusement» avec le compte d'un autre utilisateur !

Il existe des chartes informatiques ≡ réglements informatiques.

- un mot de passe ne se prête pas !
- un mot de passe ne s'oublie pas !
- un mot de passe n'est pas facile à trouver ! :
  - évitez qu'il ne se rapporte pas à vous (nom, voiture, chien)
  - évitez les mots dans des dictionnaires
  - évitez les prénoms
  - il doit comporter au moins 6 caractères, en général 8
  - les majuscules et les minuscules sont différenciées
  - utiliser des chiffres et des caractères spéciaux
    par exemple « `Kpiten[` », « `&7oubon` », etc. [1]

[1] Ces mots de passe sont mauvais. Pourquoi ?

La commande standard pour changer son mot de passe sur une machine UNIX est « passwd ».

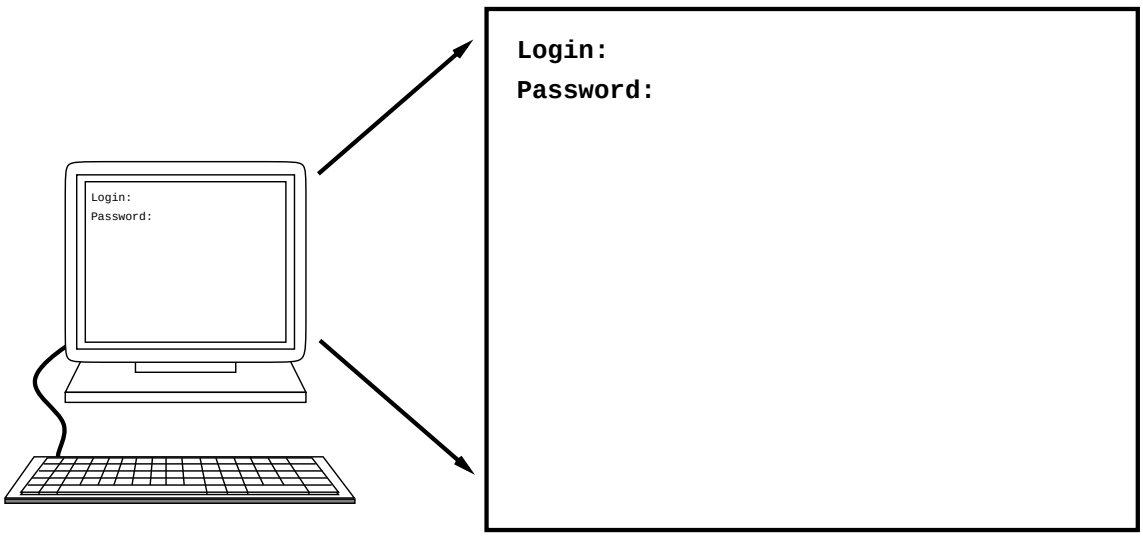
Sur les systèmes UNIX qui utilisent un mécanisme de centralisation des mots de passe (appelé NIS), la commande pour changer son mot de passe est « yppasswd ».

C'est le cas de la formation permanente ⇒ « yppasswd ».

Terminal texte (modèle VT100)

Se reporter à http://www.vt100.net/

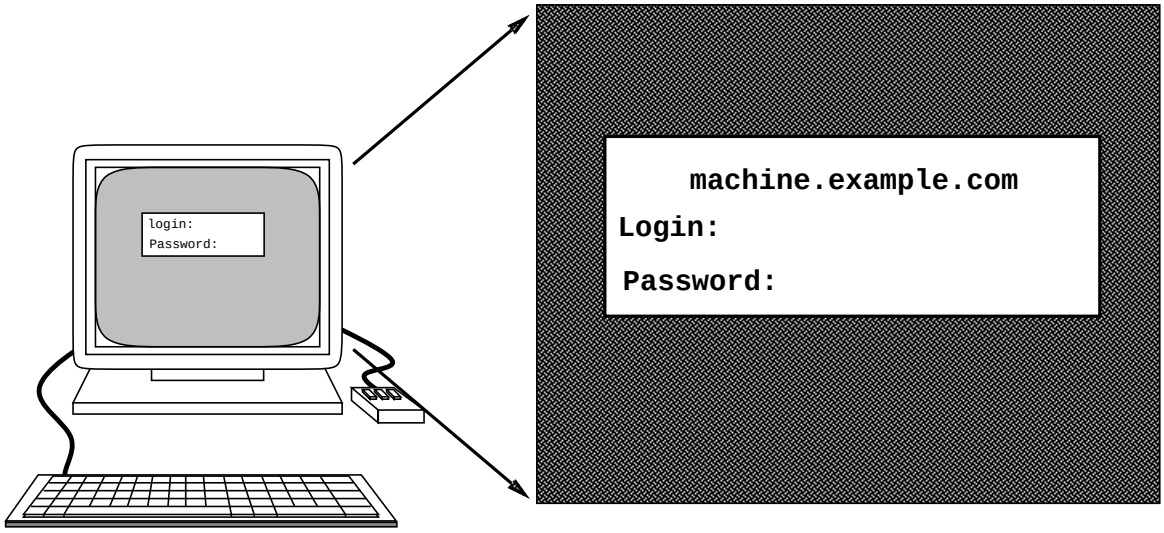
La demande du login et du mot de passe ressemble globalement à :



Chapitre 3 • Premiers contacts avec UNIX
§3.6 • Connexion sur un terminal graphique UNIX

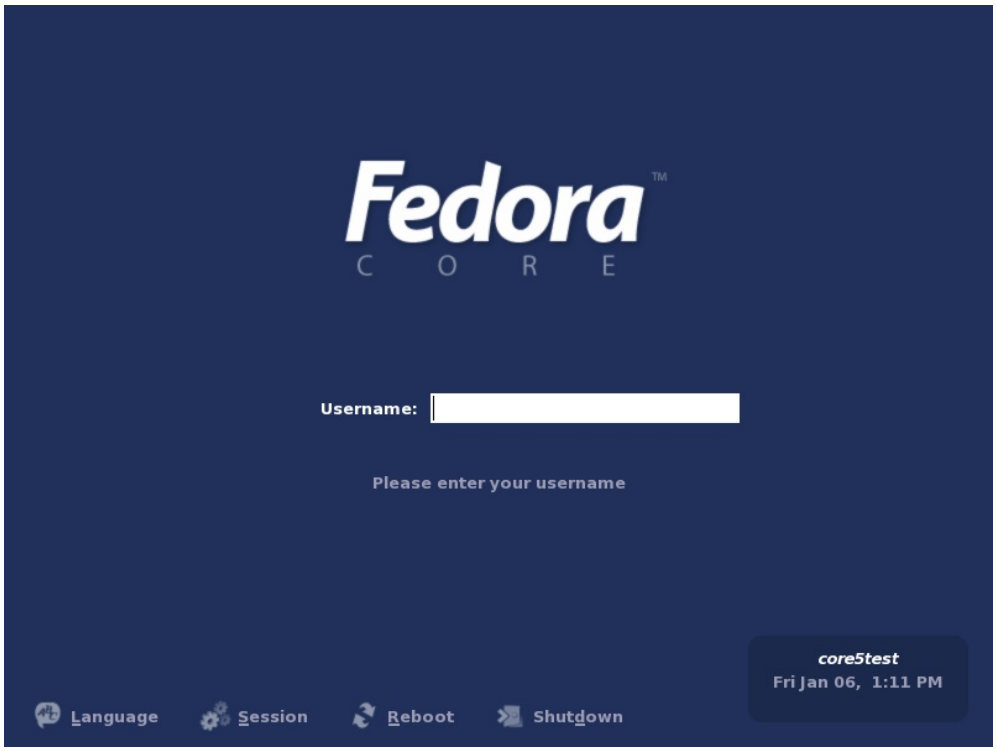
Station de travail UNIX (SUN Blade 100)

La demande du login et du mot de passe ressemble globalement à :

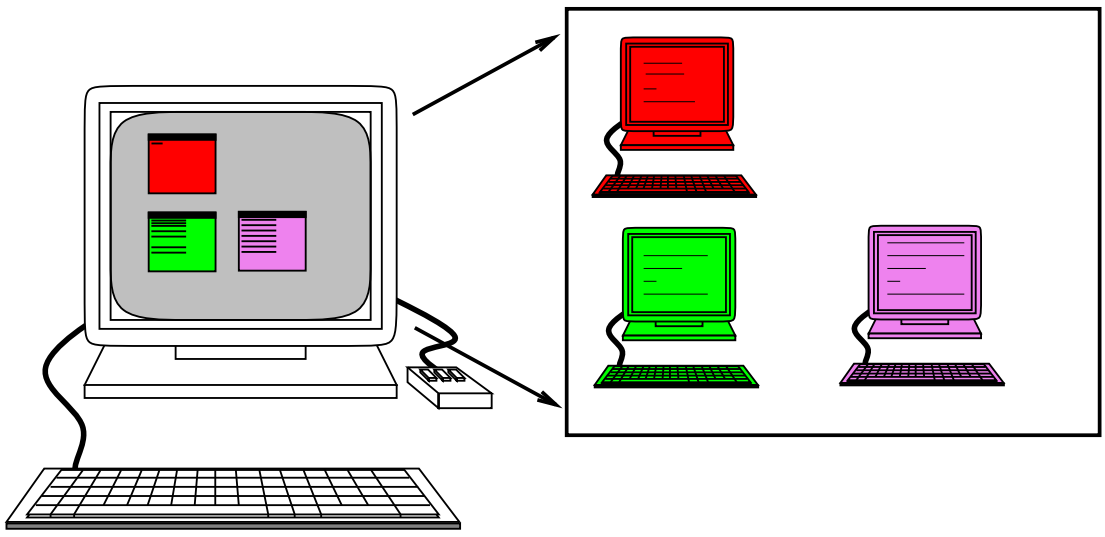


Par exemple :

Par exemple :



Une fois connecté via l'interface graphique, on utilisera principalement un programme d'émulation de terminal de type texte qui fournit dans une fenêtre une connexion comme sur un terminal texte :



L'émulateur de terminal s'appelle « `xterm` ».

A l'origine, des teletypes puis des consoles texte.
⇒ l'interaction de base se fait au moyen de phrases à taper sur un clavier (par opposition aux interfaces graphiques à la Windows ou de Macintosh).

A gauche, console DIGITAL VT100.
A droite, teletype DIGITAL.

Le shell est un programme qui permet la saisie et l'interprétation de ce qui est tapé. Le shell est juste une interface avec le système.

MS-DOS comporte un shell aux possibilités restreintes par rapport aux shells UNIX.

Le shell est aussi un vrai langage de programmation, interprété (non compilé) offrant les structures de base de programmation de tout autre langage.

Sous UNIX, le shell est un programme au même titre qu'un autre. Le shell de travail est **interchangeable** par un autre shell (à la syntaxe près comme de bien entendu).

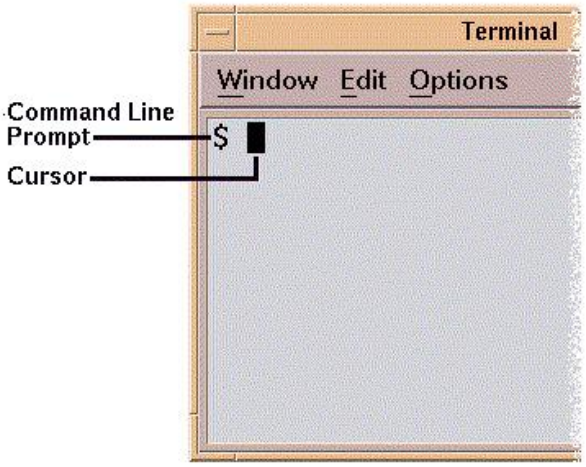
Les shells les plus répandus :

Shell	**Nom du programme**	**Description**
Bourne shell	sh	disponible sur toute plateforme UNIX
C shell	csh	shell développé pour BSD
Korn shell	ksh	Bourne shell amélioré par AT&T
Bourne again shell	bash	Shell distribué avec linux ; version améliorée de sh et csh

Dans ce cours, on distinguera le **shell de programmation** (car on peut programmer grâce à un interpréteur de commandes s'il est bien pensé) du **shell de travail** lors d'une session interactive. Les 2 shells n'ont pas de raison d'être identiques (cf plus loin sur ce que cela implique).

Tous les shells se présentent sous la même forme à l'écran lorsqu'ils fonctionnent :

- une chaîne de caractères affiche que le shell attend que l'utilisateur tape quelque chose au clavier ; c'est le **prompt**.
- un *curseur* qui va se déplacer au fur et à mesure de la saisie des commandes



Pour ce cours, on utilisera le caractère % pour désigner le prompt d'un utilisateur normal :

```
% commande-utilisateur
```

Pour ce cours, on utilisera le caractère # pour désigner le prompt de l'administrateur :

```
# commande-administrateur
```

Il ne faudra jamais taper la chaîne de prompt lorsque vous testerez par vous mêmes les commandes indiquées.

Pour terminer une session shell, on tape la commande commune à tous les shells :

```
% exit
```

Une commande UNIX ≡ un ensemble de mots séparés par des caractères blancs (caractère espace, tabulation)
Le premier mot : le nom de la commande
Le reste des mots : les paramètres de la commande

Particularités de certains mots : des options qui changent le comportement de la commande
En pratique on trouvera donc écrit :

```
commande [options] parametres
```

Les 2 crochets « `[` » et « `]` » indiquent que les options ne sont pas obligatoires. Il ne faut pas taper ces crochets sur la ligne de commande.

Dans ce cours, on met les 2 crochets « `[` » et « `]` » en couleur rouge et les options en vert (sauf oubli).

◇ Comment spécifie-t-on une option ?

Une option est quelque chose de prévu par le programme ⇒ c'est le programmeur qui aura toujours le dernier mot.
Il reste une tendance générale : Une option est introduite par le signe « – » et est souvent constituée d'une seule lettre comme par exemple « `-a` ».
(mais attention aux exceptions nombreuses)

Souvent on pourra cumuler et condenser des options :
`ls -a -l` ≡ `ls -al`

Souvent (mais pas tout le temps), l'ordre des options n'a pas d'importance. (cf `getopt(1)` ou `getopt(3)`)
`ls -a -l` ≡ `ls -al` ≡ `ls -la` ≡ `ls -l -a`

Ci joint dans la version imprimée de ce cours, une news USENET sur les différences entre les différents shells existants.

Path:
senator-bedfellow.mit.edu!bloom-beacon.mit.edu!newsfeed.internetmci.com!newsxfer3.itd.umich.edu!news-peer.sprintlink.net!news.
sprintlink.net!Sprint!cpk-news-hub1.bbnplanet.com!news.bbnplanet.com!europa.clark.net!dispatch.news.demon.net!demon!gryphon.de
mon.co.uk!news.looking-glass.org!news.looking-glass.org!not-for-mail
From: Brian Blackmore <bnb@gryphon.demon.co.uk>
Newsgroups: comp.unix.shell,comp.unix.questions,news.answers,comp.answers
Subject: UNIX shell differences and how to change your shell (Monthly Posting)
Supersedes: <shell_866063613@news.looking-glass.org>
Followup-To: comp.unix.shell
Date: 13 Jul 1997 17:41:35 +0100
Organization: None
Sender: bnb@looking-glass.org
Approved: news-answers-request@MIT.Edu
Expires: 24 Aug 1997 16:41:27 GMT
Message-ID: <shell_868812087@news.looking-glass.org>
Reply-To: shell-diff@looking-glass.org
NNTP-Posting-Host: gryphon.looking-glass.org
X-NNTP-Posting-Host: gryphon.demon.co.uk [158.152.42.76]
Lines: 369
Xref: senator-bedfellow.mit.edu comp.unix.shell:51716 comp.unix.questions:130975 news.answers:107262 comp.answers:27074

Archive-name: unix-faq/shell/shell-differences
Version: 1.17

The following article answers the frequently asked questions, what UNIX shells are available, what are the differences between them and how do you change your interactive shell. It is posted monthly to the USENET newsgroups comp.unix.shell, comp.unix.questions, news.answers and comp.answers and is additionally available on the world wide web as http://www.wonderland.org/~eternal/shell.html

Contents

Modifications since last issue

* Change of authors contact addresses

Why change your shell

The UNIX shell is most people’s main access to the UNIX operating system and as such any improvement to it can result in considerably more effective use of the system, and may even allow you to do things you couldn’t do before. The primary improvement most of the new generation shells give you is increased speed. They require fewer key strokes to get the same results due to their completion features, they give you more information (e.g. showing your directory in your prompt, showing which files it would complete) and they cover some of the more annoying features of UNIX, such as not going back up symbolic links to directories.

A brief history of UNIX shells

Note, this history is just known to be slightly out of historical order, it is in the process of being corrected, but for the moment should be taken with a pinch of salt

In the near beginning there was the Bourne shell /bin/sh (written by S. R. Bourne). It had (and still does) a very strong powerful syntactical language built into it, with all the features that are commonly considered to produce structured programs; it has particularly strong provisions for controlling input and output and in its expression matching facilities. But no matter how strong its input language is, it had one major drawback; it made nearly no concessions to the interactive user (the only real concession being the use of shell functions and these were only added later) and so there was a gap for something better.

Along came the people from UCB and the C-shell /bin/csh was born. Into this shell they put several concepts which were new, (the majority of these being job control and aliasing) and managed to produce a shell that was much better for interactive use. But as well as improving the shell for interactive use they also threw out the baby with the bath water and went for a different input language.

The theory behind the change was fairly good, the new input language was to resemble C, the language in which UNIX itself was written, but they made a complete mess of implementing it. Out went the good control of input and output and in came the bugs. The new shell was simply too buggy to produce robust shell scripts and so everybody stayed with the Bourne shell for that, but it was considerably better for interactive use so changed to the C shell, this resulted in the stupid situation where people use a different shell for interactive work than for non-interactive, a situation which a large number of people still find themselves in today.

After csh was let loose on an unsuspecting world various people decided that the bugs really should get fixed, and while they where at it they might as well add some extra features. In came command line editing, TENEX-style completion and several other features. Out went most of the bugs, but did the various UNIX operating system manufacturers start shipping tcsh instead of csh? No, most of them stuck with the standard C-Shell, adding non-standard features as they went along.

Eventually David Korn from AT&T had the bright idea to sort out this mess and the Korn shell /bin/ksh made its appearance. This quite sensibly junked the C shells language and reverted back to the bourne shell language, but it also added in the many features that made the C shell good for interactive work (you could say it was the best of both

worlds), on top of this, it also added a some features from other operating. The Korn shell became part of System V but had one major problem; unlike the rest of the UNIX shells it wasn't free, you had to pay AT&T for it.

It was at about this time that the first attempts to standardize UNIX started in the form of the POSIX standard. POSIX specified more or less the System V Bourne Shell (by this time the BSD and System V versions had got slightly different). Later the standard is upgraded, and somehow the new standard managed to look very much like ksh.

Also at about this time the GNU project was underway and they decided that they needed a free shell, they also decided that they wanted to make this new shell POSIX compatible, thus bash (the Bourne again shell) was born. Like the Korn shell bash was based upon the Bourne shells language and like the Korn shell, it also pinched features from the C shell and other operating systems (in my opinion it put them together better; guess which shell I use), but unlike the Korn shell it is free. Bash was quickly adopted for LINUX (where it can be configured to perform just like the Bourne shell), and is the most popular of the free new generation shells.

Meanwhile Tom Duff faced with the problem of porting the Bourne shell to Plan 9, revolts and writes rc instead, he publishes a paper on it, and Byron Rakitzis reimplements it under UNIX. With the benefit of a clean start Rc ended up smalled, simpler, more regular and in most peoples opinion a much cleaner shell.

The search for the perfect shell still goes on and the latest entry into this arena is zsh. Zsh was written by Paul Falstad while he was a student a Princeton and suffers from slight case of feeping creaturism. It is based roughly on the bourne shell (although there are some minor but important differences) and has so many additional features that I don't even think the author even knows all of them.

Additionally rc has been enhanced to produced es, this shell adds the ability for the user to redefine low level functions.

Deciding on a shell

Which of the many shells you choose depends on many different things, here is what I consider to be the most important, you may think differently.

How much time do I have to learn a new shell?
: There is no point in using a shell with a different syntax, or a completly different alias system if you havn't the time to learn it. If you have the time and are presently using csh or tcsh it is worth considering a switch to a Bourne shell variant.

What do I wish to be able to do with my new shell?
: The main reason for switching shells is to gain extra functionality; its vital you know what you are gaining from the switch.

Do I have to be able to switch back to a different shell?
: If you may have to switch back to a standard shell, it is fairly important you don't become too dependent on extra features and so can't use an older shell.

How much extra load can the system cope with?
: The more advanced shells tend to take up extra CPU, since they work in cbreak mode; if you are on an overloaded machine they should probably be avoided; this can also cause problems with an overloaded network. This only really applies to very old systems nowadays.

What support is given for my new shell?
: If your new shell is not supported make sure you have someone you can ask if you encounter problems or that you have the time to sort them out yourself.

What shell am I using already?
: Switching between certain shells of the same syntax is alot easier than switching between shells of a different syntax. So if you havn't much time a simple upgrade (eg csh to tcsh) may be a good idea.

Can I afford any minor bugs?
: Like most software all shells have some bugs in them (especially csh), can you afford the problems that may occur because of them.

Do you need to be able to use more than one shell?
: If you use more than one machine you may discover that you need to use more than one shell regularly. How different are these shells and can you cope with having to switch between these shells on a regular basis. It may be to your advantage to choose shells that are similar to each other.

Shell features

This table below lists most features that I think would make you choose one shell over another. It is not intended to be a definitive list and does not include every single possible feature for every single possible shell. A feature is only considered to be in a shell if in the version that comes with the operating system, or if it is available as compiled directly from the standard distribution. In particular the C shell specified below is that available on SUNOS 4.*, a considerable number of vendors now ship either tcsh or their own enhanced C shell instead (they don't always make it obvious that they are shipping tcsh.

	sh	csh	ksh	bash	tcsh	zsh	rc	es
Job control	N	Y	Y	Y	Y	Y	N	N
Aliases	N	Y	Y	Y	Y	Y	N	N
Shell functions	Y(1)	N	Y	Y	N	Y	Y	Y
"Sensible" Input/Output redirection	Y	N	Y	Y	N	Y	Y	Y

Directory stack	N	Y	Y	Y	Y	Y	F	F
Command history	N	Y	Y	Y	Y	Y	L	L
Command line editing	N	N	Y	Y	Y	Y	L	L
Vi Command line editing	N	N	Y	Y	Y(3)	Y	L	L
Emacs Command line editing	N	N	Y	Y	Y	Y	L	L
Rebindable Command line editing	N	N	N	Y	Y	Y	L	L
User name look up	N	Y	Y	Y	Y	Y	L	L
Login/Logout watching	N	N	N	N	Y	Y	F	F
Filename completion	N	Y(1)	Y	Y	Y	Y	L	L
Username completion	N	Y(2)	Y	Y	Y	Y	L	L
Hostname completion	N	Y(2)	Y	Y	Y	Y	L	L
History completion	N	N	N	Y	Y	Y	L	L
Fully programmable Completion	N	N	N	N	Y	Y	N	N
Mh Mailbox completion	N	N	N	N(4)	N(6)	N(6)	N	N
Co Processes	N	N	Y	N	N	Y	N	N
Builtin artithmetic evaluation	N	Y	Y	Y	Y	Y	N	N
Can follow symbolic links invisibly	N	N	Y	Y	Y	Y	N	N
Periodic command execution	N	N	N	N	Y	Y	N	N
Custom Prompt (easily)	N	N	Y	Y	Y	Y	Y	Y
Sun Keyboard Hack	N	N	N	N	N	Y	N	N
Spelling Correction	N	N	N	N	Y	Y	N	N
Process Substitution	N	N	N	Y(2)	N	Y	Y	Y
Underlying Syntax	sh	csh	sh	sh	csh	sh	rc	rc
Freely Available	N	N	N(5)	Y	Y	Y	Y	Y
Checks Mailbox	N	Y	Y	Y	Y	Y	F	F
Tty Sanity Checking	N	N	N	N	Y	Y	N	N
Can cope with large argument lists	Y	N	Y	Y	Y	Y	Y	Y
Has non−interactive startup file	N	Y	Y(7)	Y(7)	Y	Y	N	N
Has non−login startup file	N	Y	Y(7)	Y	Y	Y	N	N
Can avoid user startup files	N	Y	N	Y	N	Y	Y	Y
Can specify startup file	N	N	Y	Y	N	N	N	N
Low level command redefinition	N	N	N	N	N	N	N	Y
Has anonymous functions	N	N	N	N	N	N	Y	Y
List Variables	N	Y	Y	N	Y	Y	Y	Y
Full signal trap handling	Y	N	Y	Y	N	Y	Y	Y
File no clobber ability	N	Y	Y	Y	Y	Y	N	F
Local variables	N	N	Y	Y	N	Y	Y	Y
Lexically scoped variables	N	N	N	N	N	N	N	Y
Exceptions	N	N	N	N	N	N	N	Y

Key to the table above.

Y Feature can be done using this shell.

N Feature is not present in the shell.

F Feature can only be done by using the shells function mechanism.

L The readline library must be linked into the shell to enable this Feature.

Notes to the table above

1. This feature was not in the orginal version, but has since become almost standard.
2. This feature is fairly new and so is often not found on many versions of the shell, it is gradually making its way into standard distribution.
3. The Vi emulation of this shell is thought by many to be incomplete.
4. This feature is not standard but unoffical patches exist to perform this.
5. A version called ’pdksh’ is freely available, but does not have the full functionality of the AT&T version.
6. This can be done via the shells programmable completion mechanism.
7. Only by specifing a file via the ENV environment variable.

How to change your shell

If you ever look at a UNIX manual page it will say that to change your shell use chsh or passwd −s; unfortunately it often isn’t as simple as this, since it requires that your new shell is recognized as a valid shell by the system and at present many systems do not recognize the newer shells (the normal selection is, /bin/sh, /bin/csh and possibly /bin/ksh). You are thus left with having to do some sort of fudge, changing your effective login shell without changing your official entry in /etc/passwd. You may also be left with the problem that there isn’t a compiled binary on your system , so you will have to get hold of the shell’s source and compile it yourself (Its generally best to ask around to see if anyones done this already, since it isn’t that easy). Once done you should add in code to your old shells login file so that it overlays your official login shell with your new shell (remember to add the login flags to the command line, and with csh/tcsh ensure that the overlay doesn’t happen recursively since they both read the same .login file).

The shell can be recognized as a valid shell if the system administrator puts it in the file /etc/shells. If this file does not exist, it must be created and should contain all valid shells (i.e.don’t forget the traditional ones in all their forms).

WARNING

If you do decide to change your shell you must be very careful − if handled wrongly it can be almost impossible to correct, and will almost certainly cause you a lot of hassle. Never make a new shell a login shell until you have tested its new configuration files thoroughly and then tested them once again. It is also important that you make a full backup of your previous config files onto a floppy disk (or a different host if you have a second account) if you have to change any of them (which you will probably have to do if you can’t change your shell entry in /etc/passwd). You should also note that your new shell is probably not supported by your system admin, so if you have any problems you will probably have to look elsewhere.

Further information

The Bourne shell, the C−Shell and the Korn Shell (if you have it) are

all distributed as standard with your UNIX operating system, information on these should come with your operating system, bug reports etc should be sent to your operating system vendor. The free version of ksh, pdksh is available as ftp://ftp.cs.mun.ca/pub/pdksh/

A commercial "compiler" (CCsh) is available for the Bourne shell, this can provide extra speed on some applications. For more information contact Comeau Computing 91-34 120th Street, Richmond Hill, NY, USA 11418-3214. Email comeau@csanta.attmail.com.

Bash was written and is maintained by the Free Software Foundation, the primary source of information for this shell is its manual page. Bug reports should be sent to bash-maintainers@ai.MIT.Edu, while suggestions and philosophical bug reports may be mailed to bug-bash@ai.MIT.Edu or posted to the Usenet newsgroup gnu.bash.bug, the source is widely available on many ftp sites, and is subject to the GNU copyleft licence.

Tcsh is available by ftp from ftp://ftp.deshaw.com and several other places.

Rc is available by ftp from ftp://viz.tamu.edu/pub/rc, ftp://ftp.sys.utotonoto.cs/pub/rc and several other places. An FAQ exists and is posted frequently to comp.unix.shell and other places. The Rc mailing list may be subscribed to by sending mail to rc-request@hawkwind.utcs.toronto.edu, this, the manual page and the Rc FAQ are the main sources of information for this shell. The original paper on rc is available as http://plan9.att.com/plan9/plan9doc/7.ps.Z.

Zsh is available by ftp from ftp://ftp.math.gatech.edu/pub/zsh and at several mirror sites. Zsh is now maintained by the members of the zsh mailing lists, which can be subscribed to by sending email to zsh-announce-request@math.gatech.edu (announcements), zsh-users-request@math.gatech.edu (users discussions) or zsh-workers-request@math.gatech.edu (zsh hacking and development) with a subject of "subscribe email-address", there is also an FAQ which is posted frequently to comp.unix.shell. The manual page, the Z-shell FAQ and the zsh-list are the main sources of information for this shell.

ES is available by ftp from ftp://ftp.cs.utoronto/pub/es and at several other places, a mailing list can be subscribed to by sending mail to es-request@hawkwind.utcs.toronto.edu and a paper about es is at http://www.webcom.com/~haahr/es/es-usenix-winter93.html.

Questions on any of the UNIX shells and on shell script programming, may be posted to the Usenet newsgroup comp.unix.shell a quick response can normally be expected, especially on subjects relating to the more common shells.

Copyright and Disclaimer

Email relating to the content of this document should be sent to shell-diff@looking-glass.org

Brian Blackmore, bnb@looking-glass.org

Chapitre 4

Sources de documentation

Beaucoup de documentation disponible. **Il faut lire la documentation.**
Souvent en anglais.

Extrait d'un rapport d'un ancien élève de ARS :

Logiciel XXXXX : Documentation claire et exhaustive (j'avais le choix entre la version Anglaise ou Japonaise, la version Anglaise est très bien ! ! ! ! ! ! !)

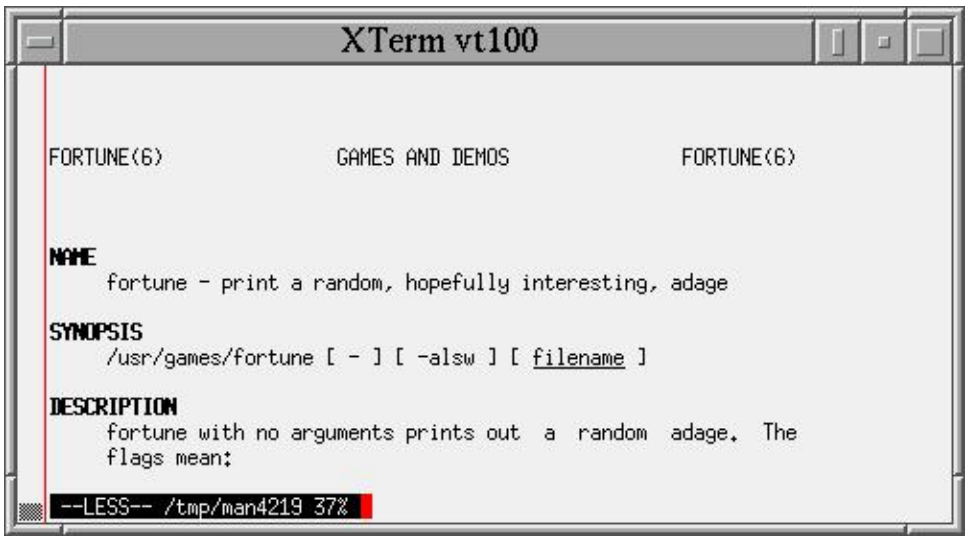
Il existe une documentation électronique accessible pendant le fonctionnement du système : c'est l'aide en ligne.

La commande donnant l'aide est « **man** ».

On peut faire « **man man** » !

◇ Structure de la documentation affichée

Exemple : « `man fortune` » renvoie :



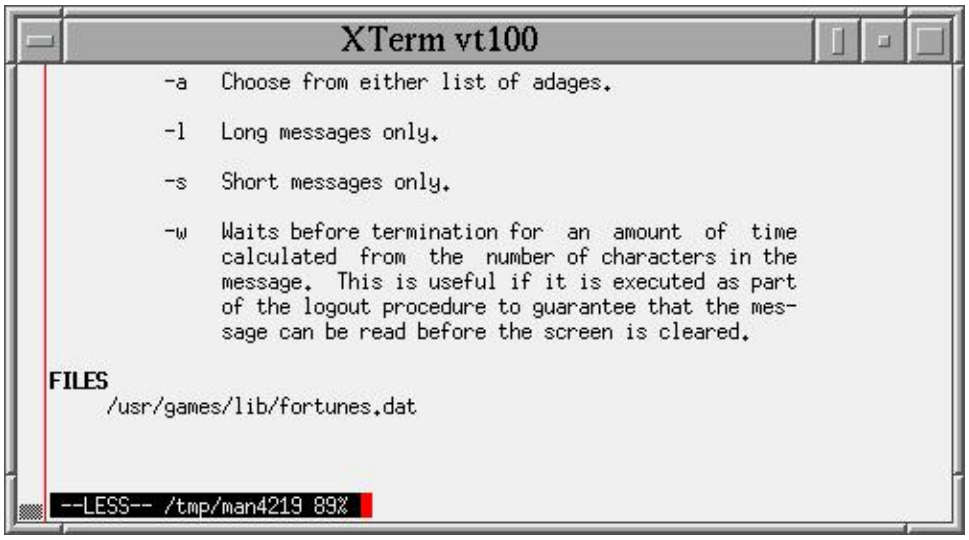
On remarque :

- affichage page d'écran par page d'écran pour mieux lire la doc
- plusieurs rubriques (NAME, SYNOPSIS, DESCRIPTION, ...)

◇ Lecture de la documentation affichée

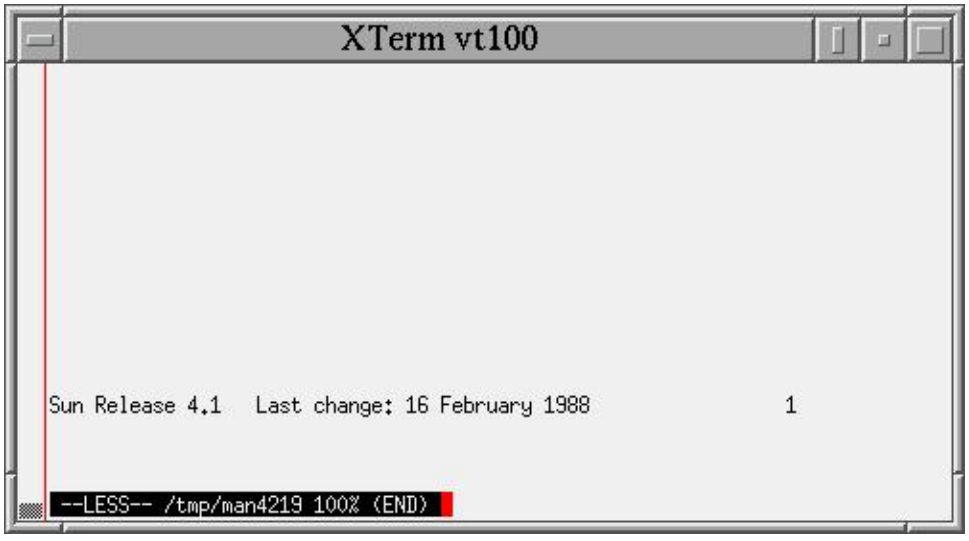
On navigue entre les pages d'écran de la documentation par :

- la touche SPC pour avancer (ou f ≡ *forward*)
- la touche b pour reculer (b ≡ *backward*)



On quitte :

- quand on arrive à la fin de la documentation
- prématurément par la touche q (q ≡ *quit*)



◇ Sections de la documentation

Les pages de manuel des commandes UNIX sont réparties en chapitres appelés des **sections**.

Liste classique des sections :

- section 1 = commandes normales
- section 2 = appels systèmes
- section 3 = fonctions de programmation C
- section 4 = périphériques et pilotes de périphériques
- section 5 = format de fichiers système
- section 6 = jeux
- section 7 = divers
- section 8 = commandes de gestion du système

◇ Convention sur les sections

Dans les livres/magazines sur UNIX/LINUX, on rencontre souvent l'écriture suivante : « `getopt(3)` »

Signification : cela fait référence à la commande « `getopt` » de la section **3** du manuel.

ATTENTION : ne pas confondre « `getopt(3)` » ayant le sens « se reporter à la documentation sur `getopt` dans la section 3 » avec « appel de la fonction C `getopt` avec le paramètre 3 ». Seul le contexte vous permettra de comprendre la bonne signification.

◇ Ordre de la recherche dans les sections

Le principe est : on affiche la documentation de la première section trouvée.

Cela pose problème quand il existe une commande de même nom dans plusieurs sections.

Par exemple :

- commande UNIX « `printf` » : section 1
- fonction de programmation C « `printf` » : section 3

Si vous faites « `man printf` », vous obtiendrez la documentation sur la commande UNIX « `printf` » (section 1 affichée avant section 3).

◇ Chercher dans une section précise sur LINUX

Sur LINUX, pour chercher dans une section précise, la syntaxe est :
man [numero-de-section] commande

Par exemple :

```
% man 1 printf
...

% man 3 printf
...
```

◇ Chercher dans une section précise sur SOLARIS

Sur SOLARIS, pour chercher dans une section précise, la syntaxe est :

```
man [-s numero-de-section] commande
```

Par exemple :

```
% man -s 1 printf
...

% man -s 3 printf
...
```


◇ Chapitres d'introduction

En général, chaque chapitre de la documentation a une introduction. L'introduction porte le nom « `intro` » et est plus ou moins développée selon les UNIX.

Pour avoir l'introduction sur la section 1, faire :

- LINUX : « `man 1 intro` »
- SOLARIS : « `man -s 1 intro` »

Pour avoir l'introduction sur la section 2, faire :

- LINUX : « `man 2 intro` »
- SOLARIS : « `man -s 2 intro` »

Pour avoir l'introduction sur la section 3, faire :

- LINUX : « `man 3 intro` »
- SOLARIS : « `man -s 3 intro` »

etc. (il suffit d'appliquer les informations du chapitre précédent)

◇ Défaut de la commande MAN

Inconvénient majeur : il faut connaitre le nom de la commande (nom anglais très souvent)

L'aide est plus là pour se rappeler les nombreuses options des commandes et leurs syntaxes particulières. Ce n'est pas un moteur de recherche à la GOOGLE.

Cependant il existe une possibilié de rechercher un mot clef apparaissant dans les pages de la documentation : syntaxe « `man -k mot-clef` ».

Par exemple :

```
% man -k more
...
bzless    bzmore (1)    - file perusal filter for crt viewing
of bzip2 compressed text
bzmore    bzmore (1)    - file perusal filter for crt viewing
of bzip2 compressed text
gzmore    gzmore (1)    - file perusal filter for crt viewing
of compressed text
less      less (1)      - opposite of more
more      more (1)      - browse or page through a text file
page      more (1)      - browse or page through a text file
...
```

Pour aller plus loin : la navigation dans la documentation MAN se fait par les mêmes raccourcis clavier que dans la commande « more » (voir page 230).

En fait, la commande « man » met en page la documentation et en confie l'affichage à la commande « more ».

A l'usage, on peut donc utiliser les mêmes raccourcis que ceux de « more ».

Par exemple, pour chercher le mot « ananas » dans la documentation, on fera « /ananas » pour chercher vers l'avant, on fera « ?ananas » pour chercher vers l'arrière, on fera « n » pour l'apparition suivante du mot.

RFC = *Request For Comments*
Documents de référence en anglais récupérables aux adresses :

- ftp://ftp.lip6.fr/pub/rfc/rfc/
- ftp://ftp.lip6.fr/pub/rfc/internet-drafts/
- http://abcdrfc.free.fr/

De nombreux autres sites existent.

(en anglais *Frequently Asked Questions*, en français *Foire Aux Questions*)

Documents en anglais récupérables aux adresses :

- `ftp://ftp.lip6.fr/pub/doc/faqs/`

De nombreux autres sites existent.

Documents en anglais récupérables aux adresses :

- `ftp://ftp.lip6.fr/pub/linux/french/docs/`

De nombreux autres sites existent.

Plusieurs guides de transition d'un système à un autre système UNIX sont disponibles :

- `http://www.unixporting.com/porting-guides.html`
- `http://www.unixguide.net/`
- etc.

Les newsgroups sont des forums de discussion sur internet.

Les thèmes en sont variés. Certains forums sont dans une langue autre que l'anglais.

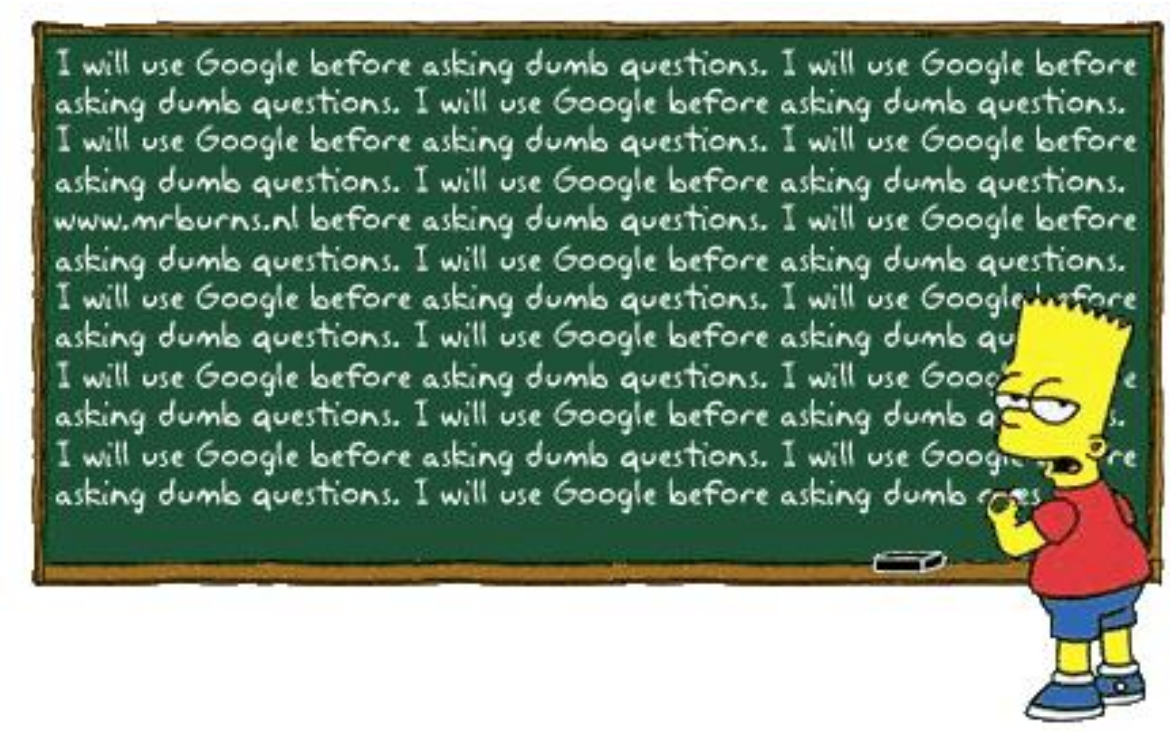
Sur jussieu, le serveur de news est « `news.jussieu.fr` »

Le protocole réseau des news s'appelle `NNTP`

Le site `http://www.google.com` offre un moteur de recherche très efficace.

De nombreux autres sites de moteur de recherche existent.

Humour :



Certains constructeurs UNIX mettent des documentations et documents online.
Se reporter par exemple à :

- `http://docs.sun.com/`
- `http://technet.microsoft.com/`
- `http://www.cisco.com/`
- etc.

De nombreux autres sites existent.

Tendance aux encyclopédies collaboratives à base de WIKI :

- `http://www.wikipedia.org` (décliné en plusieurs langues : `http://fr.wikipedia.org`, `http://en.wikipedia.org`, etc.)
- `http://www.commentcamarche.net`
- `http://www.dicodunet.com`
- `http://www.labo-cisco.com`
- `http://www.labo-microsoft.com`

De nombreux autres sites existent.

Certaines librairies ont un rayon informatique bien fourni :

- Le Monde en Tique
  6 rue Maître Albert, 75005 Paris
  `http://www.lmet.fr/`
- Eyrolles
  61 boulevard Saint Germain, 75005 Paris
  `http://www.eyrolles.fr/`
- Infothèque
  81 rue d'Amsterdam, 75008 Paris
  `http://www.infotheque.com/`

De nombreux magazines parlent de LINUX.

- vulgarisation de domaines anciennement réservés à un cercle d'initiés
- CDROM vendus avec ces magazines.
- prix abordables

Quelques magazines que j'apprécie :

- LINUX magazine france
- LINUX journal
- MISC

A vous de vous faire votre opinion. . .

Liste de quelques formats les plus répandus (en vrac)

◇ **format PDF** : extension « .pdf »

à lire avec :

- Acrobat Reader ; `http://www.adobe.com/products/acrobat/`
  plateforme : UNIX, WINDOWS, MacOS
- Ghostscript ; `http://www.ghostscript.com/`
  plateforme : UNIX, WINDOWS, MacOS
- xpdf ; `http://www.foolabs.com/xpdf/`
  plateforme : UNIX

De nombreux outils dérivés de Ghostscript existent.

◇ **format Postscript** : extension « .ps »

à lire avec :

- Ghostscript ; `http://www.ghostscript.com/`
  plateforme : UNIX, WINDOWS, MacOS
- ghostview ; `ftp://ftp.lip6.fr/pub/gnu/ghostview/`
  plateforme : UNIX, WINDOWS
- gv ; `http://wwwthep.physik.uni-mainz.de/~plass/gv/`
  plateforme : UNIX, WINDOWS

De nombreux outils dérivés de Ghostscript existent.

◇ **format Microsoft Word** : extensions « .doc », « .docx »

à lire avec :

- Microsoft Word ; `http://www.microsoft.com/office/word/`
  plateforme : WINDOWS, MacOS
- Microsoft Word viewer ; `http://www.microsoft.com/???/`
  plateforme : WINDOWS
- Open Office ; `http://www.openoffice.org/`
  plateforme : UNIX, WINDOWS, MacOS
- Google docs ; `http://docs.google.org/`
- antiword ; `http://www.antiword.org/`
  plateforme : UNIX

Peu d'outils sous UNIX en dehors de ceux-ci.

◇ **format Microsoft Excel** : extension « .xls »

à lire avec :

- Microsoft Excel ; `http://www.microsoft.com/office/word/`
  plateforme : WINDOWS, MacOS
- Microsoft Excel viewer ; `http://www.microsoft.com/???/`
  plateforme : WINDOWS
- Open Office ; `http://www.openoffice.org/`
  plateforme : UNIX, WINDOWS, MacOS
- Google docs ; `http://docs.google.org/`
- GNUmeric ; `http://www.gnome.org/projects/gnumeric/`
  plateforme : UNIX

Peu d'outils sous UNIX en dehors de ceux-ci.

◇ **format texte** : pas d'extension particulière

à lire avec n'importe quel éditeur de texte :

- vi ; standard
  plateforme : UNIX, WINDOWS, MacOS ?

- emacs ; `http://www.gnu.org/software/emacs/`
  plateforme : UNIX, WINDOWS, MacOS ?

Nombreux autres outils sous UNIX en dehors de ceux-ci.
Mais « vi » et « emacs » restent les meilleurs. Le reste est une plaisanterie ou une réinvention de « vi » ou « emacs ».

◇ **format HTML** : extension « .html » ou « .htm »

A lire avec n'importe quel navigateur web :

- Mozilla ; `http://www.mozilla.org/` ; Obsolète : voir Firefox
  plateforme : UNIX, WINDOWS, MacOS
- Firefox ; `http://www.mozilla.com/`
  plateforme : UNIX, WINDOWS, MacOS
- Opera ; `http://www.opera.com/`
  plateforme : UNIX, WINDOWS, MacOS
- Galeon ; intègré au bureau GNOME ;
  `http://galeon.sourceforge.net/`
  plateforme : UNIX
- Konqueror ; intégré au bureau KDE ; `http://www.konqueror.org/`
  plateforme : UNIX

D'autres navigateurs web existent.
A noter Nvu ; Editeur de pages HTML dérivé de Mozilla ;
`http://www.nvu.com/`
Remplé par KompoZer ; `http://kompozer.net/`
plateforme : UNIX, WINDOWS, MacOS

◇ **format graphique** : extension « .jpg » ou « .gif » ou « .png » ou autre

à lire avec :

- GIMP ; `http://www.gimp.org/`
  dessins bitmap à la Adobe Photoshop
  plateforme : UNIX, WINDOWS, MacOS
- XnView ;
  `http://perso.orange.fr/pierre.g/xnview/frhome.html`
  plateforme : UNIX, WINDOWS, MacOS
- Inkscape ; `http://www.inkscape.org/`
  dessins vectoriels à la Adobe Illustrator
  plateforme : UNIX, WINDOWS, MacOS
- Dia ; `http://www.gnome.org/projects/dia/`
  dessins à la Microsoft Visio
  plateforme : UNIX, WINDOWS

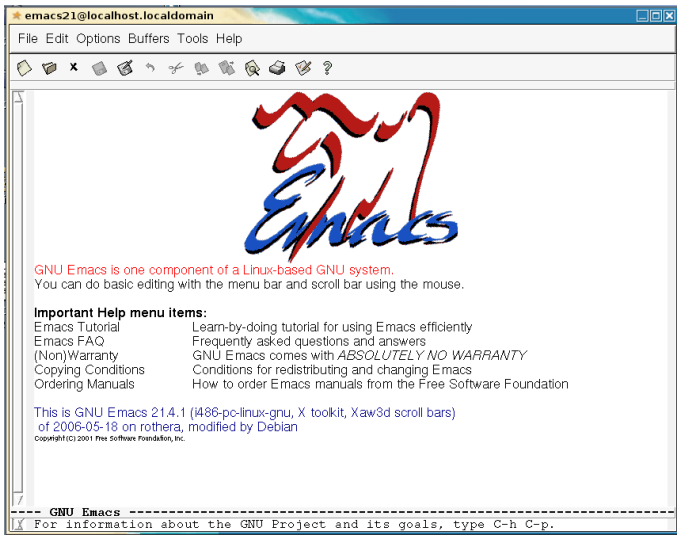
De nombreux outils graphiques existent.

Chapitre 5

Editeurs de texte UNIX

Il existe beaucoup d'éditeurs de texte sous UNIX mais seuls quelques uns sont suffisamment robustes pour être utilisés efficacement et avec confiance :

- « `vi` » : seul éditeur de texte standard sous UNIX
- « `emacs` » : très puissant, complexe à maîtriser, simple une fois qu'on sait s'en servir. Cf `http://www.emacs.org`.



(en anglais *visual interface*)

C'est l'éditeur de texte standard sur UNIX. Il fonctionne sur tout type de terminal texte, sur tout UNIX.

Inconvénient :

- il demande de la pratique

Il posséde deux modes de fonctionnement :

- un mode de saisie de commandes à appliquer au texte
- un mode de saisie du texte

Cf `http://www.math.fu-berlin.de/~guckes/vi/` pour de la doc.

Syntaxe : **vi [options] fichiers**

Options intéressantes :

- « `-r` » : recover
- « `-R` » : readonly
- « `+/ananas` » : ouverture du fichier et positionnement sur le mot « `ananas` »

Si le fichier indiqué n'existe pas, le fichier est créé.

◇ Commande passant en mode saisie de texte

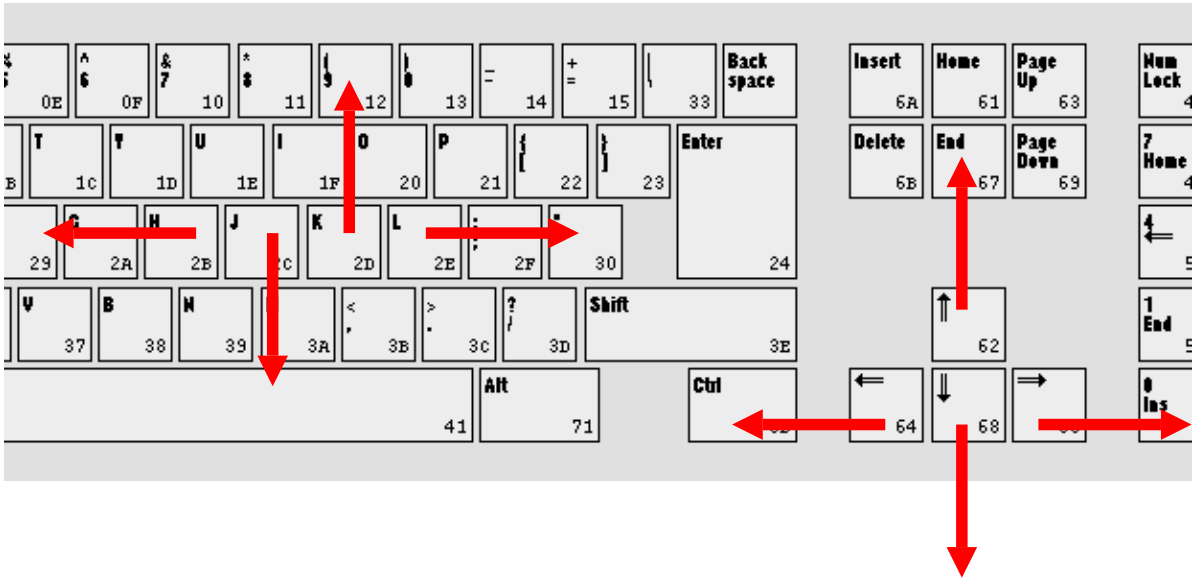
Séquence	Action
i	Insérer à la position courante du curseur (en anglais *insert*)
a	Insérer à la position suivante du curseur (en anglais *append*)
I	Insérer en début de ligne
A	Insérer en fin de ligne
o	Ouvrir une nouvelle ligne en dessous du curseur (en anglais *open*)
O	Ouvrir une nouvelle ligne au dessus du curseur
cw	Changer un mot (en anglais *change word*)
c$	Changer jusqu'à la fin de ligne

◇ Sortie du mode saisie du texte et passage en mode commandes

On passe du mode saisie de texte au mode commandes par la touche ESC.

◇ Commandes de déplacement

Séquence	Action
h ou ⇐	Déplacer le curseur d'un caractère à gauche
l ou ⇒	Déplacer le curseur d'un caractère à droite
j ou ⇓	Déplacer le curseur d'une ligne vers le bas
k ou ⇑	Déplacer le curseur d'une ligne vers le haut



◇ Commandes de déplacement

Séquence	Action
nombre `G`	Aller à la ligne «nombre» (en anglais *goto*)
`G`	Aller à la dernière ligne
`Ctrl-F`	Avance d'une page d'écran (en anglais *forward*)
`Ctrl-B`	Recule d'une page d'écran (en anglais *backward*)
`Ctrl-G`	Affiche le numéro de la ligne courante

D'autres commandes de déplacement existent...

◇ Commandes principales

Séquence	Action
`x` `8x`	Détruire le caractère sous le curseur Détruire 8 caractères
`r` suivi d'un caractère `X`	Remplacer le caractère sous le curseur par le caractère X (en anglais *replace*)

`dd`
`d8d`
`:3,7d`
`:1,$d`
`:.,21d`
	Effacer la ligne courante
	Effacer 8 lignes en comptant la ligne courante
	Effacer de la ligne 3 à la ligne 7
	Effacer de la ligne 1 à la dernière ligne
	Effacer de la ligne courante à la ligne 21

`dw` `d8w`	Effacer le mot sous le curseur (en anglais *delete word*) Effacer 8 mots
`J`	Joindre la ligne suivante avec la ligne courante (en anglais *join*)

Remplacer 8 dans les exemples ci-dessus par le nombre que vous voulez.

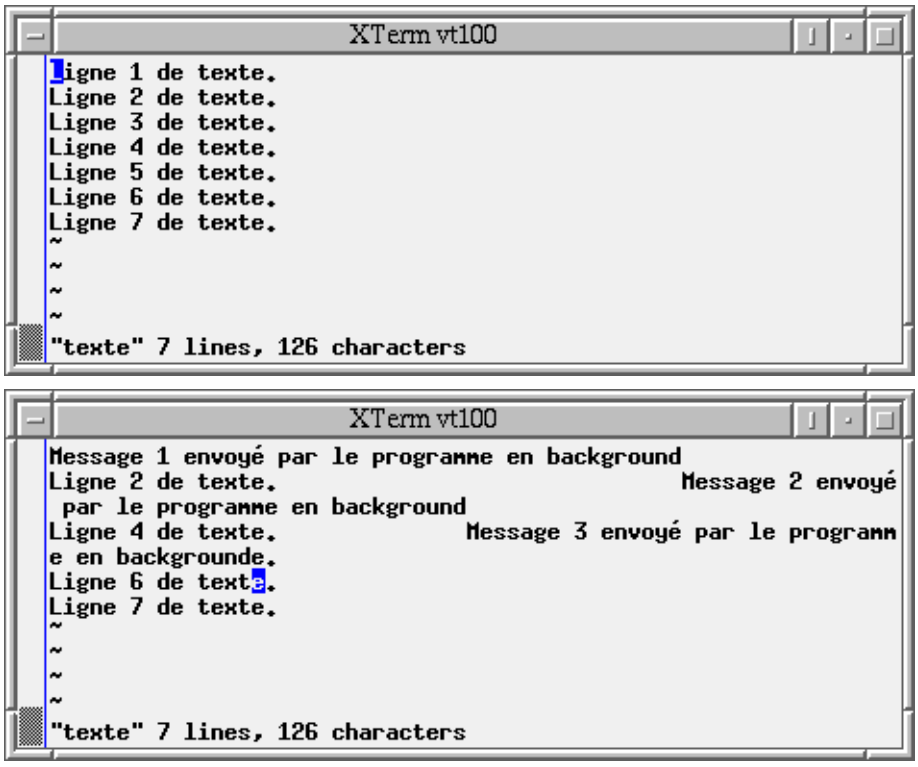
◇ Commandes principales (2)

Séquence	Action
`u`	Annuler la dernière commande (en anglais *undo*)
`Ctrl-L`	Rafraichir l'écran
`.`	Répéter la dernière commande
`/cerise`	Rechercher « cerise » dans le texte vers le bas
`?cerise`	Rechercher « cerise » dans le texte vers le haut
`/regexp`	Rechercher la regexp indiquée dans le texte vers le bas (voir page 381)
`?regexp`	Rechercher la regexp indiquée dans le texte vers le haut (voir page 381)
`n`	Répéter la dernière recherche (en anglais *next*)
`N`	Répéter la dernière recherche dans l'autre sens (en anglais *next*)

A NOTER : La commande « `.` » recommence la dernière commande qui ne commençait pas par « `:` ».

Exemple d'utilisation du `Ctrl-L` :

Supprimer les affichages parasites par exemple de programmes en tâche de fond (voir page 523).



◇ Sauvegarde / Sortie de vi

Séquence	Action
`:w`	Sauver le fichier édité (en anglais *write*)
`:w ananas`	Sauver dans le fichier « `ananas` »
`:w! ananas`	Sauver dans le fichier « `ananas` » en écrasant le contenu actuel de « `ananas` »
`:q`	Quitter vi (en anglais *quit*)
`:q!`	Quitter vi sans sauvegarder la moindre chose
`:wq`	Sauver puis quitter vi (en anglais *write* + *quit*)
`:e ananas`	Editer maintenant le fichier « `ananas` » (en anglais *edit*)
`:e! ananas`	Editer maintenant le fichier « `ananas` » en oubliant les modifications du fichier actuel
`:r ananas`	Importer le contenu du fichier « `ananas` » (en anglais *read*)

◇ Commandes de copier/coller

Séquence	Action
`yy`	Copier la ligne courante dans la mémoire copier/coller (en anglais *yank*)
`y8y`	Copier 8 lignes en comptant la ligne courante dans la mémoire copier/coller
`p` (lettre minuscule)	Coller en dessous de la ligne du curseur le contenu de la mémoire précédente (en anglais *paste*)
`P` (lettre majuscule)	Coller au dessus de la ligne du curseur le contenu de la mémoire précédente (en anglais *paste*)

◇ Commandes de substitution

Séquence	Action
`:s/ananas/cerise/`	Sur la ligne du curseur, remplacer le premier mot « `ananas` » par « `cerise` » (en anglais *substitute*)
`:s/ananas/cerise/g`	Sur la ligne du curseur, remplacer tous les mots « `ananas` » par « `cerise` »
`:1,$s/ananas/cerise/`	De la ligne 1 à la dernière ligne ($), remplacer le premier mot « `ananas` » par « `cerise` »
`:1,$s/ananas/cerise/g`	De la ligne 1 à la dernière ligne ($), remplacer tous les mots « `ananas` » par « `cerise` »

Autres exemples de séquences de substitution :

- La séquence « `:1,$s/ananas/cerise/g` » remplace de la première ligne à la dernière ligne chaque mot « `ananas` » par « `cerise` ».
- La séquence « `:%s/ananas/cerise/g` » remplace de la première ligne à la dernière ligne chaque mot « `ananas` » par « `cerise` ».
  ⇒ **On peut employer `%` à la place de `1,$`.**
- La séquence « `:1,$s/ananas//g` » remplace de la première ligne à la dernière ligne chaque mot « `ananas` » par rien du tout, c'est-à-dire que l'on supprime de la première ligne à la dernière ligne chaque mot « `ananas` »
- La séquence « `:1,$s/\/ananas/cerise/g` » remplace de la première ligne à la dernière ligne chaque mot « `/ananas` » par « `cerise` ».
- La séquence « `:1,$s;/ananas;cerise;g` » remplace de la première ligne à la dernière ligne chaque mot « `/ananas` » par « `cerise` ».
  ⇒ **On peut employer d'autres caractères de séparation que le caractère « `/` ».**

Exemples avec divers caractères de séparation :

```
:1,$s/ananas/cerise/g
```

```
:1,$s/\/ananas/\/cerise/g
```

```
:1,$s;/ananas;/cerise;g
```

```
:1,$s,/ananas,/cerise,g
```

```
:1,$s|/ananas|/cerise|g
```


- séquence « `:1,.s/ananas/cerise/g` » remplace de la première ligneà la ligne courante (désignée par « **.** ») chaque mot « `ananas` » par « `cerise` ».
- La séquence « `:.,$s/ananas/cerise/g` » remplace de la ligne courante (désignée par « **.** ») jusqu'à la dernière ligne (désignée par « **$** ») chaque mot « `ananas` » par « `cerise` ».
- La séquence « `:.,.+3s/ananas/cerise/g` » remplace de la ligne courante (désignée par « **.** ») à 3 lignes plus bas (désignée par **.+3**) chaque mot « `ananas` » par « `cerise` ».
- La séquence « `:.-3,.s/ananas/cerise/g` » remplace de 3 lignes plus haut que la ligne courante (« **.-3** ») à la ligne courante chaque mot « `ananas` » par « `cerise` ».

◇ Principales options

Séquence	Action
`:set all`	Afficher toutes les options possibles
`:set option`	Positionner l'option « option » à vrai
`:set nooption`	Positionner l'option « option » à faux

ATTENTION : si vous positionnez une option ainsi, le réglage disparait à la fin de la session VI. Voir page 109 pour savoir comment enregistrer le réglage de façon permanente.

Options sur SOLARIS 10 :

```
:set all
noautoindent              nomodelines                    nosh
autoprint                 nonumber                       nosl
noautowrite               nonovice                       tabs
nobeautify                nooptimize                     tagl
directory=/var/tmp        paragraphs=IPLPPPQPP LIpplpipnpptags
noedcompatible            prompt                         tags
noerrorbells              noreadonly                     term
noexrc                    redraw                         note
flash                     remap                          time
hardtabs=8                report=2                       ttyt
noignorecase              scroll=37                      warn
nolisp                    sections=NHSHH HUuhsh+c        wind
nolist                    shell=/bin/csh                 wrap
magic                     shiftwidth=8                   wrap
mesg                      showmatch                      nowr
```

Options intéressantes :

- option « `nu` » : affichage des numéros de ligne
- option « `showmatch` » : affichage de la parenthèse, accolade ou crochet ouvrant
- option « `list` » : affichage des caractères non ASCII
- option « `autoindent` » : indentation automatique des lignes (voir programmation en langage C)

Les options peuvent être enregistrées de façon permanente.

Il faut les copier dans le fichier « `$HOME/.exrc` ».

Par exemple :

```
% cat $HOME/.exrc
set nu
```

◇ A propos des commandes qui commencent par « : »

Les commandes commençant par le caractère « : » « apparaissent » en bas de l'écran pour pouvoir lire ce que l'on tape (par exemple un nom de fichier pour sauvegarder).

◇ Divers

En cas de plantage de `vi`, utiliser la commande « `vi -r exemple.txt` » pour essayer de récupérer ce qui est récupérable (en anglais *recover*).

Pour consulter un fichier sans le modifier, faire « `vi -R exemple.txt` » (en anglais *readonly*).

Ne pas confondre les deux.

ATTENTION :

La version de « vi » dans les salles de TP de la Formation Permanente fait automatiquement une sauvegarde du fichier texte que l'on veut éditer. La sauvegarde automatique du fichier « exemple.txt » a pour nom « exemple.txt~ ».

(en anglais *vi improved*)

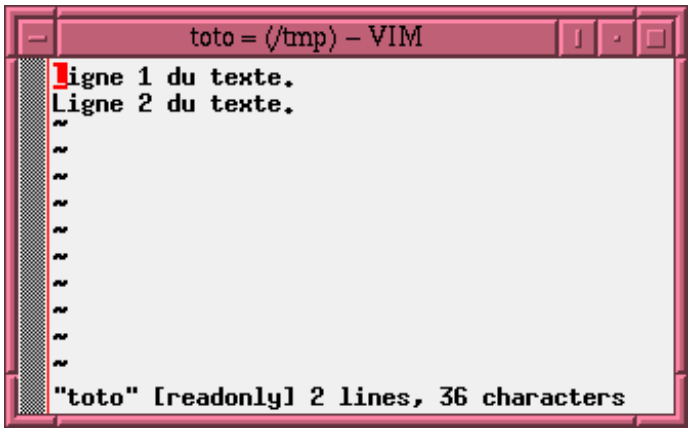
C'est une version améliorée de VI.

Cf http://www.vim.org

Intérêt : disponible sur LINUX, WINDOWS, MACOS

Attention à ne pas vous habituer à des fonctionnalités propres à VIM et non standard dans les autres VI.

La commande « view » lance « vi » en mode readonly.



On a donc : « view exemple.txt » équivalent à « vi -R exemple.txt ».

Commande très pratique.

Comment est codée une fin de ligne dans un fichier texte ?

Pas de standard !

En pratique :

- Sur UNIX : une fin de ligne est codée par le code ASCII 10 (Ctrl-J ; notation du langage C « \n »)
- Sur WINDOWS : une fin de ligne est codée par le code ASCII 13 suivi du code ASCII 10 (Ctrl-M Ctrl-J ; notation du langage C « \r\n »)
- Sur MACINTOSH : une fin de ligne est codée par le code ASCII 13 (Ctrl-M ; notation du langage C « \r »)

Nécessité de savoir convertir.

◇ Rappel sur la base 8 (base octale)

Voir cours de programmation en langage C.
Rappel : un nombre en base 8 (base octale) s'écrit avec un 0 initial.

Nombre en base 10	Nombre en base 8	Code ASCII équivalent
1	01	CTRL-A
2	02	CTRL-B
3	03	CTRL-C
4	04	CTRL-D
5	05	CTRL-E
6	06	CTRL-F
7	07	CTRL-G
8	010	CTRL-H
9	011	CTRL-I
10	012	CTRL-J
11	013	CTRL-K
12	014	CTRL-L
13	015	CTRL-M

Nombre en base 10	Nombre en base 8	Code ASCII équivalent
14	016	CTRL-N
15	017	CTRL-O
16	020	CTRL-P
17	021	CTRL-Q
18	022	CTRL-R
19	023	CTRL-S
20	024	CTRL-T
21	025	CTRL-U
22	026	CTRL-V
23	027	CTRL-W
24	030	CTRL-X
25	031	CTRL-Y
26	032	CTRL-Z

◇ Echanges entre UNIX et WINDOWS

Rappel :

- Sur UNIX : une fin de ligne est codée par le code ASCII 10 (Ctrl-J ; notation du langage C « `\n` »)
- Sur WINDOWS : une fin de ligne est codée par le code ASCII 13 suivi du code ASCII 10 (Ctrl-M Ctrl-J ; notation du langage C « `\r\n` »)

⇒ Passage de WINDOWS à UNIX facile car on passe de deux caractères de fin de ligne à un seul (supprimer est toujours plus facile qu'ajouter).
⇒ Passage de UNIX à WINDOWS difficile car on passe d'un caractère de fin de ligne à deux caractères de fin de ligne.

- WINDOWS vers UNIX (voir « `tr` » page 271) :
  « `tr -d '\015' < windows.txt > unix.txt` »
- UNIX vers WINDOWS (voir « `sed` » page 406) :
  « `sed -e 's/$/^M/' < unix.txt > windows.txt` »

◇ Echanges entre UNIX et MACINTOSH

Rappel :

- Sur UNIX : une fin de ligne est codée par le code ASCII 10 (Ctrl-J ; notation du langage C « `\n` »)
- Sur MACINTOSH : une fin de ligne est codée par le code ASCII 13 (Ctrl-M ; notation du langage C « `\r` »)

⇒ Passage de UNIX à MACINTOSH facile car dans les deux cas un seul caractère de fin de ligne !

- UNIX vers MACINTOSH (voir « `tr` » page 271) :
  « `tr '\012' '\015' < unix.txt > mac.txt` »
- MACINTOSH vers UNIX (voir « `tr` » page 271) :
  « `tr '\015' '\012' < mac.txt > unix.txt` »

Chapitre 6

Commandes de manipulation de base d'objets UNIX

Sur UNIX, plusieurs types d'objets :

- fichiers
- répertoires
- objets associés aux disques durs, CDROMs, bandes magnétiques, etc.
- objets système de communication inter applications
- autres objets systèmes (doors Solaris, etc.)

Ceux que l'on manipule le plus souvent en tant qu'utilisateur :

- fichiers
- répertoires

Sur UNIX, on fait la différence entre lettres minuscules et lettres majuscules en ce qui concerne les noms d'objets !

```
% ls -l
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 EXEMPLE.txt
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 ExEmPlE.txt
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 Exemple.txt
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 eXeMpLe.TXT
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 exemple.TXT
-rw-r--r--  1 besancon ars     0 Oct 16 21:44 exemple.txt
```

6 fichiers différents !

Sur UNIX, on évitera autant que possible les caractères espace, apostrophe, guillemets et les lettres accentuées dans les noms d'objets !

Sur UNIX, on préférera nommer les objets avec les lettres minuscules a-z, lettres majuscules A-Z, les chiffres 0-9, le tiret « - », le underscore « _ », le point « . ».

NE PAS UTILISER LE RESTE !

Sur UNIX, il y a certaines conventions pour les extensions dans les noms des objets.

- Extensions pour les langages de programmation : « `programme.c` », « `include.h` »,
- Extensions pour les archives ou les fichiers compressés : « `archive.tar` », « `rapport.gz` »
- pas d'extension pour les fichiers texte
  Dans ce cours, on clarifiera les choses en utilisant l'extension « `.txt` » du monde Windows quand cela sera plus parlant.
- pas d'extension pour les fichiers exécutables
  Dans ce cours, on clarifiera les choses en utilisant l'extension « `.exe` » du monde Windows quand cela sera plus parlant.
- etc.

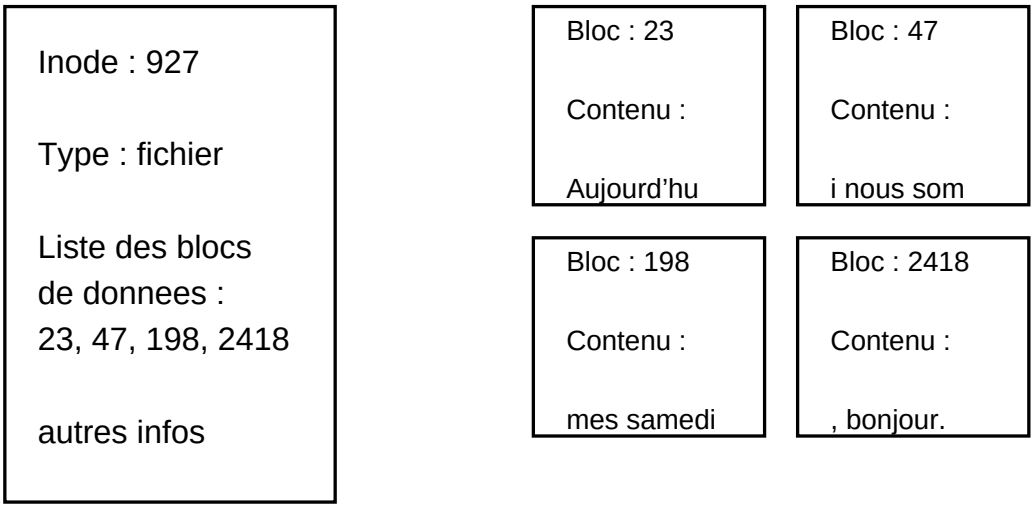
Les objets sont manipulables sur le disque dur via l'intermédiaire d'une structure de données appelée « **inode** ».
Cela sera revu en détails dans le tôme 2.

En gros :

1. un inode a un numéro unique
2. un inode indique le type de l'objet
3. un inode posséde la liste des blocs de données de l'objet
4. le système UNIX passe son temps à manipuler les inodes

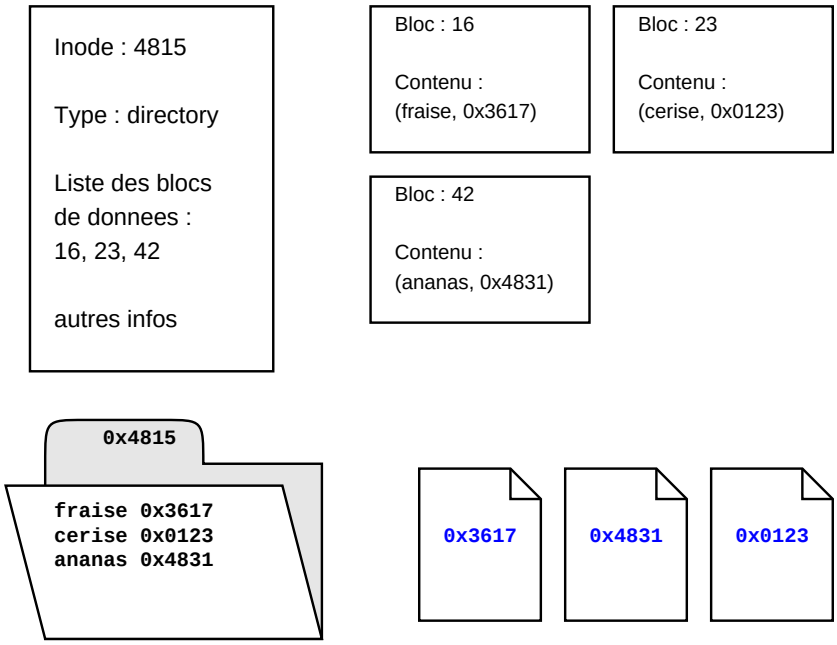
NOTA BENE : l'inode d'un objet ne stocke pas le nom de l'objet !

Un fichier correspond à un inode de type fichier :

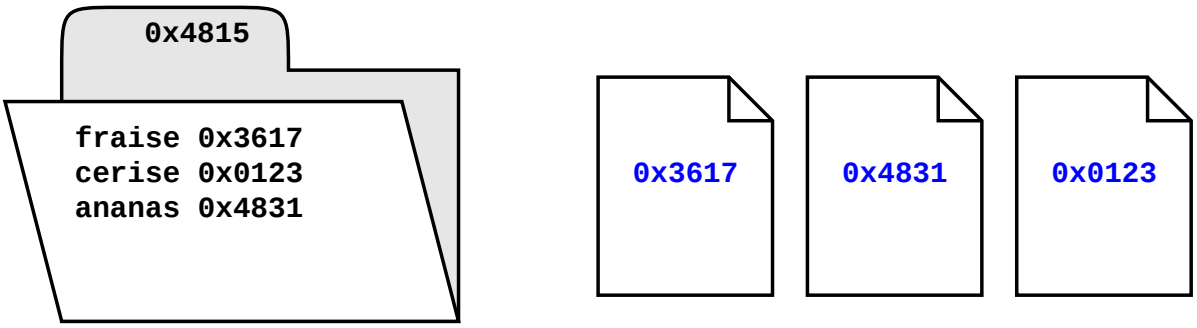


Terminologie : répertoire, dossier, *directory* en anglais

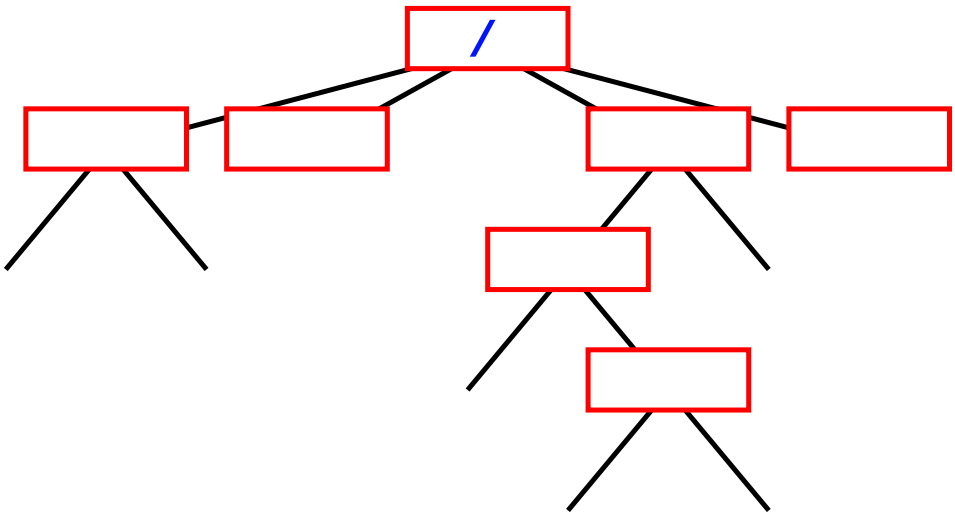
Un répertoire est un « fichier » dont les données sont une liste de noms + numéros d'inode.



C'est le répertoire qui donne un nom à un objet :

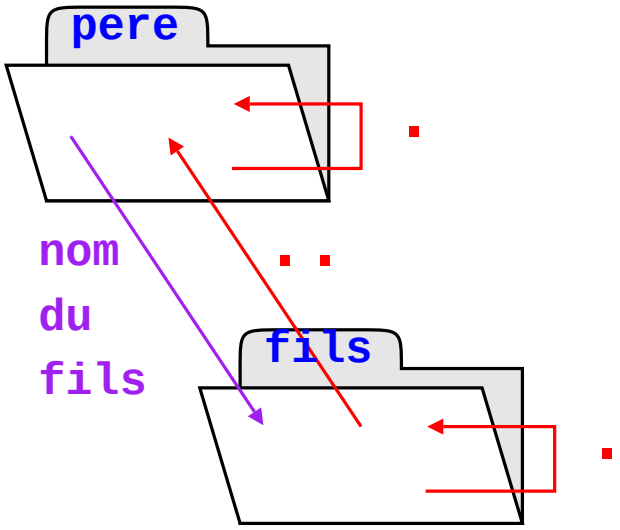


Un répertoire peut renvoyer sur un autre répertoire et ainsi de suite.
Cela permet de construire une arborescence représentable par un arbre :



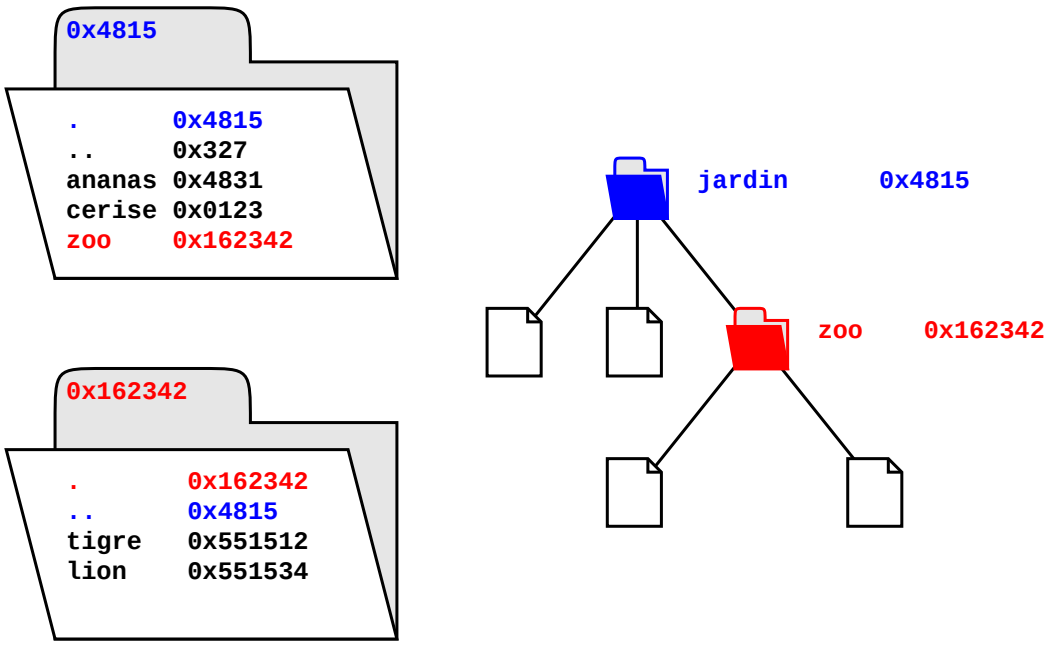
Les objets seront répartis dans l'arborescence.
La racine s'appelle « / », prononcé *slash*.

Plus exactement les directories sont organisés entre eux de la façon suivante :

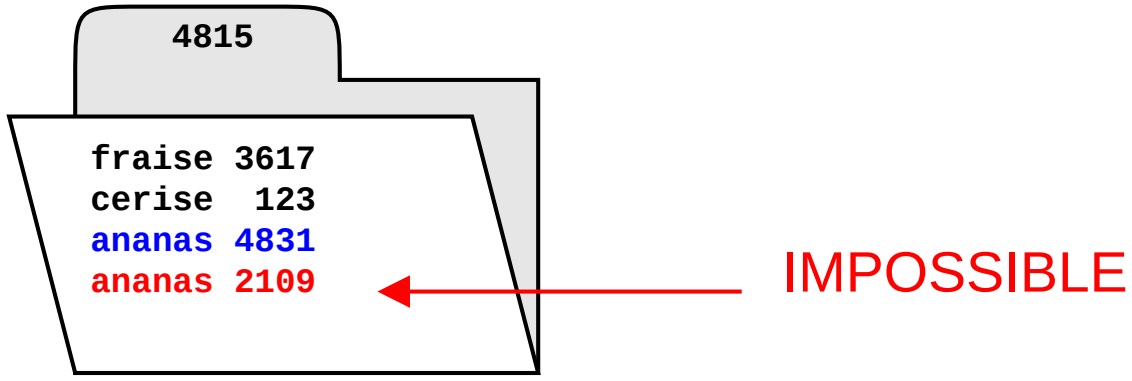


UNIX garantit qu'il n'y a pas de boucle dans l'arborescence.

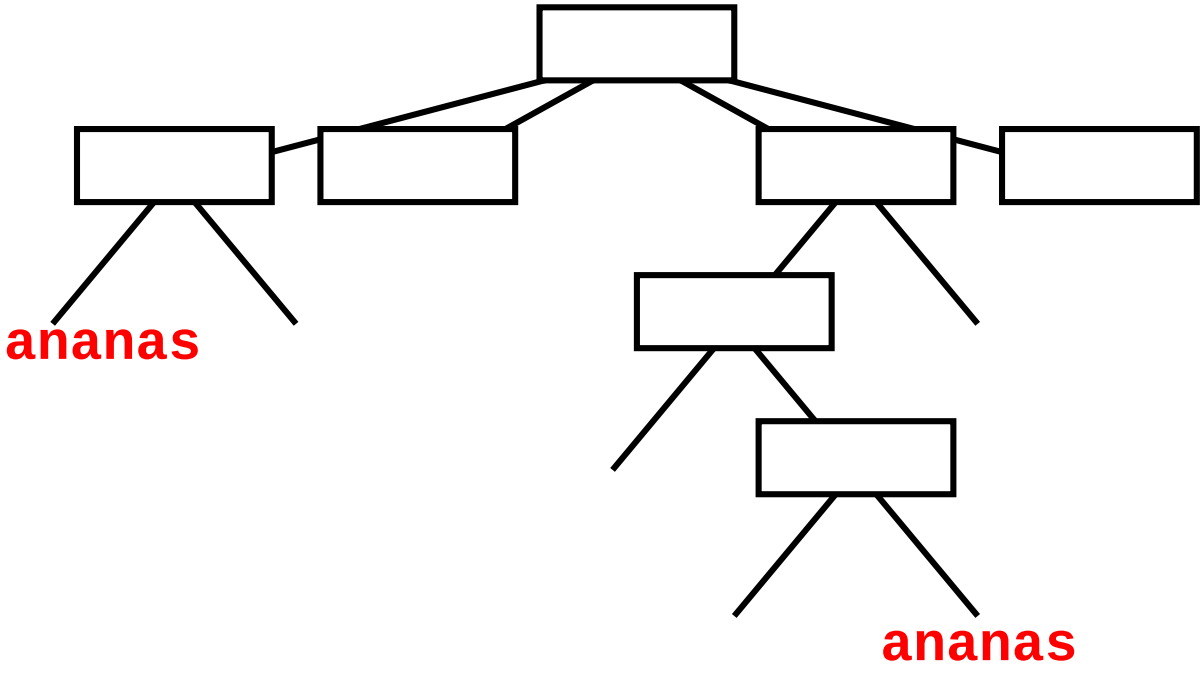
Exemple :



ATTENTION : un dossier ne peut pas contenir deux fois le même nom d'objet !



Par contre, on peut avoir plusieurs objets différents ayant le même nom dans l'arborescence :



Sur UNIX, un objet est manipulable par son chemin dans l'arborescence depuis le point de départ de l'arborescence.
Le chemin est constitué de la liste des noms des répertoires traversés et est terminé par le nom de l'objet en soi.

/ répertoire1 / répertoire 2/ ... / nom

Point fondamental : utilisation du caractère « / » comme séparateur dans l'énumération des répertoires traversés.

Attention : par abus de langage, on confondra l'objet avec le nom de l'objet et avec le chemin d'accès à l'objet.

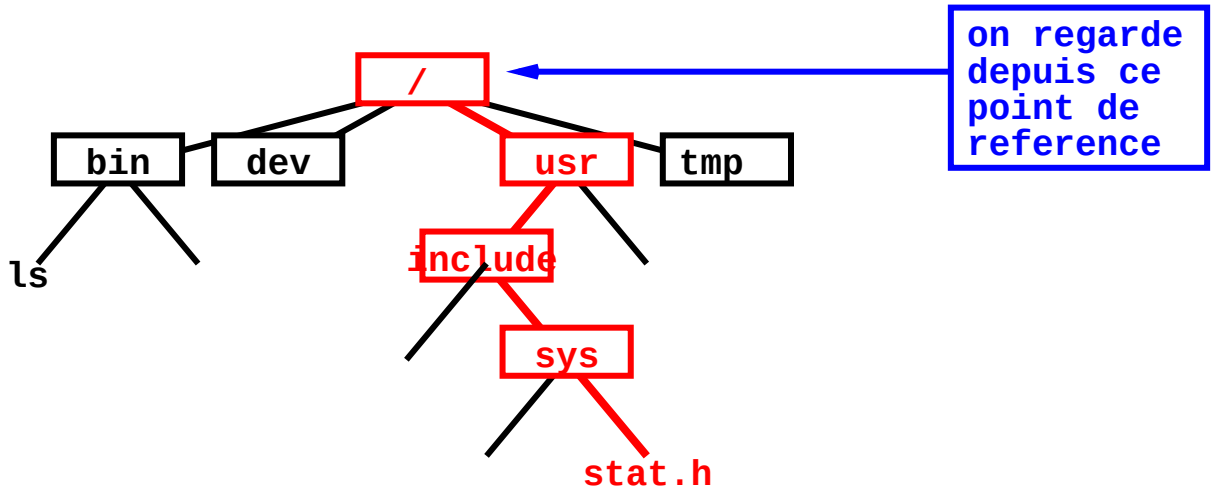
◇ **chemin d'accès absolu** :

Si le chemin d'accès commence par « / », il s'agit d'un chemin absolu :

chemin d'accès absolu = / répertoire1 / répertoire2 / ... / nom

Un chemin absolu s'exprime par rapport à la racine « / ».

Par exemple : « /usr/include/sys/stat.h »



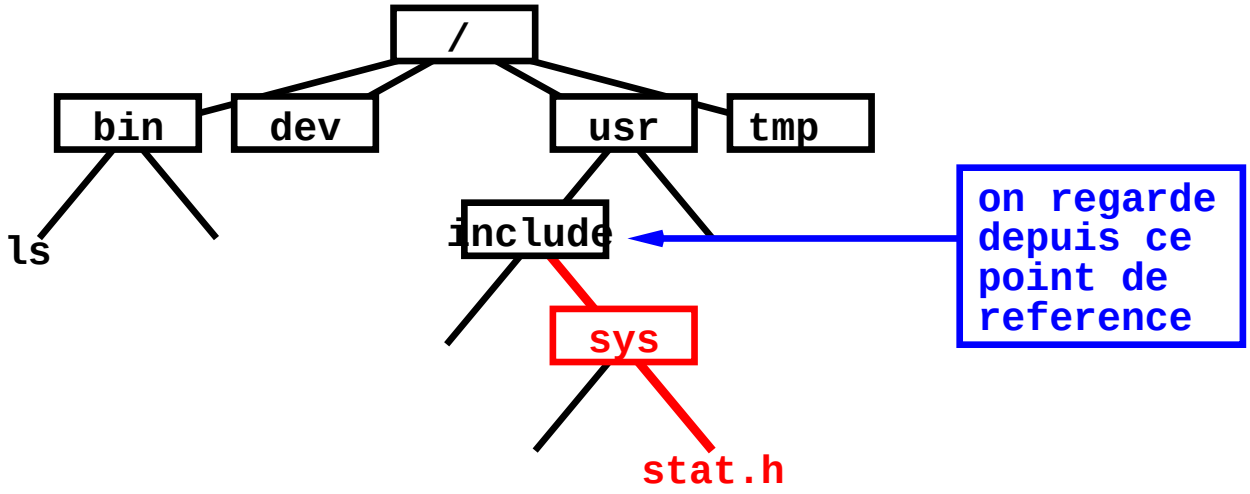
◇ **chemin d'accès relatif** :

Si le chemin d'accès ne commence pas par « / », il s'agit d'un chemin relatif :

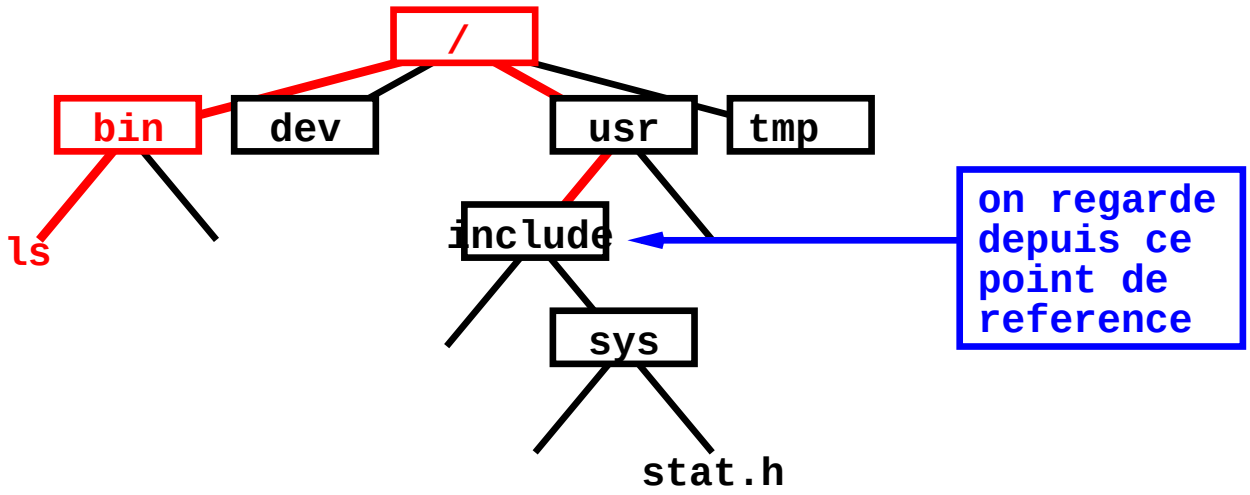
chemin d'accès relatif = répertoire1 / répertoire2 / ... / nom

Un chemin est relatif **par rapport à un point de référence**.

Par exemple, depuis « `/usr/include/` », on a le chemin relatif « `sys/stat.h` » :



Par exemple, depuis « `/usr/include/` » on a le chemin relatif « `../../bin/ls` » :



Grande importance dans les chemins relatifs des écritures « . » et « .. ».

Exemples d'utilisation du répertoire courant noté « . » (revus ultérieurement) :

- commande « `find` » pour lancer une recherche à partir de l'endroit courant :

```
find . -name exemple.txt -print
```

- lancer une commande « `commande.exe` » qui se trouve dans le répertoire courant :

```
./commande.exe
```


(en anglais *change directory*)
Syntaxe : `cd répertoire`

```
% cd /etc
% cd /usr/include
% cd ../ananas
% cd ../../src

% cd /inexistant
/inexistant: bad directory
```

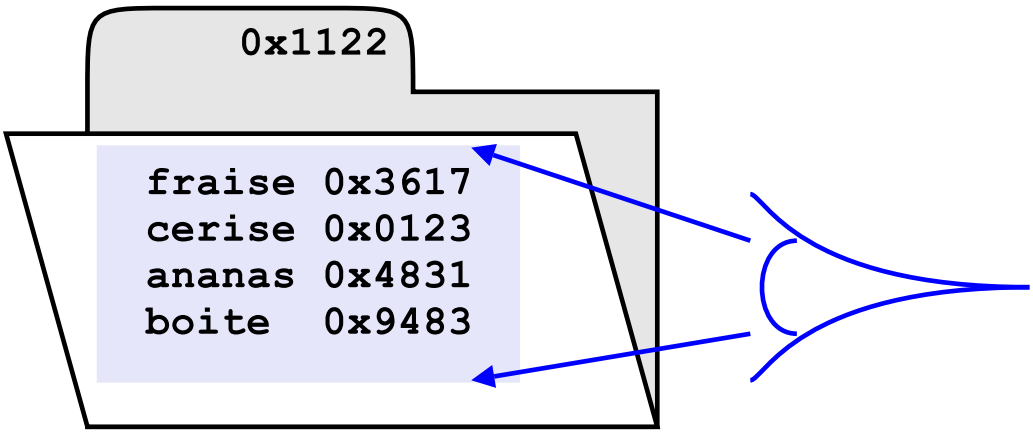
Selon le shell, le message d'erreur dans le dernier cas peut changer :

```
% cd /inexistant
bash: /inexistant: No such file or directory
```

(en anglais *present working directory*)
Syntaxe : `pwd`

```
% cd /etc
% pwd
/etc

% cd /usr/include
% pwd
/usr/include
% cd ../bin
% pwd
/usr/bin
```


Obtenir une liste d'objets, c'est lire le contenu d'un directory :

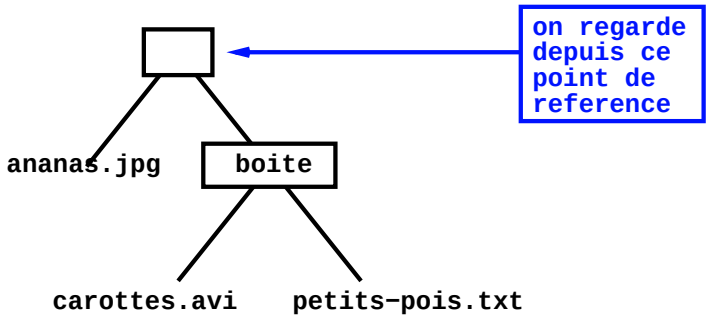


(en anglais *list*)
Syntaxe : **ls [options] objets**

Principales options (cumulables) :

- option « -l » : affichage au format long des informations relatives aux objets
- option « -g » : affichage des groupes propriétaires des objets
- option « -R » : liste récursive des objets indiqués
- option « -d » : affichage des noms des objets et non de leurs contenus
- option « -F » : affichage des objets avec un suffixe désignant le type de l'objet
- option « -a » : affichage des objets dont les noms commencent par « . »

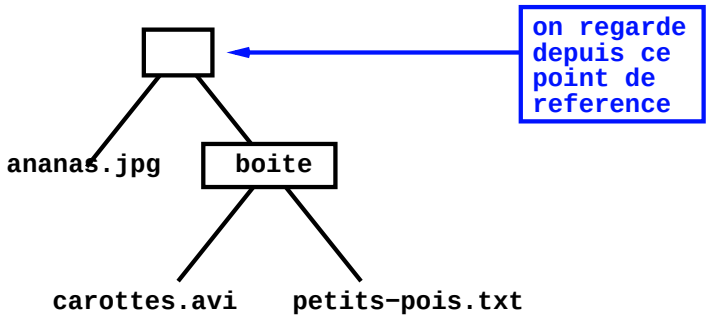
◇ Exemple 1 : commande seule



« ls » renvoie la liste des objets :

```
% ls
ananas.jpg   boite
```

◇ Exemple 2 : option « -l »



« `ls -l` » renvoie la liste des objets et de leurs informations :

```
% ls -l
-rw-r--r--   1 besancon ars    1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x   2 besancon ars     512 May 16 20:18 boite
```


ATTENTION : Selon l'âge des objets, l'affichage n'est pas le même !

Objets vieux de moins de 6 mois :

```
% ls -l
-rw-r--r--   1 besancon ars    3506 Nov 27  2005 ananas.jpg
drwxr-xr-x   2 besancon ars     512 May 16 20:18 jardin
```

Objets vieux de plus de 6 mois : affichage de l'année mais pas de l'heure :

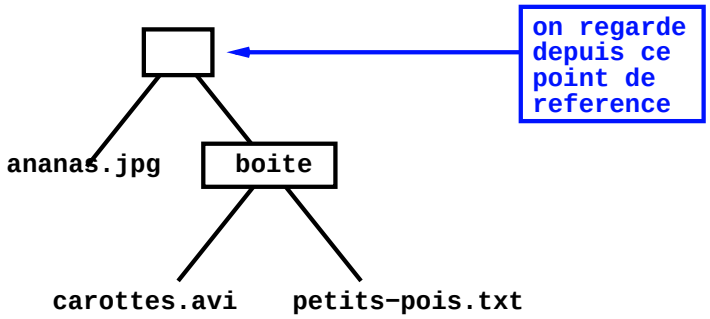
```
% ls -l
-rw-r--r--   1 besancon ars    3506 Nov 27  2005 ananas.jpg
drwxr-xr-x   2 besancon ars     512 May 16 20:18 jardin
```

⇒ l'utilisation de « `ls -l` » sera difficile dans des scripts

Solaris : options « -e » ou « -E »

```
% ls -e poire.txt
-rw-r--r--   1 besancon ars    3506 Nov 27 19:10:37 2005 poire.txt

% ls -E poire.txt
-rw-r--r--   1 besancon ars    3506 2005-11-27 19:10:37.000000 +01
00 poire.txt
```

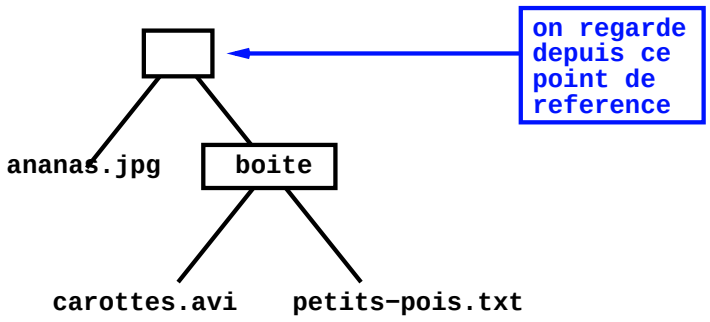

◇ Exemple 3 : option « -R »



« ls -R » renvoie la liste des objets de la sous-arborescence :

```
% ls -R
.:
ananas.jpg   boite

./boite:
carottes.avi      petits-pois.txt
```

◇ Exemple 4 : combinaison de l'option « -l » et de l'option « -R »

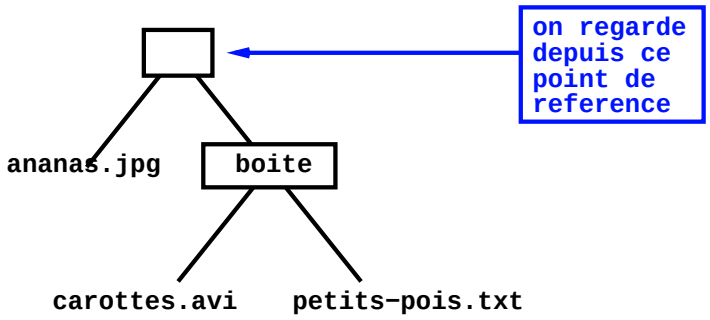


« `ls -Rl` » renvoie la liste des objets de la sous-arborescence et de leurs informations :

suite sur transparent suivant


```
% ls -Rl
.:
-rw-r--r--   1 besancon ars  1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x   2 besancon ars   512 May 16 20:18 boite

./boite:
-rw-r--r--   1 besancon ars   315 Dec 14 09:35 carottes.avi
-rw-r--r--   1 besancon ars   613 Jan 23 22:18 petits-pois.t
```

◇ Exemple 5 : option « -F »

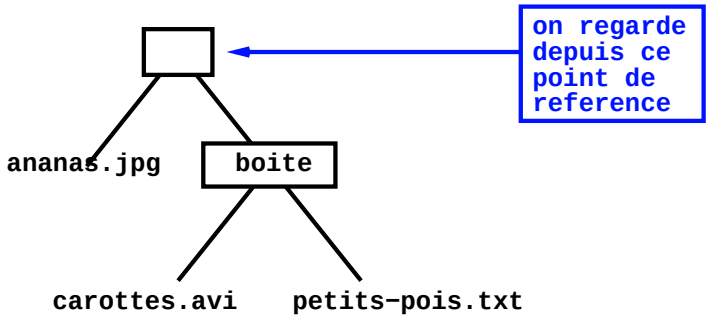


« ls -F » colle au nom de l'objet une indication sur sa nature :

- « / » pour un répertoire
- « * » pour un exécutable
- « @ » pour un lien symbolique (voir page 201)
- etc.

```
% ls -F
ananas.jpg   boite/
```


◇ Exemple 6 : combinaison de l'option « -F » et de l'option « -l »

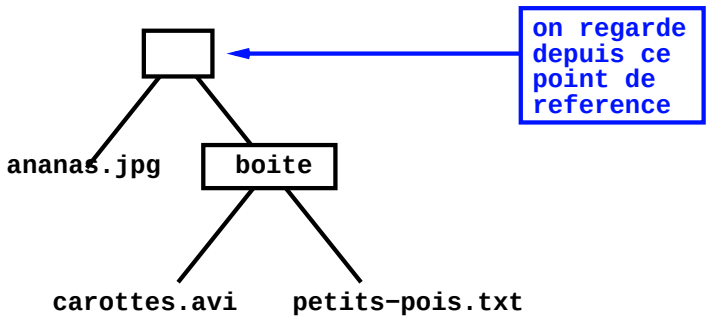


« ls -lF » colle au nom de l'objet une indication sur sa nature :

- « / » pour un répertoire
- « * » pour un exécutable
- « @ » pour un lien symbolique (voir page 201)
- etc.

```
% ls -lF
-rw-r--r--   1 besancon ars   1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x   2 besancon ars    512 May 16 20:18 boite/
```

◇ Exemple 7 : option « -a »



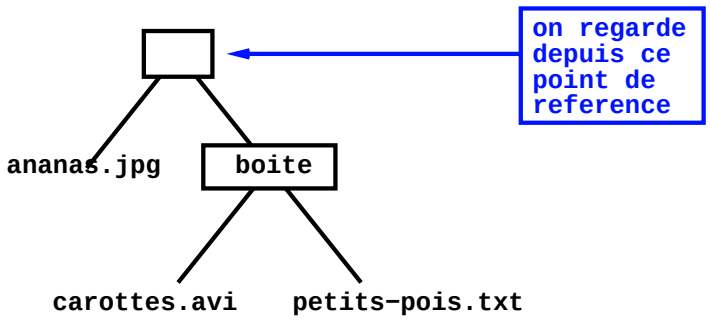

```
% ls -a
.        ..       ananas.jpg  boite
```



```
% cd $HOME
% ls -aF
./                .dia/             .kshrc*           .qt/
../               .dt/              .less             .rhosts
.ICAClient/       .dtprofile*       .lessrc           .shosts
.TTauthority      .exrc             .mailcap          .signature
.Xauthority       .fetchmail.pid    .mailrc           .ssh/
.Xdefaults        .fetchmailrc      .mime.types       .sunw/
.Xresources       .fonts.cache-1    .mozilla/         .sversionrc*
.acrobat/         .foprc            .mpdefaults       .tcshrc@
.adobe/           .gimp-2.0/        .mushuser         .xine/
.antiword/        .gnome/           .mysql_history    .xinitrc@
.bash_history     .gnome2/          .netscape/        .xnviewrc
.bash_login       .gphoto/          .plan             .xserverrc*
.bash_logout      .hushlogin/       .profile*         .xsession@
.bashrc           .ispell           .project          ananas.txt
.dbxinit          .java/            .psql_history     cerise.txt
```

Par défaut, la commande « ls » n'affiche pas les noms de objets commençant par « . » qui par convention sont des fichiers de configuration d'utilitaires.

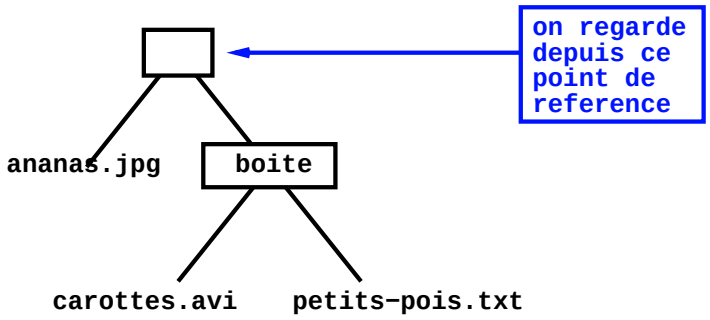
◇ Exemple 8 : différence entre contenant et contenu



Affichage du contenu :

```
% ls boite
carottes.avi      petits-pois.txt
```

```
% ls -l boite
-rw-r--r--   1 besancon ars    315 Dec 14 09:35 carottes.avi
-rw-r--r--   1 besancon ars    613 Jan 23 22:18 petits-pois.t
```

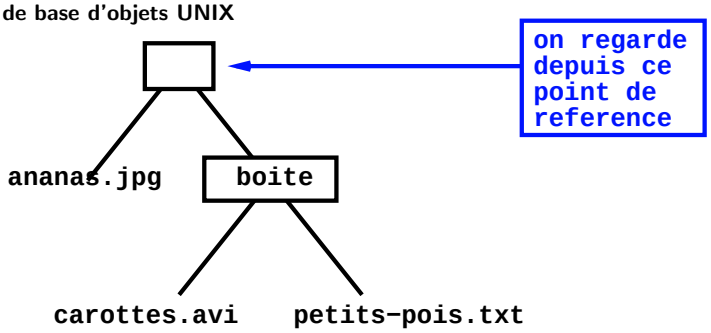



Affichage du contenant : utiliser l'option « `-d` » pour cela :

```
% ls -d boite
boite
```

Plus utile : combinaison de l'option « `-d` » et de l'option « `-l` » :

```
% ls -ld boite
drwxr-xr-x   2 besancon ars    512 May 16 20:18 boite
```



Affichage du contenant (suite) :
« `ls` » (sans option) renvoie la liste des objets **contenus dans le répertoire courant** :

```
% ls
ananas.jpg   boite
```

équivalent à « `ls .` » :

```
% ls .
ananas.jpg   boite
```


Commande « `dir.exe` »

Commande UNIX	Commande Windows
`ls -C`	`dir.exe /d`
`ls -lR`	`dir.exe /s`
sans équivalence (pas de 8x3)	`dir.exe /x`
`ls -l`	`dir.exe /q`
`ls -l | more`	`dir.exe /p`

(en anglais *make directory*)
Syntaxe : **mkdir [options] répertoires**

```
% mkdir jardin
% ls -ld jardin
drwxr-xr-x   2 besancon ars        512 May 25 23:46 jardin
```

Taille minimale d'un répertoire (même vide) : 512 octets

Raison : 512 octets ≡ allocation minimale par le système pour cette catégorie d'objets (la taille sera toujours un multiple de 512)

(hmmm... certains LINUX ne respectent plus cela)

Pour créer des répertoires emboités :

```
% mkdir repertoire1
% mkdir repertoire1/repertoire2
% mkdir repertoire1/repertoire2/repertoire3
```

Pas pratique !

Plus pratique : création directe de sous répertoires en cascade possible via option « -p » :

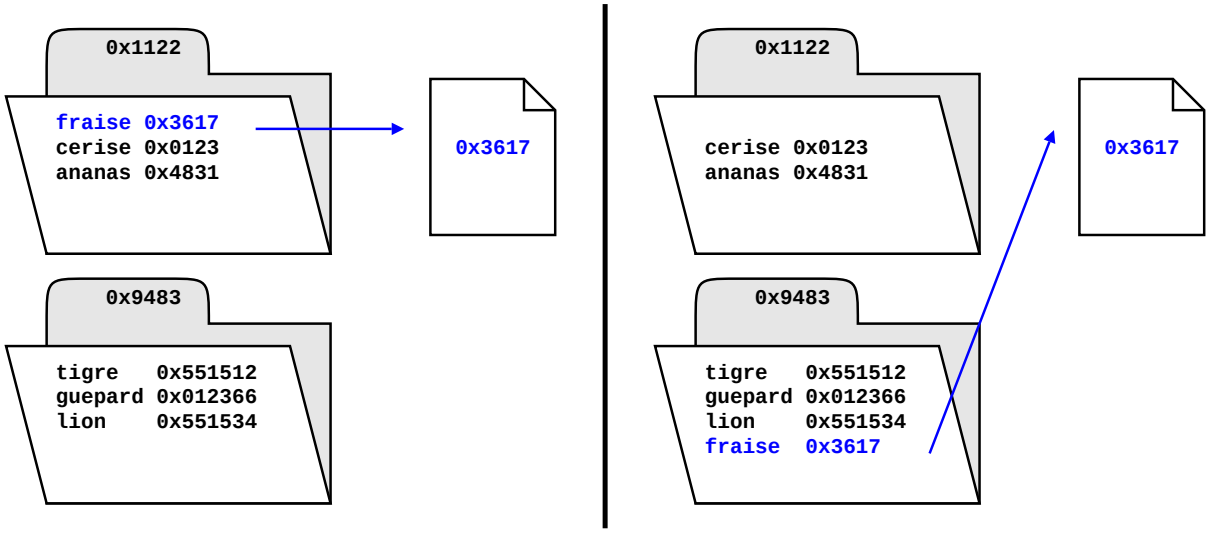
```
% mkdir -p repertoire1/repertoire2/repertoire3
% ls -R
.:
repertoire1

./repertoire1:
repertoire2

./repertoire1/repertoire2:
repertoire3

./repertoire1/repertoire2/repertoire3:
```


Chapitre 6 • Commandes de manipulation de base d'objets UNIX
§6.11 • (Windows : : création de répertoires : md.exe, mkdir.exe)

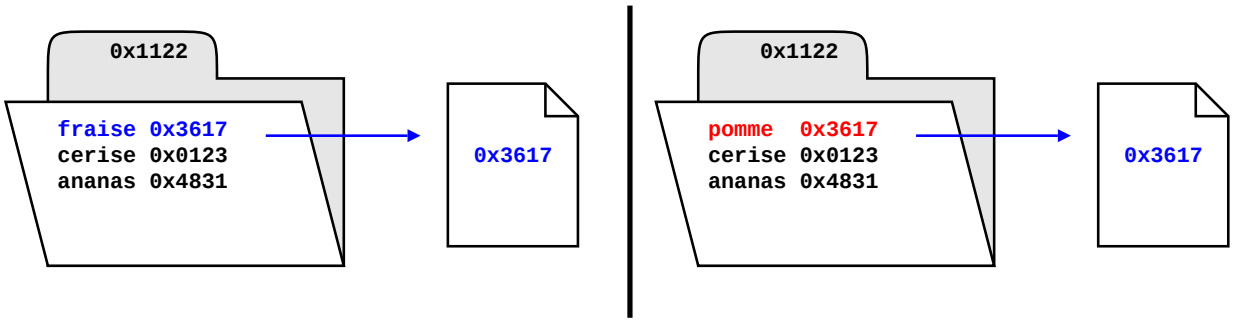
Commande « md.exe » ou « mkdir.exe ».

Commande UNIX	Commande Windows
mkdir dossier	md.exe dossier
mkdir dossier	mkdir.exe dossier

Déplacer un objet est le rattacher ailleurs dans l'arborescence à un autre répertoire :



Renommer un objet est changer son rattachement dans le répertoire :



UNIX fournit une seule commande pour ces opérations.

La commande permet à la fois :

- de changer le rattachement d'un objet
- de changer le nom du rattachement

(en anglais *move*)
Syntaxe générale :

```
mv [options] objets objet
```

Quelques options :

- « `-i` » : demande de confirmation à chaque écrasement d'objet

Syntaxes possibles en pratique :

- `mv [options] fichier1 fichier2-inexistant`
- `mv [options] fichier1 fichier2-existant`
- `mv [options] objets répertoire-existant`

◇ Renommage d'objet :

Soit :

```
% ls -l
-rw-r--r--   1 besancon ars   1035 May 25 22:59 ananas.avi
-rw-r--r--   1 besancon ars   2893 May 25 22:59 banane.txt
```

On fait :

```
% mv ananas.avi film.avi
```

L'objet « film.avi » n'existait pas avant. Maintenant on a :

```
% ls -l
-rw-r--r--   1 besancon ars   2893 May 25 22:59 banane.txt
-rw-r--r--   1 besancon ars   1035 May 25 22:59 film.avi
```


◇ Déplacement d'objets :

```
% ls -l
-rw-r--r--   1 besancon ars   2893 May 25 22:59 banane.txt
drwxr-xr-x   2 besancon ars    512 May 25 23:03 cinematheque
-rw-r--r--   1 besancon ars   1035 May 25 22:59 film.avi
% mv film.avi cinematheque
% ls -l
-rw-r--r--   1 besancon ars   2893 May 25 22:59 banane.txt
drwxr-xr-x   2 besancon ars    512 May 25 23:05 cinematheque
% ls -lR
.:
-rw-r--r--   1 besancon ars   2893 May 25 22:59 banane.txt
drwxr-xr-x   2 besancon ars    512 May 25 23:05 cinematheque

./cinematheque:
-rw-r--r--   1 besancon ars   1035 May 25 22:59 film.avi
```

◇ Confirmation avec écrasement :

```
% ls -l
-rw-r--r--   1 besancon ars   1035 May 25 22:59 film.avi
-rw-r--r--   1 besancon ars   2893 May 25 23:16 mummy.avi

% mv -i film.avi mummy.avi
mv: overwrite mummy.avi (yes/no)? y

% ls -l
-rw-r--r--   1 besancon ars   1035 May 25 22:59 mummy.avi
```


Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.13 • (Windows : : déplacer des objets : move.exe)

Commande « move.exe »

Commande UNIX	Commande Windows
mv objet dossier	move.exe [/Y | /-Y] [lecteur:][chemin
mv dossier1 dossier2	mkdir.exe [/Y | /-Y] [lecteur:][chemi

Commande « ren.exe »
Commande « rename.exe »

Commande UNIX	Commande Windows
mv objet1 objet2	rename.exe [lecteur:][chemin]nom_de_fichi
mv objet1 objet2	ren.exe [lecteur:][chemin]nom_de_fichier1

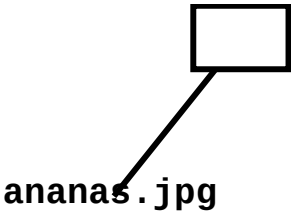
(en anglais *copy*)

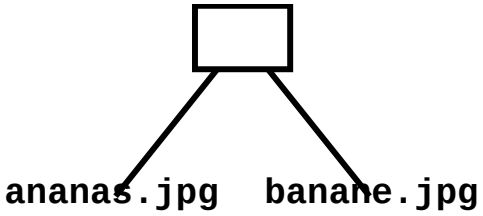
Plusieurs syntaxes possibles :

1. **cp [options] fichier1 fichier2**
   dupliquer l'objet de départ sous le nom de destination
2. **cp [options] fichiers dossier**
   dupliquer les fichiers dans le dossier indiqué
3. **cp -r [options] dossiers dossier**
   dupliquer les dossiers dans le dossier indiqué

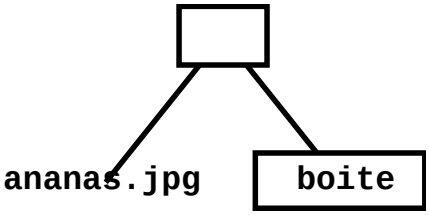
Quelques options :

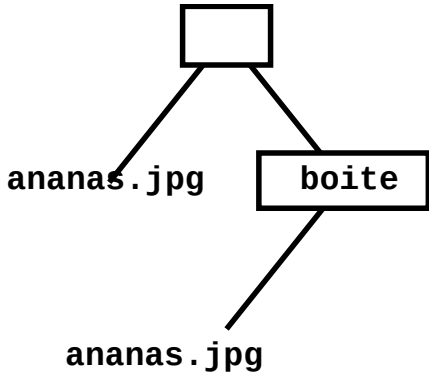
- « -r » : copie récursive
- « -i » : confirmation à chaque écrasement de fichier
- « -p » : conservation des dates (et des propriétaires utilisateur et groupe si commande lancée par l'administrateur)



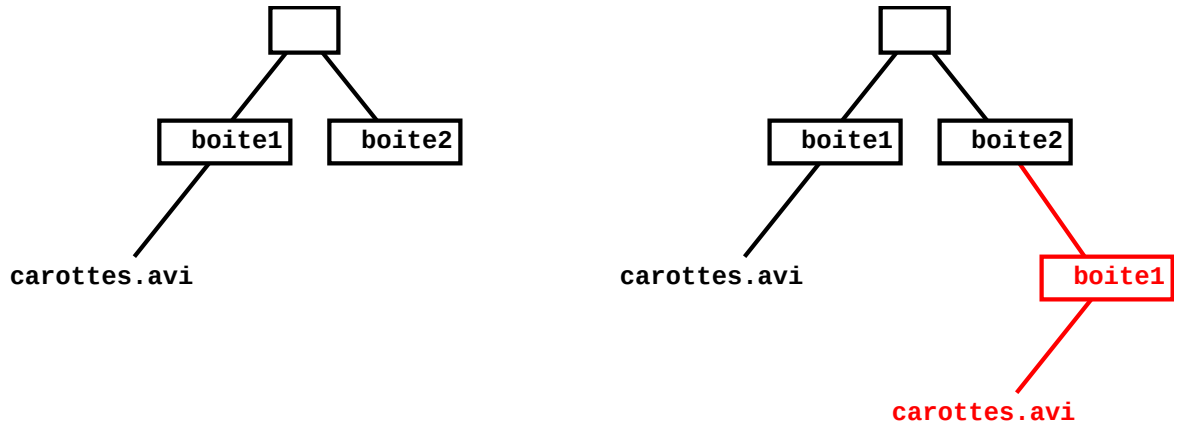



```
% cp ananas.jpg banane.jpg
```




```
% cp ananas.jpg boite
```




```
% cp -r boite1 boite2
```


La copie ne conserve pas les dates

```
% cp /etc/motd  exemple.txt
% ls -l /etc/motd exemple.txt
-rw-r--r--   1 root     sys    49 Apr  7  2002 /etc/motd
-rw-r--r--   1 besancon ars    49 Jul  6 19:11 exemple.txt
```

⇒ option « -p » pour conserver les dates pendant la copie

```
% cp -p /etc/motd exemple.txt
% ls -l /etc/motd exemple.txt
-rw-r--r--   1 root     sys    49 Apr  7  2002 /etc/motd
-rw-r--r--   1 besancon ars    49 Apr  7  2002 exemple.txt
```

Commande « copy.exe »

```
C:\>copy.exe /?
Copie un ou plusieurs fichiers sur un autre emplacement.

COPY [/V] [/N] [/Y | /-Y] [/Z] [/A | /B ] source [/A | /B]
     [+ source [/A | /B] [+ ...]] [cible [/A | /B]]
```

Pas de copie récursive.

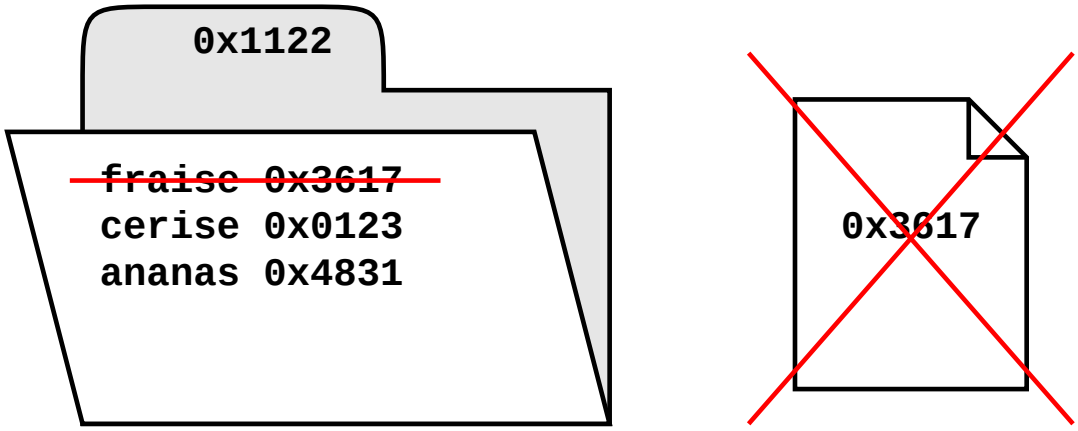
Commande « xcopy.exe »

```
C:\>xcopy.exe /?
Copie des fichiers et des arborescences de répertoires.

XCOPY source [destination] [/A | /M] [/D[:date]] [/P] [/S [/E]] [/V] [/W]
                           [/C] [/I] [/Q] [/F] [/L] [/H] [/R] [/T] [/U]
                           [/K] [/N] [/O] [/X] [/Y] [/-Y] [/Z]
                           [/EXCLUDE:fich1[+fich2][+fich3]...]
```

Utilisation pratique :

- créer la destination : « `md.exe dest` »
- copie par : « `xcopy *.txt dest /e /c /h /k /o` »

Détruire un objet, c'est le supprimer du directory qui l'associe à l'inode :

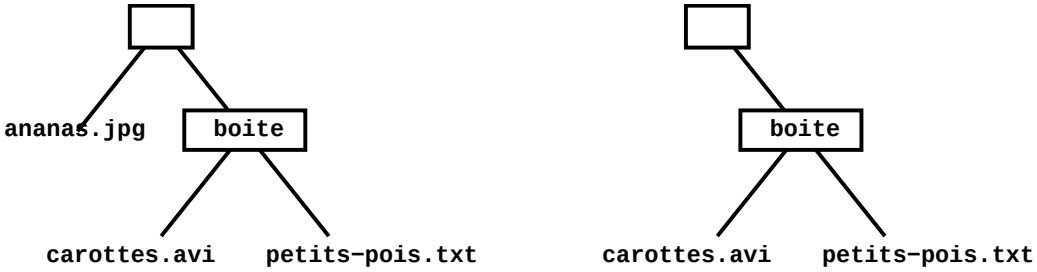


(en anglais *remove*)
Syntaxe : **rm [options] objets**

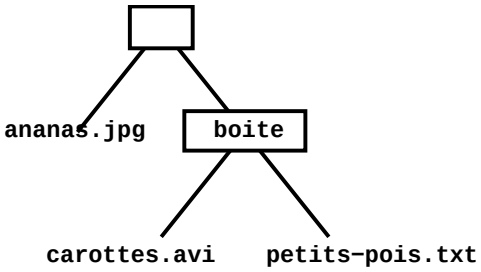
Quelques options :

- option « -i » : confirmation à chaque suppression (garde fou)
- option « -r » : suppression récursive
- option « -f » : suppression en force d'un objet même si ses droits ne s'y prêtent pas

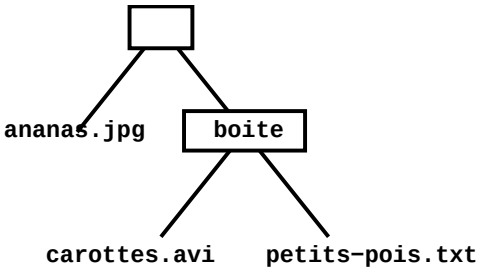
« rm -rf répertoires » permet de supprimer récursivement toute une arborescence sans demande de confirmation. Attention : dangereux.

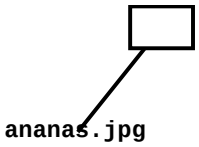



```
% ls
ananas.jpg  boite

% rm ananas.jpg
% ls
boite
```

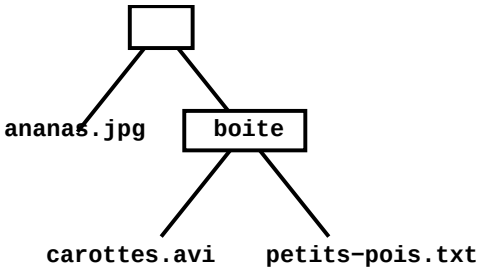



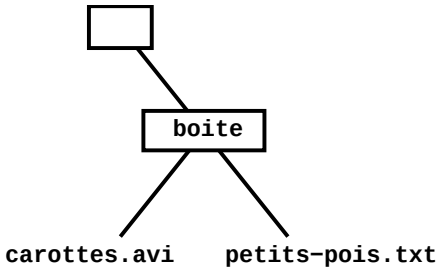

```
% rm boite
rm: boite is a directory

% rmdir boite
rmdir: boite: Directory not empty
% ls boite
carottes.avi      petits-pois.txt
```






```
% rm -rf boite
% ls
ananas.jpg
```




```
% rm -i ananas.jpg
rm: remove ananas.jpg? y
% ls
boite
```

Commande UNIX	Commande Windows
`rm -i`	`del.exe /p`
`rm -r`	`del.exe /s`
`rm -f`	`del.exe /f`

(en anglais *remove directory*)
Syntaxe : **`rmdir répertoires`**
On ne peut effacer avec cette commande qu'un répertoire vide.
⇒ pénalisant
⇒ on préferera souvent la commande « `rm -rf` »

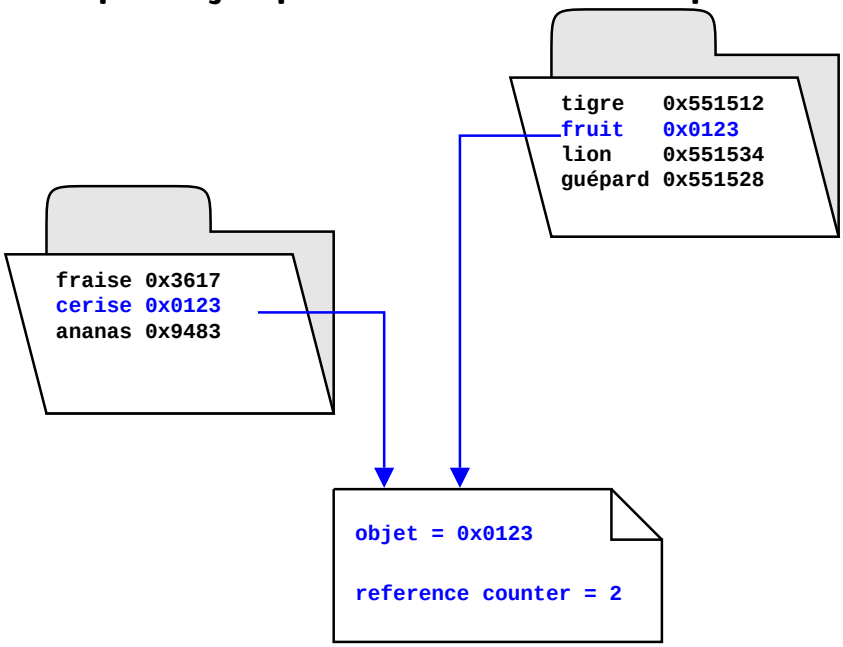
```
% cp /etc/motd dossier/fichier.txt
% rmdir dossier
rmdir: dossier: Directory not empty
```

Pas d'options dignes d'intérêt.

Commande « rd.exe »

Commande UNIX	Commande Windows
Pas d'équivalent car demande de confirmation inexistante sur UNIX	rd.exe rep
rmdir rep	rd.exe /q rep
rm -rf rep	rd.exe /s /q rep

Rappel : les noms sont stockés dans les répertoires.
Un nom est appelé un lien sur l'objet.
Sur UNIX à chaque objet peuvent être associés plusieurs noms.



Dans l'inode d'un objet, il y a un compteur de liens :

- compteur incrémenté lors de la création d'un nouveau lien
- compteur décrémenté lors de la suppression d'un lien
- l'objet est détruit lorsque le dernier lien sur l'objet est supprimé

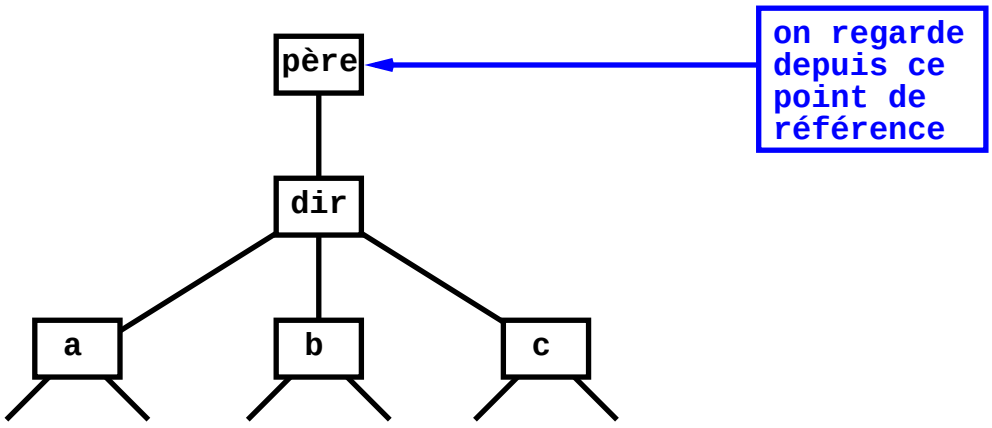
On voit les valeurs des compteurs de liens via la commande « `ls -l` » :

```
% ls -l
-rw-r--r--   1 besancon ars        39 Oct 26  2003 ananas
-rw-r--r--   1 besancon ars        35 Jul  3 17:38 banane
drwxr-xr-x   4 besancon ars       512 Jul  4 15:48 cerise
drwxr-xr-x   2 besancon ars       512 Nov 20  2003 endive
drwxr-xr-x   3 besancon ars       512 Jul  5 00:36 fraise
```

Rappel :

- « `.` » est un lien
- « `..` » est un lien

Soit l'arborescence :

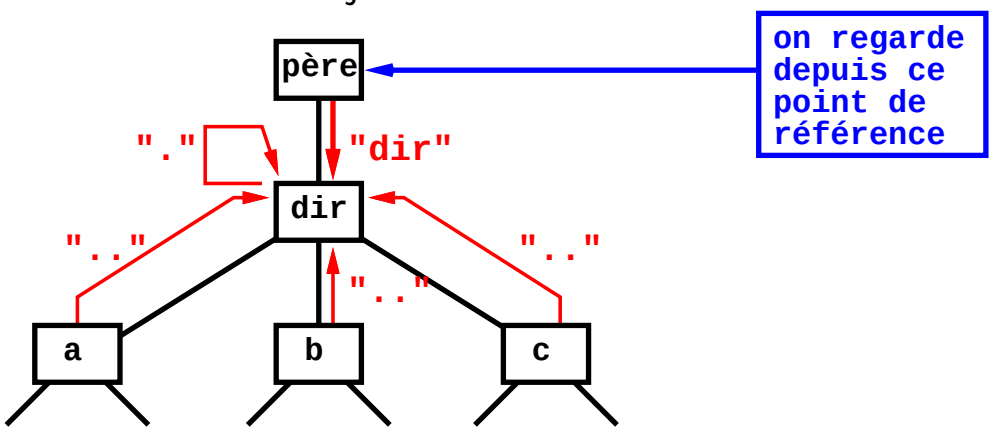


et l'affichage :

```
% ls -l
drwxr-xr-x   5 besancon ars       512 Jul  5 00:29 dir
```

Pourquoi a-t-on l'indication de 5 liens sur « dir » ?

Il y a en effet 5 liens sur l'objet nommé « dir » :



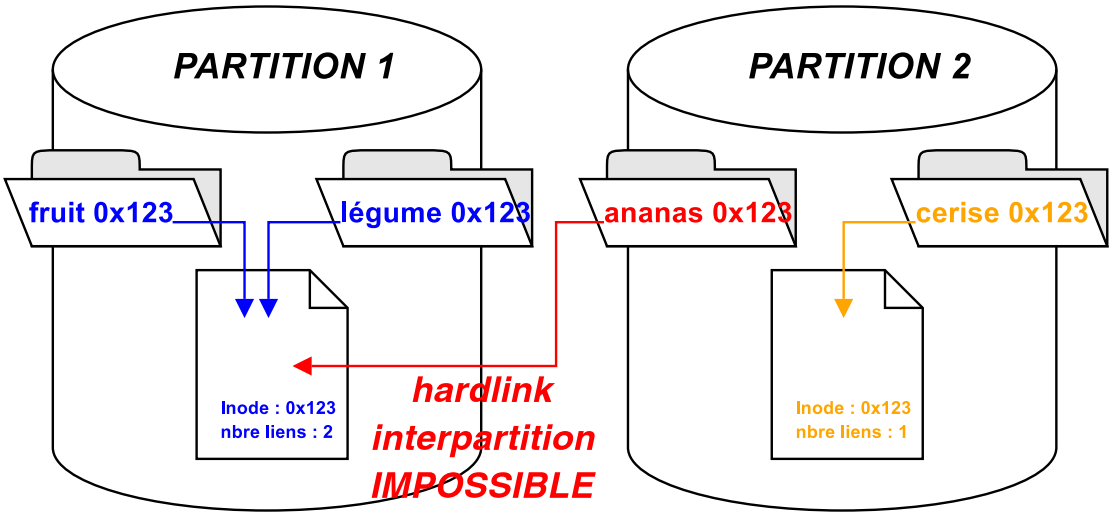
Ces 5 noms sont :

1. lien « `/chemin/vers/dir` »
2. lien « `/chemin/vers/dir/.` »
3. lien « `/chemin/vers/dir/a/..` »
4. lien « `/chemin/vers/dir/b/..` »
5. lien « `/chemin/vers/dir/c/..` »

Preuve via l'utilisation de l'option « `-i` » de « `ls` » qui affiche les numéros d'inodes :

```
% ls -ldi dir dir/. dir/a/.. dir/b/.. dir/c/..
550907 drwxr-xr-x  5 besancon ars     512 Jul  5 00:29 dir
550907 drwxr-xr-x  5 besancon ars     512 Jul  5 00:29 dir/.
550907 drwxr-xr-x  5 besancon ars     512 Jul  5 00:29 dir/a/..
550907 drwxr-xr-x  5 besancon ars     512 Jul  5 00:29 dir/b/..
550907 drwxr-xr-x  5 besancon ars     512 Jul  5 00:29 dir/c/..
```

(en anglais *link*)
Le lien **hard** utilise le numéro d'inode pour trouver l'objet. Le numéro est unique par partition. ⇒ un lien hard reste interne à une partition.

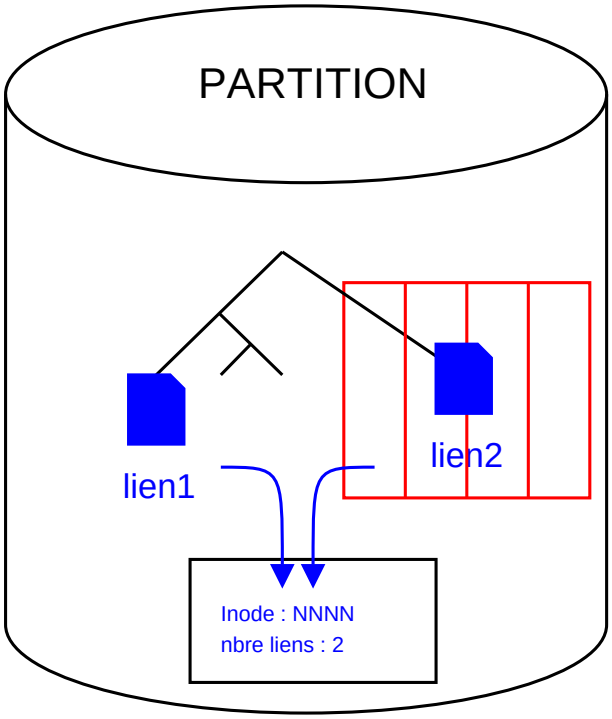


Les interdits :

- on ne peut pas faire de hard link vers une autre partition (car impossibilité d'adresser l'inode d'une autre partition depuis un répertoire)
- on ne peut pas faire de hard link vers un répertoire (car sinon boucles invisibles impossibles à détecter dans l'arborescence)

A quoi sert un lien hard ?

Exemple : environnement chrooté (sera revu plus tard)



Principe du chroot : il restreint l'accès au contenu d'une partie d'arborescence (dite la *cage*) et on ne peut pas accèder au contenu extérieur de la cage

Pourquoi utiliser un lien hard ? : avec un lien hard, un objet peut être à l'intérieur et à l'extérieur de la cage du chroot selon le lien utilisé

(en anglais *link*)

La commande à utiliser est : **ln original synonyme**

```
% ls -l ananas.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg

% ln ananas.jpg fruit.jpg

% ls -l ananas.jpg fruit.jpg
-rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
-rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg

% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg
```

Suppression d'un lien hard par la commande « rm »

```
% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg

% rm ananas.jpg

% ls -li fruit.jpg
357 -rw-r--r--  1 besancon ars  9919 Jul 14 10:15 fruit.jpg
```


Place occupée

```
% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg

% du -k .
3       .
% rm fruit.jpg
% du -k .
3       .
```

En aucune façon, on ne double la place consommée !

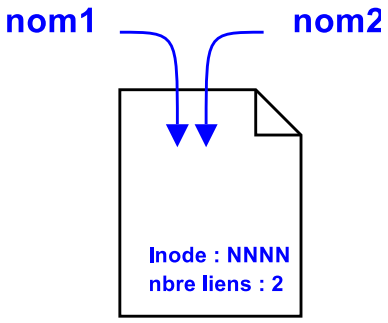
Les liens hard sont définis au sein d'une même partition.

Comment trouver tous les liens d'un même inode ?

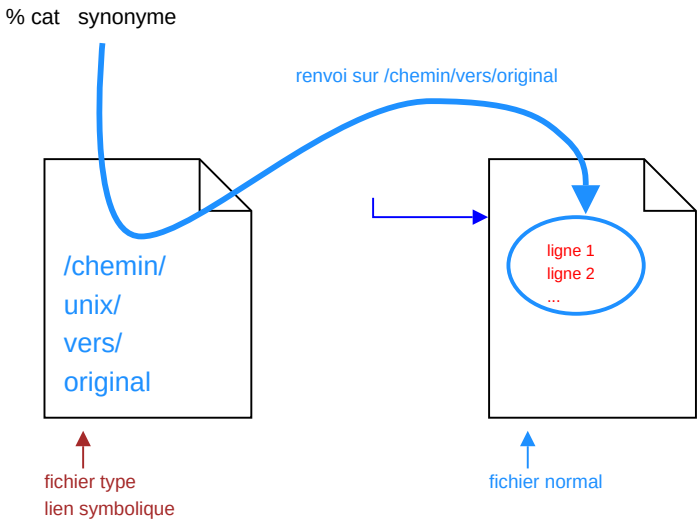
Grâce à la commande « `find` ». Voir page 440.

ATTENTION : le mot lien a deux sens sur UNIX :

lien **hard**



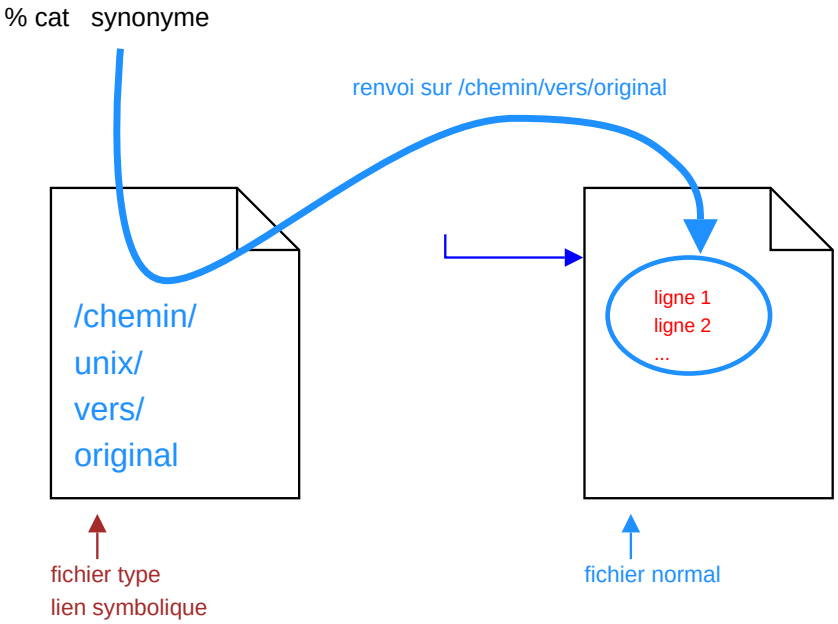
lien **symbolique**



ATTENTION : ce sont des notions différentes !

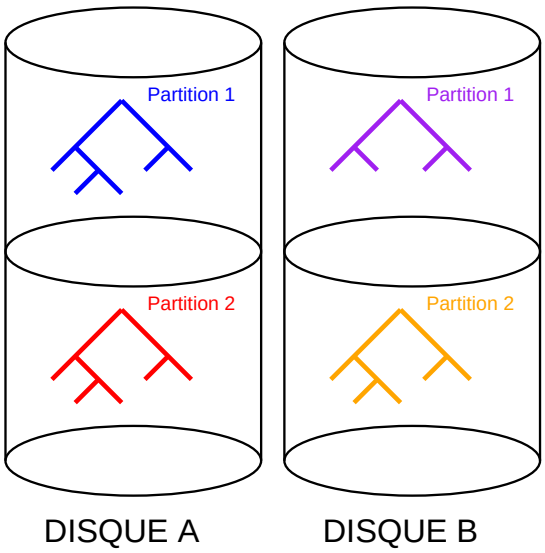
(en anglais *symbolic link*)

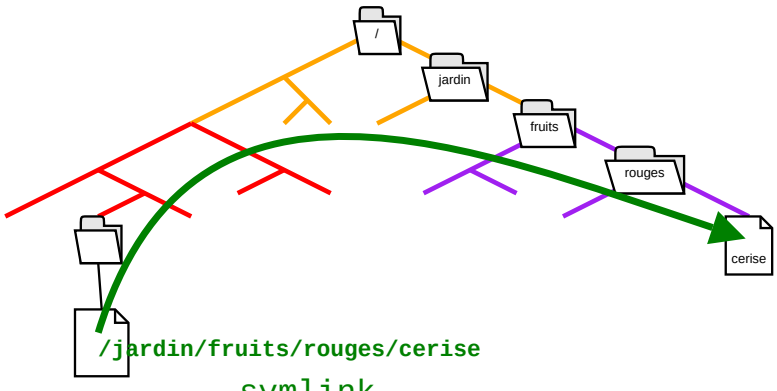
Un lien **symbolique** est un fichier spécial contenant le chemin d'un autre objet.



On dit « lien symbolique », « symbolic link », « symlink ».

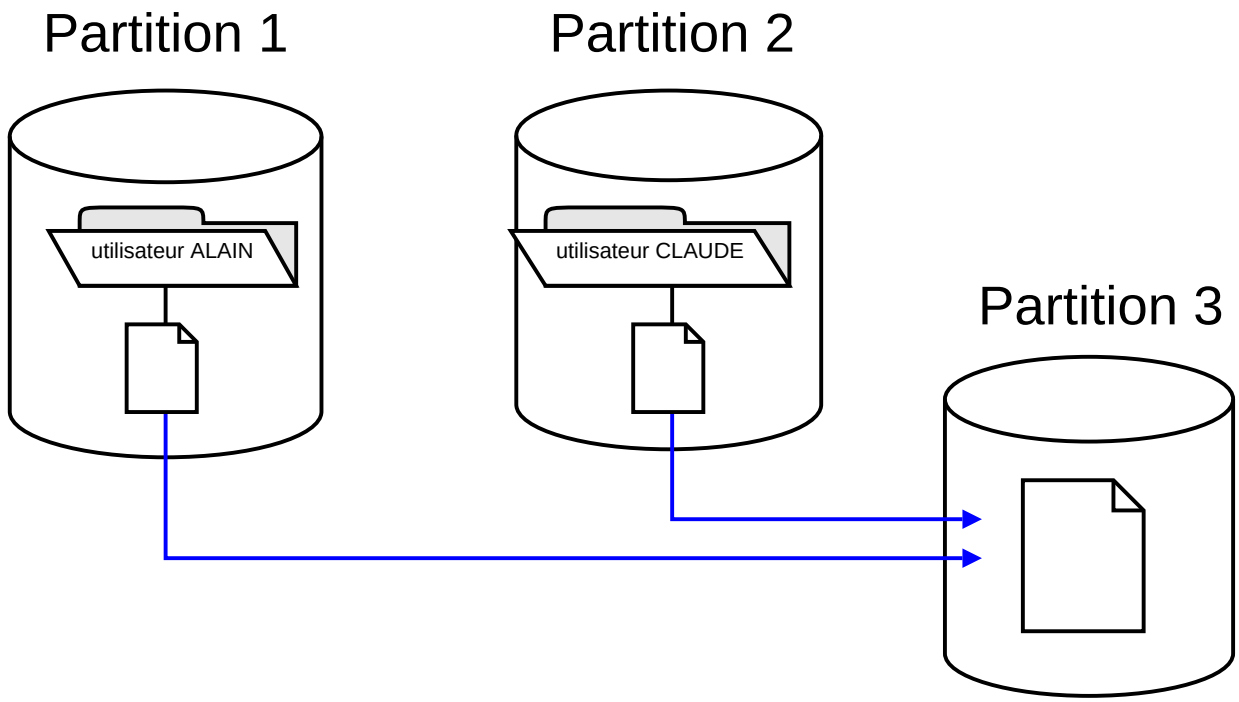
Un lien **symbolique** est non limité à une partition d'un disque dur parce qu'utilisant le chemin d'un objet et non pas son numéro d'inode.





A quoi sert un lien symbolique ?

Exemple : fichier de configuration commun à tous les utilisateurs



Pourquoi ne peut-on pas employer des liens hard dans cet exemple ?

La commande à utiliser est : **ln -s original synonyme**

```
% ls -l ananas.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
% ln -s ananas.jpg fruit.jpg

% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
358 lrwxr-xr-x  1 besancon ars    10 Oct 17 18:26 fruit.jpg -> ananas.jpg

% ls -lL ananas.jpg fruit.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 fruit.jpg
```

Suppression d'un lien symbolique par « rm »

```
% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
358 lrwxr-xr-x  1 besancon ars    10 Oct 17 18:26 fruit.jpg -> ananas.jpg

% rm ananas.jpg
% ls -liL fruit.jpg
358 lrwxr-xr-x  1 besancon ars    10 Oct 17 18:26 fruit.jpg -> ananas.jpg
% cat fruit.jpg
cat: fruit.jpg: No such file or directory
```


Les systèmes UNIX imposent les droits `lrwxr-xr-x` sur le lien (selon l'UNIX cela pourra être à la place `lrwxrwxrwx`).
Ils ne peuvent pas être modifiés.
On ne peut que changer les droits d'un fichier pointé par un lien symbolique :

```
% ls -l ananas.jpg fruit.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
lrwxr-xr-x  1 besancon ars    10 Oct 17 18:26 fruit.jpg -> ananas.jpg

% chmod 600 fruit.jpg
% ls -l
-rw-------  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
lrwxr-xr-x  1 besancon ars    10 Oct 17 18:26 fruit.jpg -> ananas.jpg

% ls -lL ananas.jpg fruit.jpg
-rw-------  1 besancon ars  9919 Jul 14 10:15 ananas.jpg
-rw-------  1 besancon ars  9919 Jul 14 10:15 fruit.jpg
```

Sous Windows notion de *raccourci*, *shortcut*.

Un raccourci Windows n'est pas équivalent à un lien symbolique UNIX !

- pas de commande standard pour générer un raccourci ; uniquement via interface graphique ou API (VBS, etc.) a priori
- impossibilité d'accéder via des commandes en ligne à l'objet via le raccourci comme on le ferait sous UNIX (voir ci-après) ; il faut passer par l'interface graphique à la place

◇ Echec d'un « `cd` » sur le raccourci sur un dossier :

```
C:\Documents and Settings\besancon\My Documents>dir
 Volume in drive C is Windows XP
 Volume Serial Number is 0C5C-E708

 Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  20:53    <DIR>          .
06/07/2004  20:53    <DIR>          ..
06/07/2004  20:52               725 Shortcut-to-My-Pictures.lnk
               1 File(s)            725 bytes
               2 Dir(s)     551 477 248 bytes free

C:\Documents and Settings\besancon\My Documents>cd Shortcut-to-My-Pictures.lnk

The directory name is invalid.
```

◇ Echec d'un « dir » sur le raccourci sur un dossier :

```
C:\Documents and Settings\besancon\My Documents>dir Shortcut-to-My-Pictures.lnk
 Volume in drive C is Windows XP
 Volume Serial Number is 0C5C-E708

 Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  20:52               725 Shortcut-to-My-Pictures.lnk
               1 File(s)            725 bytes
               0 Dir(s)     551 477 248 bytes free
```


◇ Echec d'un « type » sur le raccourci sur un fichier texte :

```
C:\Documents and Settings\besancon\My Documents>dir
 Volume in drive C is Windows XP
 Volume Serial Number is 0C5C-E708

 Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  21:06    <DIR>          .
06/07/2004  21:06    <DIR>          ..
06/07/2004  21:00             1 870 a.txt
06/07/2004  21:06               757 b.txt.lnk
...

C:\Documents and Settings\besancon\My Documents>type b.txt.lnk
L   ?q?     +      Fc       á?lÖèc-?É?sIîc-?á.álïc-?N      ?               e?k ?
PaO- O:i?ó +00Y? /C:\                    \ 1     -0?p? \DOCUME~1  D ? ? n+N0Auu0lô
q   D o c u m e n t s   a n d   S e t t i n g s   ? @ 1     @0,V? besancon  ( ?
? n+O0qsu0lôq   b e s a n c o n   ? d 1     S0sk? MYDOCU~1  0 ? ? n+O0qsu0lôq
M y   D o c u m e n t s   ? ? ? ? n+b e s a n c o n   ? 6 2 N  u0?y  a.txt " ? ?
 n+u0-ûu0?ÿq   a . t x t   q   n   ?   ?   ?   7       m   ?   ? t\??   Windows
XP C:\Documents and Settings\besancon\My Documents\a.txt   . \ a . t x t / C : \
 D o c u m e n t s   a n d   S e t t i n g s \ b e s a n c o n \ M y   D o c u m
 e n t s ` ?  áX       best            =+A-?=+N2bb++-î,33lYl-+?a· PV=+A-?=+Nb
b++-î,3ElYl-+?a· PV+
```

Le système NTFS de Windows offre la notion de « *junction* » mais il y a peu d'utilitaires pour les utiliser.

On trouve :

- « `linkd.exe` », « `delrp.exe` » dans le resource kit Windows 2K/XP/2K3
- « `junction.exe` » d'URL `http://www.sysinternal.com/ntw2k/source/misc.shtml\#junc`

◇ Création d'une junction :

```
C:\Documents and Settings\besancon\My Documents>junction tools2 tools

Junction v1.03 - Win2K junction creator and reparse point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com

Created: C:\Documents and Settings\besancon\My Documents\tools2
Targetted at: C:\Documents and Settings\besancon\My Documents\tools
```

Et on voit bien la junction « tools2 » :

```
C:\Documents and Settings\besancon\My Documents>dir
 Volume in drive C is Windows XP
 Volume Serial Number is 0C5C-E708

 Directory of C:\Documents and Settings\besancon\My Documents


06/07/2004  22:14    <DIR>          .
...
06/07/2004  22:13    <DIR>          tools
06/07/2004  22:14    <JUNCTION>     tools2
...
```

◇ Utilisation d'une junction :

```
C:\Documents and Settings\besancon\My Documents>dir tools2
 Volume in drive C is Windows XP
 Volume Serial Number is 0C5C-E708

 Directory of C:\Documents and Settings\besancon\My Documents\tools2

06/07/2004  22:13    <DIR>          .
06/07/2004  22:13    <DIR>          ..
15/01/2000  09:34               749 README.TXT
               1 File(s)            749 bytes
               2 Dir(s)     423 448 576 bytes free

C:\Documents and Settings\besancon\My Documents>
```


◇ Utilisation d'une junction (2) :

```
C:\Documents and Settings\besancon\My Documents>junction tools2

Junction v1.03 - Win2K junction creator and reparse point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com

C:\Documents and Settings\besancon\My Documents\tools2: JUNCTION
   Substitute Name: C:\Documents and Settings\besancon\My Documents\tools
```

◇ Destruction de la junction :

```
C:\Documents and Settings\besancon\My Documents>junction -d tools2

Junction v1.03 - Win2K junction creator and reparse point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com

Deleted tools2.
```

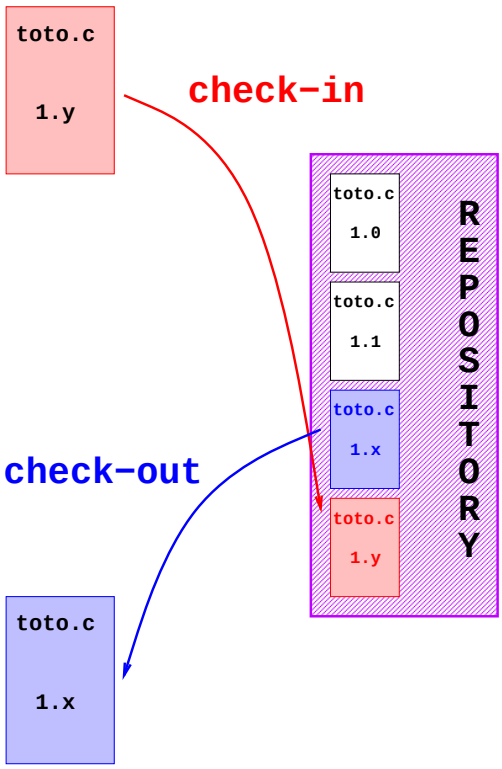
Chapitre 7

Gestion de versions de fichiers

Principe général à tous les outils de gestion de versions de fichiers :

On stocke en fait les « `diff` » entre versions successives du fichier.

Seuls les noms du répertoire de repository et les noms des commandes de check-in et check-out changeront d'un package de gestion à un autre.



SCCS ≡ *Source Code Control System*
1975

Nom du repository	« `SCCS` »
Création dans le repository	« `sccs create programme.c` »
Check-out en read-write	« `sccs edit programme.c` »
Check-in	« `sccs delta programme.c` »
Check-out en read-only	« `sccs get programme.c` »
Comparaison avec le repository	« `sccs diffs programme.c` »
Historique des versions	« `sccs prt programme.c` »

On a :

```
sccs deledit == sccs delta + sccs edit
sccs delget  == sccs delta + sccs get
```


RCS ≡ *Revision Control System*
« `ftp://ftp.lip6.fr/pub/gnu/rcs/` »

Nom du repository	« `RCS` »
Création dans le repository	« `rcs -i programme.c` »
Check-out en read-write	« `co -l programme.c` »
Check-in	« `ci programme.c` »
Check-out en read-only	« `co programme.c` »
Comparaison avec le repository	« `rcsdiff programme.c` »
Historique des versions	« `rlog programme.c` »

On a :

```
ci -l  == ci + co -l
```

CVS ≡ *Concurrent Version System*
Construit au dessus de RCS (pour sa gestion interne des fichiers mais les commandes RCS ne sont pas utilisées)

Utilisé par de nombreuses équipes de développeurs de programmes sur Internet.
⇒ CVS utilisera le terme de *projet*
⇒ CVS peut fonctionner en réseau, le repository peut être sur une machine distante

Pour utiliser CVS, on doit définir 2 variables d'environnement :

- variable « `CVSROOT` » : chemin du repository par défaut
- variable « `EDITOR` » : éditeur de texte par défaut (« `vi` » par défaut)

Pour travailler sous CVS :

- créer un repository ou se connecter à un repository

```
% CVSROOT=/chemin/vers/mon/projet
% export CVSROOT
% cvsinit
```

  La commande « `cvsinit` » crée les répertoires et les fichiers d'administration nécessaires.
- commande « `cvs checkout filename` »
  Une copie du fichier indiqué est extraite du repository.
- commande « `cvs add filename` »
  Ajoute le fichier indiqué à l'arborescence du repository.
- commande « `cvs remove filename` »
  Supprime un fichier du repository.
- commande « `cvs commit` »
  Entérine les actions des « `cvs add` » et « `cvs remove` ».
  Incorpore au repository vos modifications.
- commande « `cvs update` »
  Pour récupérer sur son disque l'état actuel du repository.

- commande « `cvs diff` »
  Montre les différences entre votre copie locale et le repository.
- commande « `cvs export` »
  ?
- commande « `cvs history` »
  Affiche l'historique des modifications.
- commande « `cvs log` » ou commande « `cvs status` »
  Affiche des informations sur le module du répertoire de travail courant.

◇ Pour travailler avec CVS à travers le réseau

2 contextes :

- on veut suivre un projet en tant que spectateur ; on ne modifiera pas les fichiers
  ⇒ CVS anonyme
- on veut suivre un projet en tant qu'acteur actif ; on modifiera des fichiers
  ⇒ CVS authentifié

◇ CVS anonyme

On veut suivre un projet en tant que spectateur ; on ne modifiera pas les fichiers.

La phase de connexion est du type suivant :

```
% CVSROOT=:pserver:anonymous@anoncvs.example.com:/cvs/gnome
% export CVSROOT
% cvs login
CVS password: <-- taper retour chariot
```

On procède ensuite avec les commandes normales de récupération de fichiers.

◇ CVS authentifié

On veut suivre un projet en tant qu'acteur actif ; on modifiera des fichiers.

On utilisera SSH pour sécuriser la connexion :

```
% CVS_RSH=ssh
% export CVS_RSH
% CVSROOT=:ext:user@cvs.example.com:/chemin/vers/repository
% export CVSROOT
```

On procède ensuite avec les commandes normales de récupération de fichiers.

A completer...

A completer...

Chapitre 8

Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *concatenate*)
Syntaxe : **cat fichiers**

Par exemple :

```
% cat exemple.txt
This system is for the use of authorized users only. Individuals
using this computer system without authority, or in excess of
their authority, are subject to having all of their activities
on this system monitored and recorded by system personnel.
```

Commande « `type.exe` »

(en anglais *more*)
En cas de texte très long, la commande « `cat` » n'est pas pratique. On lui préférera la commande « `more` » pour son affichage page d'écran par page d'écran.

Syntaxe : **`more fichiers`**

- Caractère « `q` » pour quitter (en anglais *quit*)
- Caractère espace pour avancer d'une page d'écran
- Caractère « `b` » pour revenir en arrière d'une page (en anglais *backward*)
- Caractère « `f` » pour avancer d'une page d'écran (en anglais *forward*)

La commande « `man` » affiche en fait les pages du manuel au moyen de la commande « `more` »

Lorsque le texte est affiché, on peut rechercher un mot au sein du texte.

Pour chercher « ananas » vers l'avant, faire « `/ananas` ».

Pour chercher « ananas » vers l'arrière, faire « `?ananas` ».

Pour chercher l'apparition suivante du mot, faire « `n` ».

Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.4 • (Windows : : Affichage du contenu d'un fichier texte : `more.exe`)

Commande « `more.exe` »

(en anglais *less*)
La commande « less » possède quelques fonctionnalités agréables de plus que la commande « more ». Par exemple : remonter dans le fichier (à l'origine « more » ne le faisait pas).

Syntaxe : **less fichiers**

Même méthode d'utilisation que pour « more ».

ATTENTION : il existe une variable d'environnement (voir page 596) appelée « LESSOPEN » qui complique le fonctionnement de la commande en fait (sera vu en TP).
⇒ Désactiver la variable en pratique (faire « unset LESSOPEN », voir page 591).

(en anglais *word count*)
Syntaxe : **wc [option] fichiers**

Quelques options intéressantes :

- « -c » : nombre de caractères uniquement (en anglais *character*)
- « -w » : nombre de mots uniquement (en anglais *word*)
- « -l » : nombre de lignes uniquement (en anglais *line*)

Par exemple :

```
% wc exemple.txt
       3      16      82 exemple.txt
% wc -l exemple.txt
       3 exemple.txt
```

(en anglais *difference*)
Syntaxe : **diff [options] fichier1 fichier2**

Objet : réaliser la comparaison ligne à ligne du fichier texte « fichier2 » par rapport au fichier texte « fichier1 ».

Deux options intéressantes :

- option « -c » : affiche de quelques lignes du contexte (en anglais *contextual*)
- option « -u » : mode unifié (en anglais *unified*)

◇ Exemple 1 : utilisation sans paramêtre

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% diff fichier1 fichier2
2c2
< Deux fotes d'ortographe ici.
---
> Deux fautes d'orthographe ici.
```

Interprétation de « 2c2 » : la ligne 2 de « fichier2 » est **c**hangée par rapport à la ligne 2 de « fichier1 »

◇ Exemple 2 : décalage de lignes

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% cat fichier2
1
2 Blabla bla bla.
3 Deux fautes d'ortographe ici.
4 Encore du blabla bla bla.
```

```
% diff fichier1 fichier2
0a1
>
2c3
< Deux fotes d'ortographe ici.
---
> Deux fautes d'ortographe ici.
```

Interprétation de « `0a1` » : la ligne 1 de « `fichier2` » est **a**joutée par rapport à la ligne 0 de « `fichier1` »
Interprétation de `2c3` : la ligne 3 de « `fichier2` » est **c**hangée par rapport à la ligne 2 de « `fichier1` »

◇ Exemple 3 : option « -c »

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% diff -c fichier1 fichier2
*** fichier1  Sun Sep  9 19:06:13 2001
--- fichier2  Sun Sep  9 19:06:24 2001
***************
*** 2 ****
! Deux fotes d'ortographe ici.
--- 2 ----
! Deux fautes d'orthographe ici.
```

Interprétation de « `*** 2 ****` » et « `--- 2 ----` » :
la ligne 2 de « `fichier2` » est **c**hangée par rapport à la ligne 2 de « `fichier1` ».

◇ Exemple 4 : option « -u »

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.
```

```
% diff -u fichier1 fichier2
--- fichier1    Fri Sep 16 22:05:15 2005
+++ fichier2    Fri Sep 16 22:05:20 2005
@@ -1,3 +1,3 @@
 Blabla bla bla.
-Deux fotes d'ortographe ici.
+Deux fautes d'ortographe ici.
 Encore du blabla bla bla.
```

L'affichage est un peu plus lisible.

Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.8 • Comparaison de deux fichiers binaires : cmp

(en anglais *compare*)
Syntaxe : **cmp fichier1 fichier2**

Objet : réaliser la comparaison octet à octet du fichier binaire « fichier2 » par rapport au fichier binaire « fichier1 ».
Utilité par exemple : comparer des binaires d'un système à la recherche de programmes piratés, etc.

Exemple :

```
% cmp programme1.exe programme2.exe
programme1.exe programme2.exe differ: char 3174, line 9
```

Moralité : les deux fichiers sont différents

(en anglais *head*)
Syntaxe : **head [-nombre] fichier**

◇ Exemple : utilisation intéressante

```
% head -3 /etc/motd
SunOS Release 4.1.4 (EXCALIBUR.LPS.ENS.FR [1.1]): Fri Aug 8 17:43:56 GMT 1997
 This system is for the use of authorized users only. Individuals using
 this computer system without authority, or in excess of their authority,
```


(en anglais *tail*)
Plusieurs syntaxes :

- **tail [-nombre] fichier** : les N dernières lignes
- **tail [+nombre] fichier** : de la Nième ligne à la fin du fichier
- **tail [-f] fichier** : affichage « en live »

◇ Exemple : utilisation intéressante

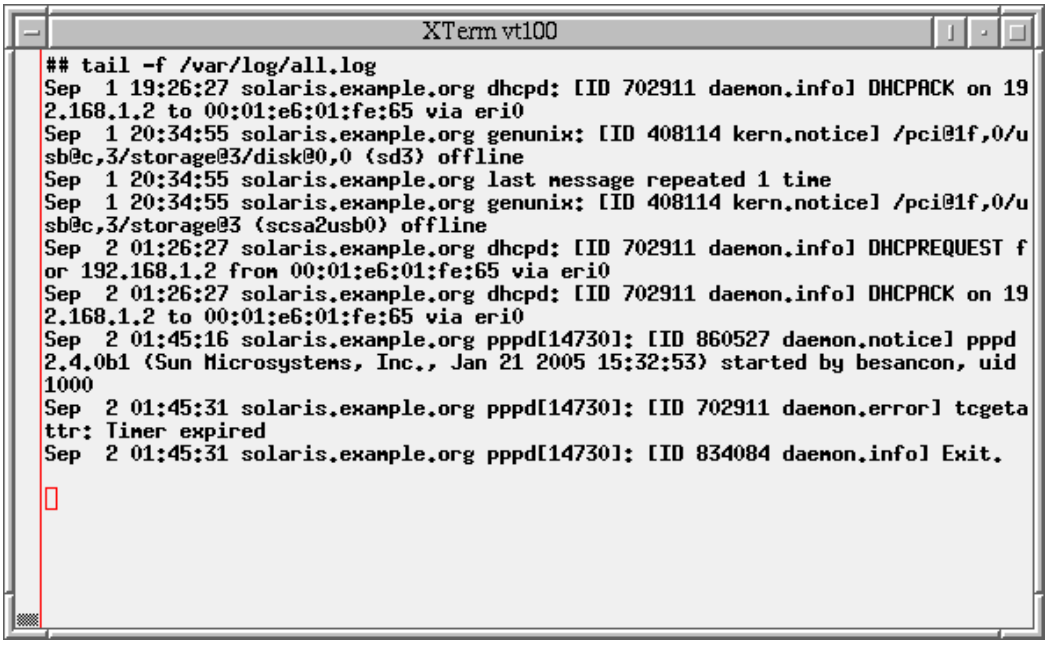
```
% tail -3 /etc/motd
 advised that if such monitoring reveals possible evidence of criminal
 activity, system personnel may provide the evidence of such monitoring
 to law enforcement officials.
```

◇ Exemple 3 : très pratique

```
% tail +3 /etc/motd
this computer system without authority, or in excess of their authority,
are subject to having all of their activities on this system monitored
and recorded by system personnel.

In the course of monitoring individuals improperly using this system, or
in the course of system maintenance, the activities of authorized users
may also be monitored.

Anyone using this system expressly consents to such monitoring and is
advised that if such monitoring reveals possible evidence of criminal
activity, system personnel may provide the evidence of such monitoring
to law enforcement officials.
```


◇ Exemple 4 : Affichage des dernières lignes en temps réel

Syntaxe : **`tail -f fichier`**



Quitter en faisant Ctrl-C (voir page 522).

(en anglais *cut*)
Syntaxe : **cut [options] fichiers**

Quelques options :

- « -c » : découpage selon des positions de caractères (en anglais *character*)
- « -f » : découpage selon des positions de mots (en anglais *field*)
- « -d » : indique le délimiteur de mots (en anglais *delimiter*)

Exemples :

- extraction sur chaque ligne de caractères pris isolément :

```
cut -c 1,8,27 fichier.txt
```

- extraction sur chaque ligne de caractères d'une position 1 à une position 2 :

```
cut -c 25-42 fichier.txt
```

- extraction sur chaque ligne (constituée de mots séparés par un certain délimiteur) de mots pris isolément :

```
cut -d: -f 1,5 fichier.txt
```

- extraction sur chaque ligne (constituée de mots séparés par un certain délimiteur) du mot *i* au mot *j* :

```
cut -d: -f 4-7 fichier.txt
```

(en anglais *sort*)
Syntaxe : **sort [options] fichiers**

Quelques options :

- option « -n » : tri numérique (en anglais *numerical*)
- option « -r » : tri par ordre décroissant (en anglais *reverse*)
- option « -t » : permet de spécifier le séparateur de mots
- option « -k » : spécifie la clef de tri (en anglais *key*) ; on peut indiquer plusieurs clefs de tri

◇ Exemple 1

```
% cat exemple.txt
arbre
12
ascenceur
2
ordinateur
```

```
% sort exemple.txt
12
2
arbre
ascenceur
ordinateur
```

◇ Exemple 2

```
% cat exemple.txt
12
2
3
33
22

% sort exemple.txt
12
2
22
3
33
```

```
% sort -n exemple.txt
2
3
12
22
33

% sort -rn exemple.txt
33
22
12
3
2
```


◇ Exemple 3

```
% cat exemple.txt
or:100000
argent:40000
bois:15

% sort exemple.txt
argent:40000
bois:15
or:100000
```

```
% sort -t : -k 2 exemple.txt
or:100000
bois:15
argent:40000

% sort -t : -k 2 -n exemple.tx
bois:15
argent:40000
or:100000
```

◇ Exemple 4 : trier des adresses IP

Soit le fichier à trier :

```
192.168.1.1     ananas.example.com
192.168.1.4     poire.example.org
127.0.0.1       localhost
192.168.1.3     cerise.example.org
134.157.46.129  serveur.formation.jussieu.fr
192.168.1.100   kiwi.example.org
192.168.1.2     banane.example.org
192.168.2.1     freebox.example.org
```

On a donc :

```
% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 -k 4n,4 exemple.txt
127.0.0.1       localhost
134.157.46.129  serveur.formation.jussieu.fr
192.168.1.1     ananas.example.com
192.168.1.2     banane.example.org
192.168.1.3     cerise.example.org
192.168.1.4     poire.example.org
192.168.1.100   kiwi.example.org
192.168.2.1     freebox.example.org
```


Importance de bien préciser les clefs de tri :

```
% cat exemple.txt
2.2.3.0
2.10.20.0
2.10.3.0
10.2.2.0
2.10.100.0
2.10.100.5
2.10.100.43
2.10.10.8

% sort exemple.txt
10.2.2.0      <- mal classé
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5    <- mal classé
2.10.20.0     <- mal classé
2.10.3.0      <- mal classé
2.2.3.0       <- mal classé
```

```
% sort -t . exemple.txt
10.2.2.0      <- mal classé
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5    <- mal classé
2.10.20.0     <- mal classé
2.10.3.0      <- mal classé
2.2.3.0       <- mal classé

% sort -t . -k 1n,1 exemple.txt
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5    <- mal classé
2.10.20.0     <- mal classé
2.10.3.0      <- mal classé
2.2.3.0       <- mal classé
10.2.2.0
```

Importance de bien préciser les clefs de tri (suite) :

```
% sort -t . -k 1n,1 -k 2n,2 exemple.txt
2.2.3.0
2.10.10.8
2.10.100.0
2.10.100.43  <- mal classé
2.10.100.5
2.10.20.0    <- mal classé
2.10.3.0     <- mal classé
10.2.2.0

% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 exemple.txt
2.2.3.0
2.10.3.0
2.10.10.8
2.10.20.0
2.10.100.0
2.10.100.43  <- mal classé
2.10.100.5
10.2.2.0
```


Importance de bien préciser les clefs de tri (suite) :

La solution pour trier ces adresses IP :

```
% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 -k 4n,4 exemple.txt
2.2.3.0
2.10.3.0
2.10.10.8
2.10.20.0
2.10.100.0
2.10.100.5
2.10.100.43
10.2.2.0
```

```
C:\documents and settings\besancon\mes documents>sort /?
SORT [/R] [/+n] [/M kilo-octets] [/L locale] [/RE octets_enregistrement]
  [[lecteur1:][chemin1]nom_fichier1] [/T [lecteur2:][chemin2]]
  [/O [lecteur3:][chemin3]nom_fichier3]
```

Point de vue unixien sur le sort de Windows :

- considération préhistorique de la place mémoire
- pas de possibilité de tri sur des champs mais uniquement sur des positions de caractères
- pas de tri numérique

(en anglais *uniq*)
Syntaxe : **uniq [options] fichier**

Objet : élimine les lignes **consécutives** redondantes

Quelques options :

- option « -c » : précéde chaque ligne du résultat du nombre d'occurences de cette ligne dans le fichier original (en anglais *count*)

◇ Exemple 1

```
% cat exemple.txt
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
```

```
% uniq exemple.txt
Ceci est un test.
TEST
unix
TEST
```

◇ Exemple 2

```
% cat exemple.txt
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
```

```
% uniq -c exemple.txt
   2 Ceci est un test.
   1 TEST
   1 unix
   2 TEST
```


◇ Exemple 3

```
% cat exemple.txt
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
```

```
% sort exemple.txt | uniq -c
   2 Ceci est un test.
   3 TEST
   1 unix
```

Cette syntaxe « | » sera revue page 570.

(en anglais *touch*)
Syntaxe : **touch fichiers**

ATTENTION : cette commande ne permet de créer un fichier vide que si le fichier mentionné n'existe pas déjà !

◇ Exemple

```
% ls -l exemple.txt
exemple.txt: No such file or directory

% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars          0 Sep 27 12:58 exemple.txt
```


(en anglais *touch*)
Syntaxe : **touch [options] -t time objet**

Un objet UNIX a **trois** dates parmi ses attributs (ce sera revu plus loin).
Quelques options :

- option « `-a` » : modification de la date d'accès du fichier (a ≡ accesstime)
- option « `-m` » : modification de la date de modification du fichier (m ≡ mtime)
- option « `-t time` » : indique une date à mettre autre que la date de l'instant ;
  format : `AAAAMMJJhhmm.ss`

◇ Exemples

```
% ls -l exemple.txt
-rw-r--r--   1 besancon ars        49 Sep 27 13:07 exemple.txt

% touch -m -t 199901012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars        49 Jan  1  1999 exemple.txt

% touch -m -t 09012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars        49 Sep  1 22:33 exemple.txt
```


(en anglais *temporary*)
Le répertoire `/tmp` sert à stocker des objets temporaires.

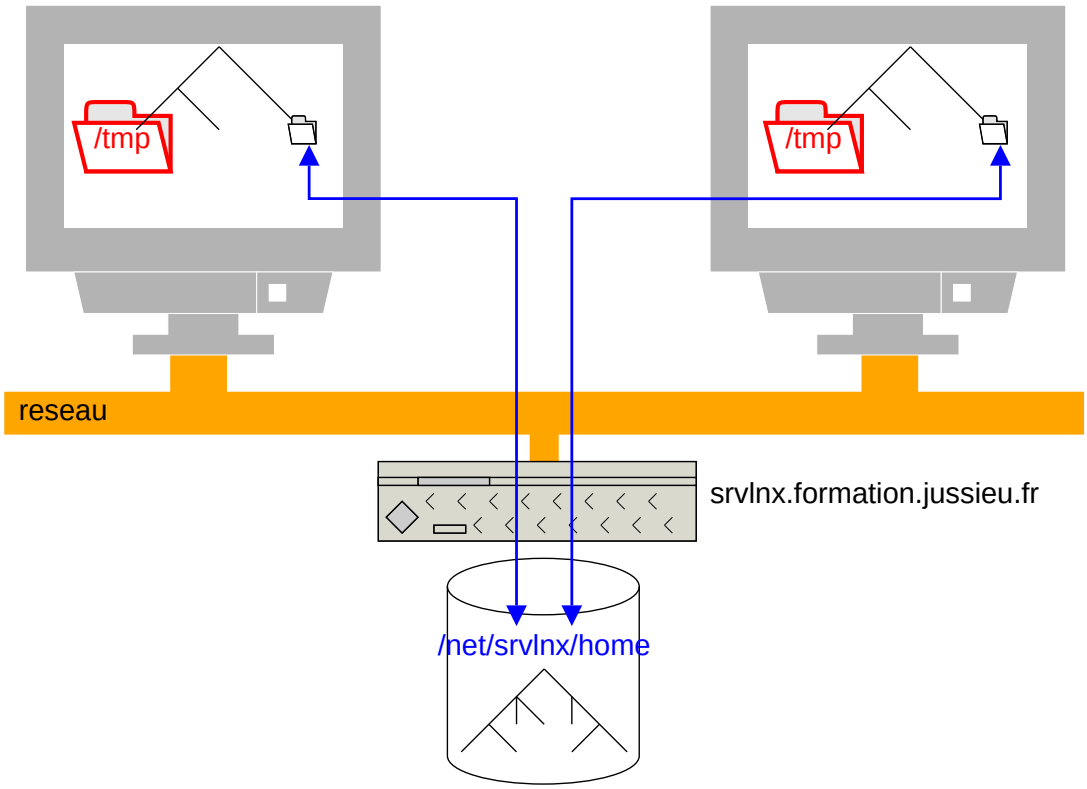
```
% ls -ld /tmp
drwxrwxrwt  12 root     sys         2648 Sep 28 13:02 /tmp
```

Les droits d'accès de `/tmp` sont **1777** exprimé en octal. Leurs significations :

- signification de 777 : tout le monde sur la machine peut créer, modifier, effacer des objets
- signification de 1000 : un utilisateur ne peut effacer que les objets qui lui appartiennent

Ces droits d'accès seront revus et expliqués page 368.

ATTENTION : le répertoire « /tmp » est **local** à chaque machine :



Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.18 • (Windows : : variable temp, répertoire temp)

Il existe sous Windows :

- répertoire public « C:\winnt\temp » sous Windows 2000
- répertoire public « C:\windows\temp » sous Windows XP et Windows 2003
- pour chaque utilisateur d'un Windows 2000 server ou mieux, répertoire temporaire donné par la variable « %temp% », en fait « C:\Documents and settings\utilisateur\local settings\temp\ »

(en anglais *base of name*)
Syntaxe : **basename fichier [suffixe]**

Objet : supprimer un suffixe (extension) du nom d'un objet

```
% basename document.doc .doc
document
```


◇ Utilisation archi classique :

Changer l'extension « .txt » de tous les fichiers du répertoire courant en l'extension « .doc » :

```
for i in *.txt
do
  mv $i `basename $i .txt`.doc
done
```

(la syntaxe de cet exemple sera revue et expliquée page 706).

Exemple à connaitre !

(en anglais *file*)
Syntaxe : **file objets**

Cette commande permet de deviner à quelle application est lié l'objet. Elle s'appuie pour rendre un avis sur la reconnaissance de motifs connus dans le contenu de l'objet.

Pour voir les motifs, se reporter au fichier « /etc/magic »

Version améliorée de la commande (l'amélioration porte sur le nombre de types de fichiers reconnus) :
ftp://ftp.astron.com/pub/file/file-4.09.tar.gz

Exemple d'utilisation :
Soit des fichiers pris dans le cache d'un navigateur web
De quels types sont ces fichiers aux noms sans extension ?

```
% ls -l
total 720110
-rw-------   1 besancon ars      23185 Dec 18 18:41 C3866435d01
-rw-------   1 besancon ars  357515264 Dec 18 18:33 DF0D61DCd01
-rw-------   1 besancon ars    2306816 Dec 18 18:41 _CACHE_001_
-rw-------   1 besancon ars    2670592 Dec 18 18:41 _CACHE_002_
-rw-------   1 besancon ars     135168 Dec 18 18:01 _CACHE_MAP_
```

```
% file *
C3866435d01: HTML document text
DF0D61DCd01: Zip archive data, at least v2.0 to extract
_CACHE_001_: MP32, Mono
_CACHE_002_: data
_CACHE_MAP_: pfm?
```

(en anglais *octal dump*)
Syntaxe : **od [options] objet**

Principales options (cumulables) :

- option « -c » : affichage en ascii (plus pseudo codes C)
- option « -b » : affichage en base 8
- option « -x » : affichage en base 16

Quelques exemples non exhaustifs (2) :

```
% cat exemple.txt
abcde

% od -b exemple.txt
0000000 141 142 143 144 145 012
0000006

% od -c exemple.txt
0000000   a   b   c   d   e  \n
0000006

% od -x exemple.txt
0000000 6162 6364 650a
0000006
```

(en anglais *translate*)
Commande de base sur tous les UNIX.
Syntaxe : **tr [options] jeu1 [jeu2]**

Deux utilisations possibles de la commande :

- remplacer un par un chaque caractère du jeu 1 par le caractère en même position dans le jeu 2
- effacer les caractères du jeu 1 (pas de jeu2 mentionné)

On notera qu'aucun nom de fichiers n'est à donner sur la ligne de commande.

Soit le fichier contenant les lignes suivantes :

```
toto
cheval
```

◇ Exemple 1

On veut convertir les lettres o en lettres e :

```
% tr 'o' 'e' < exemple.txt
tete
cheval
```

◇ Exemple 2

On veut convertir les lettres minuscules en lettres majuscules :

```
% tr '[a-z]' '[A-Z]' < exemple.txt
TOTO
CHEVAL
```

◇ Exemple 3

Meilleure façon de convertir les lettres minuscules en lettres majuscules (ou vice versa) :

```
% cat exemple.txt
aoieàôùéêè

% tr '[:lower:]' '[:upper:]' < exemple.txt
AOIEÀÔÙÉÊÈ
```


Nouvelles écritures :

- « [:lower:] » : en anglais *lower case* : lettres minuscules
- « [:upper:] » : en anglais *upper case* : lettres majuscules
- « [:digit:] » : chiffres de la base 10
- « [:xdigit:] » : chiffres de la base 16
- « [:punct:] » : signes de ponctuation
- « [:alnum:] » : XXX
- « [:alpha:] » : XXX
- « [:graph:] » : XXX
- « [:print:] » : XXX
- « [:blank:] » : XXX
- « [:space:] » : XXX

ATTENTION : Cela nécessite la variable d'environnement LC_CTYPE positionnée à une valeur correcte (voir page 596 pour les variables d'environnement) :

```
% echo $LC_CTYPE
en_US
% tr '[:lower:]' '[:upper:]' < exemple.txt
AOIEÀÔÙÉÊÈ

% unset LC_CTYPE
% tr '[:lower:]' '[:upper:]' < exemple.txt
AOIEàôùéêè
```


(en anglais *disk filesystems*)
Syntaxe : **df [options] répertoires**

Quelques options intéressantes :

- « -k » : affichage des capacités en kilo octets
- « -i » : affichage des capacités en inodes (sur Solaris faire « -o i »)
- « -h » : affichage sous forme plus lisible (en anglais *human readable*) ; affichage variable ⇒ peu utilisable en programmation

Si l'on indique un répertoire, la commande affiche le remplissage du disque dur (local ou réseau) contenant ce répertoire.

Exemple pris sur une machine du réseau de la Formation Permanente :

```
% df -k
Filesystem      1k-blocks    Used Available Use% Mounted on
/dev/hda1         1143208  693050    391091  64% /
serveur:/net/serveur/home
                  1015695  783819    170935  82% /.automount/serveur/net/
serveur/home
serveur:/var/mail  246167   84838    136713  38% /.automount/serveur/var/
mail
```


Exemple pris sur une machine Solaris :

```
% df
/                  (/dev/dsk/c0t0d0s0 ):21867748 blocks  2209818 files
/devices           (/devices          ):       0 blocks        0 files
/system/contract   (ctfs              ):       0 blocks 2147483620 files
/proc              (proc              ):       0 blocks     9892 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/etc/svc/volatile  (swap              ): 3186528 blocks    90949 files
/system/object     (objfs             ):       0 blocks 2147483450 files
/dev/fd            (fd                ):       0 blocks        0 files
/tmp               (swap              ): 3186528 blocks    90949 files
/var/run           (swap              ): 3186528 blocks    90949 files
/extra             (/dev/dsk/c0t0d0s7 ):79432732 blocks  6870541 files
/entrepot          (/dev/dsk/c0t2d0s2 ):11126010 blocks  6415734 files
/users             (/entrepot/users   ):11126010 blocks  6415734 files
/src               (/entrepot/src     ):11126010 blocks  6415734 files
/ars               (/entrepot/projets/ars):11126010 blocks  6415734 files
```

On notera le type d'affichage sans l'option « `-k` ».

Exemple pris sur une machine Solaris :

```
% df -k
Filesystem             kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0    20648041 9714167 10727394    48%    /
/devices                   0       0       0     0%    /devices
ctfs                       0       0       0     0%    /system/contract
proc                       0       0       0     0%    /proc
mnttab                     0       0       0     0%    /etc/mnttab
swap                 1594208     952 1593256     1%    /etc/svc/volatile
objfs                      0       0       0     0%    /system/object
fd                         0       0       0     0%    /dev/fd
swap                 1685728   92472 1593256     6%    /tmp
swap                 1593312      56 1593256     1%    /var/run
/dev/dsk/c0t0d0s7    56091807 16375441 39155448    30%    /extra
/dev/dsk/c0t2d0s2    57708710 52145705 4985918     92%    /entrepot
/entrepot/users      57708710 52145705 4985918     92%    /users
/entrepot/src        57708710 52145705 4985918     92%    /src
/entrepot/projets/ars
                     57708710 52145705 4985918     92%    /ars
```


Exemple pris sur une machine Solaris :

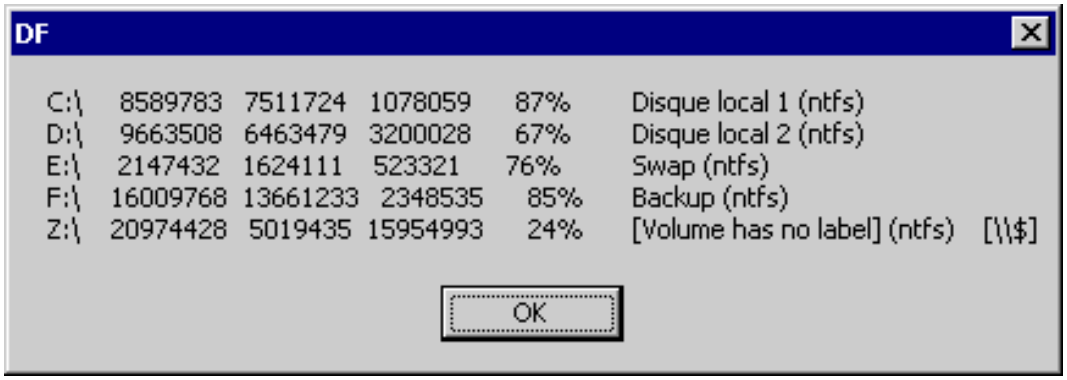
```
% df -o i
df: operation not applicable for FSType autofs
df: operation not applicable for FSType ctfs
df: operation not applicable for FSType devfs
df: operation not applicable for FSType fd
df: operation not applicable for FSType lofs
df: operation not applicable for FSType mntfs
df: operation not applicable for FSType objfs
df: operation not applicable for FSType proc
df: operation not applicable for FSType tmpfs
Filesystem              iused   ifree  %iused  Mounted on
/dev/dsk/c0t0d0s0      324582 2209818    13%   /
/dev/dsk/c0t0d0s7        9459 6870541     0%   /extra
/dev/dsk/c0t2d0s2      662665 6415735     9%   /entrepot
```

Récupérer le package `cmdutils.zip` sur le site `http://paulsadowski.com/`. Il contient un binaire `df.exe` montrant disques locaux et réseau :

```
C:\>net use z: \\winserveur\c$
La commande s'est terminée correctement.

C:\>df.exe -b
C:\    8589783    7511724    1078059   87%   Disque local 1 (ntfs)
D:\    9663508    6463479    3200028   67%   Disque local 2 (ntfs)
E:\    2147432    1624111     523321   76%   Swap (ntfs)
F:\   16009768   13661233    2348535   85%   Backup (ntfs)
Z:\   20974428    5020499   15953928   24%   [Volume has no label]
```


Affichage dans fenêtre graphique possible :

```
C:\>df.exe -b -w
```



(en anglais *disk usage*)
Syntaxe : **du [options] répertoires**

Principales options (cumulables) :

- option « -s » : affichage uniquement du total (en anglais *sum*)
- option « -k » : affichage des totaux exprimés en kilo octets (en anglais *kilobytes*)
- option « -h » : affichage sous forme plus lisible (en anglais *human readable*) ;
  affichage variable ⇒ peu utilisable en programmation

Attention : ne pas confondre « df » et « du ».

◇ Exemple 1 :

Affichage récursif avec des messages d'erreur :

```
% du -k picqueno
152     picqueno/.kde/share/config
du: cannot change to directory 'picqueno/.kde/share/fonts': Permission de
du: cannot change to directory 'picqueno/.kde/share/apps': Permission den
du: cannot change to directory 'picqueno/.kde/share/mimelnk': Permission
du: cannot change to directory 'picqueno/.kde/share/services': Permissio
du: cannot change to directory 'picqueno/.kde/share/icons': Permission de
176     picqueno/.kde/share
184     picqueno/.kde
588     picqueno/.mcop/trader-cache
592     picqueno/.mcop
20      picqueno/tp/2
32      picqueno/tp/4
4       picqueno/tp/1
4       picqueno/tp/3
68      picqueno/tp/5
776     picqueno/tp
1672    picqueno
```

◇ Exemple 2 :

Affichage récursif avec suppression des messages d'erreur :

```
% du -k picqueno 2>/dev/null
152     picqueno/.kde/share/config
176     picqueno/.kde/share
184     picqueno/.kde
588     picqueno/.mcop/trader-cache
592     picqueno/.mcop
20      picqueno/tp/2
32      picqueno/tp/4
4       picqueno/tp/1
4       picqueno/tp/3
68      picqueno/tp/5
776     picqueno/tp
1672    picqueno
```

Voir page 573 et page 579 pour l'explication de « 2>/dev/null ».

◇ Exemple 3 :

Affichage du total avec des messages d'erreur :

```
% du -k -s picqueno
du: cannot change to directory 'picqueno/.kde/share/fonts': Permission de
du: cannot change to directory 'picqueno/.kde/share/apps': Permission den
du: cannot change to directory 'picqueno/.kde/share/mimelnk': Permission
du: cannot change to directory 'picqueno/.kde/share/services': Permissio
du: cannot change to directory 'picqueno/.kde/share/icons': Permission de
1672    picqueno
```

◇ Exemple 4 :

Affichage du total avec suppression des messages d'erreur :

```
% du -k -s picqueno 2>/dev/null
1672    picqueno
```

Voir page 573 et page 579 pour l'explication de « 2>/dev/null ».

Vous ne devez pas laisser votre compte se remplir de fichiers. Les disques durs n'ont pas une capacité infinie et hors de question de stocker toute la documentation disponible sur Internet chez vous !

La commande « du -k $HOME » vous donnera la taille disque que votre homedirectory occupe. La commande passe en revue tous les répertoires et en affiche la taille.

Le résultat affiché est exprimé en kilo octets (1 ko = 1024 octets).

La commande « diruse.exe » (resource kit Windows 2000, support kit Windows XP/2003) calcule la place occupée dans une arborescence :

```
C:\>diruse.exe /s /k C:\Docume~1

    Size (kb)  Files  Directory
         0.00      0  \DOCUME~1
         0.00      0  \DOCUME~1\Administrator
         0.00      0  \DOCUME~1\Administrator\Application Data
         0.00      0  \DOCUME~1\Administrator\Application Data\Microsoft
...
         0.02      1  \DOCUME~1\Default User\Start Menu\Programs\Startup
        36.14     12  \DOCUME~1\Default User\Templates
    732377.45   4711  SUB-TOTAL: \DOCUME~1

    732377.45   4711  TOTAL: \DOCUME~1
```


La commande « diskuse.exe » (resource kit 2000/XP/2003) calcule la place occupée dans une arborescence mais la calcule par utilisateur :

```
C:\>diskuse.exe /s /e:nul /t "c:\Documents and settings"
DiskUse                  Version 1.3

Scanning Path .\.....................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................
Resolving Names....
Sorting....

WINXP               Administrator           1653
WINXP               besancon                705532503
WINXP               root                    36876743
BUILTIN             Administrators          6375184
```

Pas d'équivalent immédiat sous UNIX.

Syntaxes de quelques commandes de compression ou décompression :

- **compress [options] fichiers**
- **uncompress [options] fichiers.Z**
- **zcat fichiers.Z**

Le fichier compressé s'appelle après compression « fichier.Z ».
Le fichier décompressé retrouve son nom « fichier ».

◇ Exemples

```
% ls -l 20091029.log
-rw-r--r--  1 besancon ars  44532413 Oct 30 01:41 20091029.log

% compress 20091029.log

% compress -v 20091029.log
20091029.log: Compression: 85.01% -- replaced with 20091029.log.Z

% ls -l 20091029.log.Z
-rw-r--r--  1 besancon ars   6673089 Oct 30 01:41 20091029.log.Z

% zcat 20091029.log.Z
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

Cette série de commandes compresse mieux les fichiers que la famille autour de `compress`.

Syntaxes de quelques commandes de compression ou décompression :

- **`gzip [options] fichiers`**
- **`gunzip [options] fichiers.gz`**
- **`gzcat fichiers.gz`**

Le fichier compressé s'appelle après compression « `fichier.gz` ».
Le fichier décompressé retrouve son nom « `fichier` ».

De plus en plus répandu. Au cas où absent :
`ftp://ftp.lip6.fr/pub/gnu/gzip/`

Attention : dans les salles de TP, il faut utiliser « `zcat` » au lieu de « `gzcat` ».

◇ Exemples

```
% ls -l 20091029.log
-rw-r--r--  1 besancon ars  44532413 Oct 30 01:41 20091029.log

% gzip 20091029.log

% gzip -v 20091029.log
20091029.log:    91.6% -- replaced with 20091029.log.gz

% ls -l 20091029.log.gz
-rw-r--r--  1 besancon ars   3718570 Oct 30 01:41 20091029.log.gz

% gzcat 20091029.log.gz
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

Cette série de commandes compresse mieux les fichiers que la famille autour de `compress`.

Syntaxes de quelques commandes de compression ou décompression :

- **bzip2 [options] fichiers**
- **bunzip2 [options] fichiers.bz2**
- **bzcat fichiers.bz2**

Le fichier compressé s'appelle après compression « `fichier.bz2` ».
Le fichier décompressé retrouve son nom « `fichier` ».

Pas encore très répandu. Cf `http://sources.redhat.com/bzip2/`

◇ Exemples

```
% ls -l 20091029.log
-rw-r--r--  1 besancon ars  44532413 Oct 30 01:41 20091029.log

-CMDUSERbzip2 20091029.log
20091029.log: 17.050:1,  0.469 bits/byte, 94.14% saved, 44532413 in, 2611

% ls -l 20091029.log.bz2
-rw-r--r--  1 besancon ars   2611813 Oct 30 01:41 20091029.log.bz2

% bzcat 20091029.log.bz2
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

Cette série de commandes compresse mieux les fichiers que la famille autour de `compress`.

Syntaxes de quelques commandes de compression ou décompression :

- **`7za a -t7z archive.7z objets`**
- **`7za l archive.7z`**
- **`7za e archive.7z`**

Le fichier compressé s'appelle après compression « `fichier.7z` ».
Le fichier décompressé retrouve son nom « `fichier` ».

Pas encore très répandu. Cf `http://www.7-zip.org/`

◇ Exemples

```
% ls -l 20091029.log
-rw-r--r--  1 besancon ars  44532413 Oct 30 01:41 20091029.log

% 7z a -t7z 20091029.log 20091029.log

7-Zip (A) 9.04 beta  Copyright (c) 1999-2009 Igor Pavlov  2009-05-30
p7zip Version 9.04 (locale=C,Utf16=off,HugeFiles=on,1 CPU)
Scanning

Creating archive 20091029.log.7z

Compressing  20091029.log

Everything is Ok
% ls -l
-rw-r--r--  1 besancon adm  44532413 Oct 30 01:41 20091029.log
-rw-r--r--  1 besancon adm   2734049 Oct 30 09:23 20091029.log.7z
```

```
% ls -l 20091029.log
-rw-r--r--  1 besancon ars  44532413 Oct 30 01:41 20091029.log

% time compress -v 20091029.log
20091029.log: Compression: 85.01% -- replaced with 20091029.log.Z

real    0m15.718s
user    0m13.756s
sys     0m1.024s

% time gzip -v 20091029.log
20091029.log:    91.6% -- replaced with 20091029.log.gz

real    0m8.412s
user    0m7.359s
sys     0m0.471s
```



```
% time bzip2 -v 20091029.log
20091029.log: 17.050:1,  0.469 bits/byte, 94.14% saved, 44532413 in, 2611

real    2m54.250s
user    2m42.890s
sys     0m1.128s

% time 7za a -t7z 20091029.log.7z  20091029.log

7-Zip (A) 9.04 beta  Copyright (c) 1999-2009 Igor Pavlov  2009-05-30
p7zip Version 9.04 (locale=C,Utf16=off,HugeFiles=on,1 CPU)
Scanning

Creating archive 20091029.log.7z

Compressing  20091029.log

Everything is Ok

real    4m58.118s
user    2m1.764s
sys     0m15.259s
```

Algorithme	Gain	Taille finale	Temps
Original		44532413	
compress	85.01 %	6673089	38.28 s
gzip	91.60 %	3718570	8.41 s
bzip2	94.14 %	2611813	174.25 s
7z (autres options)	94.38 %	2502905	20302.67 s

Les utilitaires de compression ne fonctionnent pas comme sur ces photos :

En apparence :

```
% ls -l
-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log

% gzip 20091029.log

% ls -l
-rw-r--r--  1 besancon ars   3718570 Oct 30 00:35 20091029.log.gz
```

En réalité : la compression consomme de la place disque durant la compression car les deux fichiers (fichier de départ non compressé, fichier compressé en cours de génération) existent simultanément pendant quelques instants.

Si l'on observe le répertoire pendant la compression, on voit bien les deux fichiers exister (ici affichage chaque seconde) :

```
-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars    294912 Oct 30 00:42 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars    786432 Oct 30 00:42 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   1212416 Oct 30 00:42 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   1720320 Oct 30 00:43 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   2113536 Oct 30 00:43 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   2572288 Oct 30 00:43 20091029.log.gz
...
```

Suite...

```
...
-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   3080192 Oct 30 00:43 20091029.log.gz

-rw-r--r--  1 besancon ars  44532413 Oct 30 00:35 20091029.log
-rw-------  1 besancon ars   3588096 Oct 30 00:43 20091029.log.gz

-rw-r--r--  1 besancon ars   3718570 Oct 30 00:35 20091029.log.gz
```


Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.34 • Archivage d'objets : `tar`

(en anglais *tape archive*)
La commande `tar` permet d'archiver dans un seul fichier une arborescence.

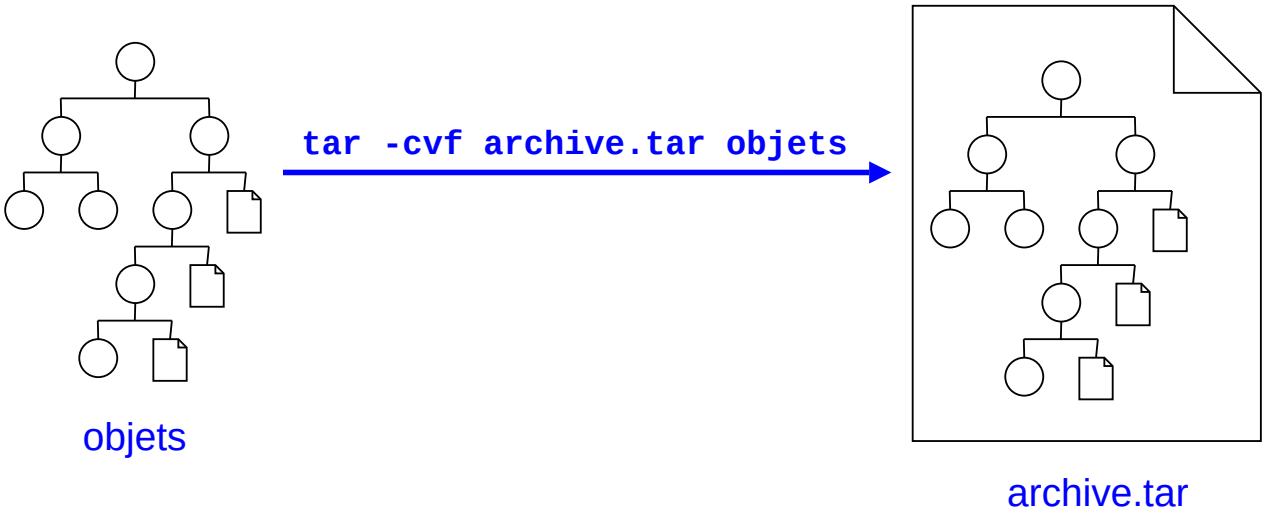
3 syntaxes selon l'opération :

- Création d'une archive
- Affichage du contenu d'une archive
- Extraction d'objets présents dans l'archive

Extension traditionnelle : « `.tar` »

◇ Création d'une archive

Syntaxe : « **tar -cvf archive.tar objets** »



Attention : la commande autorise d'écrire les options sans le signe « – ».
Par exemple on peut rencontrer : « `tar cvf archive.tar objets` ».

◇ Affichage du contenu d'une archive

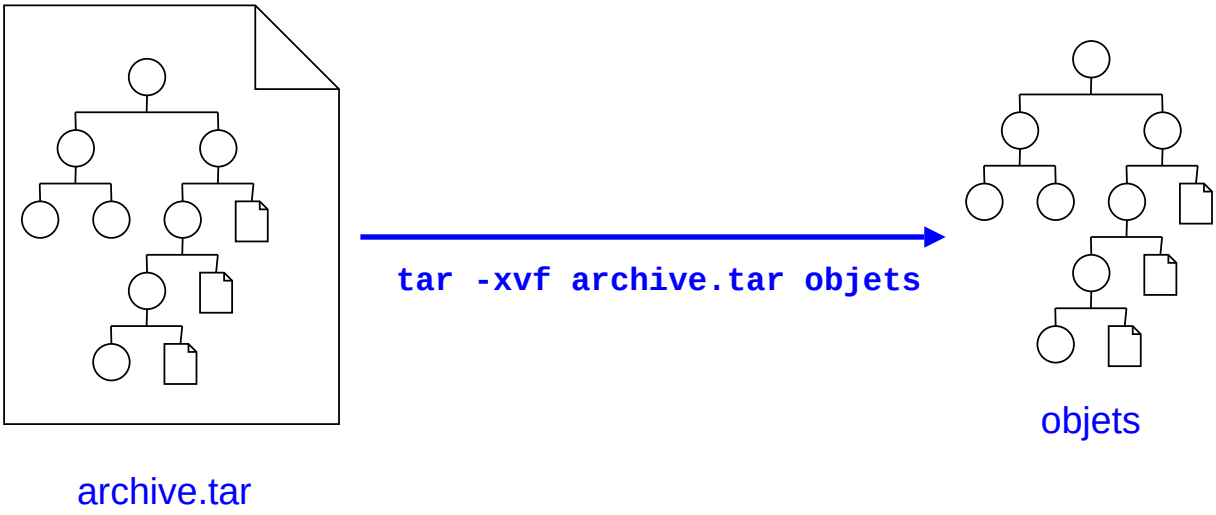
Syntaxe : « **tar -tvf archive.tar objets** »

Attention : la commande autorise d'écrire les options sans le signe « – ».
Par exemple on peut rencontrer : « `tar tvf archive.tar` ».

◇ Extraction depuis l'archive

Syntaxes :

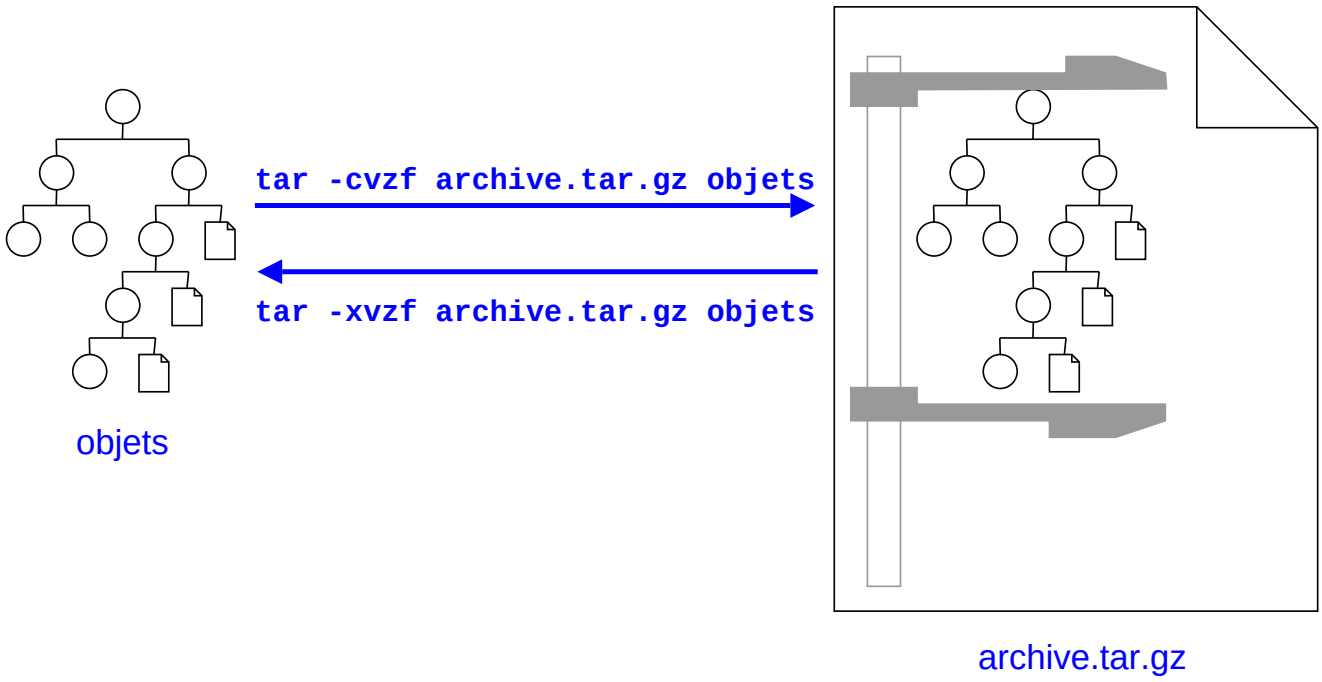
1. Extraction de l'archive complète : « **tar -xvf archive.tar** »
2. Extraction d'un ou plusieurs objets de l'archive : « **tar -xvf archive.tar objets** »



Attention : la commande autorise d'écrire les options sans le signe « - ».
Par exemple on peut rencontrer : « `tar xvf archive.tar objets` ».

◇ Compression à la volée lors de l'archivage

Vous pouvez selon les systèmes UNIX compresser l'archive au fur et à mesure de sa construction :



◇ Archivage + Compression par compress

Option « Z » :

- Création d'une archive :
  « tar -cvZf archive.tar.Z objets »
- Affichage du contenu d'une archive :
  « tar -tvZf archive.tar.Z »
- Extraction de l'archive complète :
  « tar -xvZf archive.tar.Z »
- Extraction d'un ou plusieurs objets de l'archive :
  « tar -xvZf archive.tar.Z objets »

Extension traditionnelle : « .tar.Z »

◇ Archivage + Compression par gzip

Option « z » :

- Création d'une archive :
  « tar -cvzf archive.tar.gz objets »
- Affichage du contenu d'une archive :
  « tar -tvzf archive.tar.gz »
- Extraction de l'archive complète :
  « tar -xvzf archive.tar.gz »
- Extraction d'un ou plusieurs objets de l'archive :
  « tar -xvzf archive.tar.gz objets »

Extension traditionnelle : « .tar.gz »

Un fichier « .tar.gz » s'appelle un « *tarball* ».

Extension « .tgz » courante : équivalent à « .tar.gz »

◇ Archivage + Compression par bzip2

Option « j » :

- Création d'une archive :
  « tar -cvjf archive.tar.bz2 objets »
- Affichage du contenu d'une archive :
  « tar -tvjf archive.tar.bz2 »
- Extraction de l'archive complète :
  « tar -xvjf archive.tar.bz2 »
- Extraction d'un ou plusieurs objets de l'archive :
  « tar -xvjf archive.tar.bz2 objets »

Extension traditionnelle : « .tar.bz2 »

Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.35 • Commandes issues du monde Windows : zip, unzip

Syntaxes de quelques commandes de compression ou décompression :

- **zip [options] fichier**
- **unzip [options] fichier.zip**

URL : http://www.info-zip.org/

◇ Exemple 1 : création d'une archive

```
% zip -r archive.zip fichier1 fichier2
  adding: fichier1 (deflated 63%)
  adding: fichier2 (deflated 66%)
```

◇ Exemple 2 : consultation de la table des matières de l'archive

```
% unzip -l archive.zip
Archive:  archive.zip
 Length     Date    Time    Name
 ------     ----    ----    ----
   3213  10-25-03  01:50   fichier1
  10371  10-25-03  01:50   fichier2
 ------                     -------
  13584                     2 files
```


◇ Exemple 3 : extraction du contenu entier de l'archive

```
% unzip archive.zip
Archive:  archive.zip
  inflating: fichier1
  inflating: fichier2
```

◇ Exemple 4 : extraction du contenu partiel de l'archive

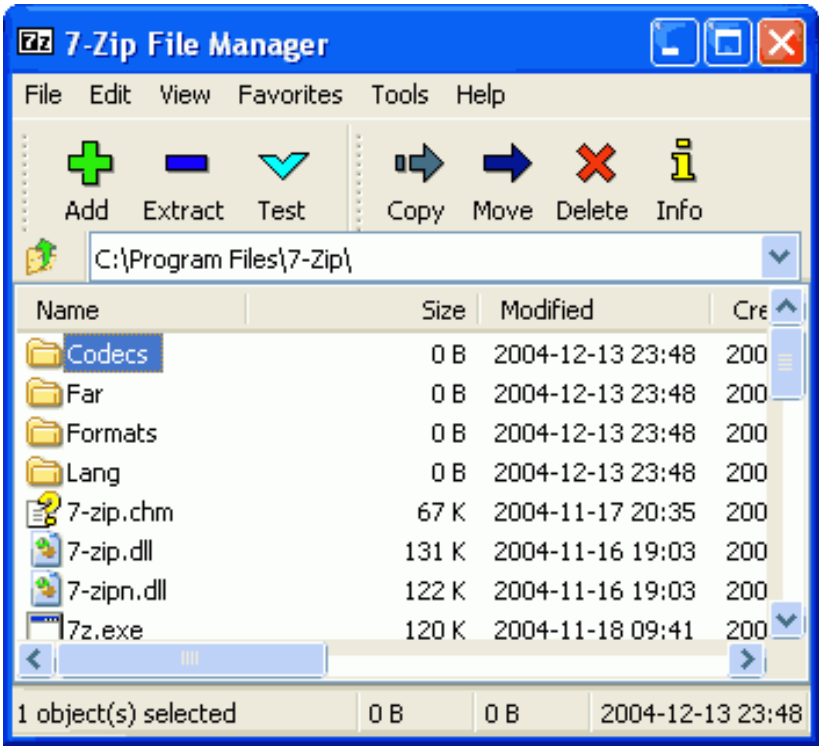
```
% unzip archive.zip fichier2
Archive:  archive.zip
  inflating: fichier2
```

Attention : utiliser l'option « `-a` » pour extraire des fichiers texte avec conversion des fins de ligne de MSDOS vers UNIX

Obsolète : voir 7-ZIP

Version 3.50 gratuite et téléchargeable sur
`http://www.powerarchiver.com/`
Version 4 payante

- « `PAComp.exe` » compresse sous les formats : ZIP, CAB, LHA, BH (BlakHole), JAR (JavaARchiver), TAR, TAR.GZ (GZIPed TAR), TAR.BZ2 (BZIPed TAR)
- « `PAExt.exe` » extrait les formats : ZIP, RAR, ARJ, CAB, LHA(LZH), ARC, ACE, GZIP, BZIP2, TAR (TAR.GZ, TAR.BZ2), UUE, XXE, ZOO, JAR (JavaARchiver) et autres formats auto-extractibles.



(en anglais *line printer*, *line printer queue*, *line printer remove*)

Une impression nécessite de connaitre le nom de l'imprimante et d'avoir un fichier au bon format à imprimer.

Pour imprimer, utiliser la commande « `lpr -Pimprimante fichiers` »

Pour consulter la queue d'impression, utiliser la commande « `lpq -Pimprimante` »

Pour retirer un fichier de la queue d'impression, utiliser la commande « `lprm -Pimprimante numéro-dans-la-queue-renvoyé-par-lpq` »

A la formation permanente, les noms des imprimantes sont :

- pièce 213 : « **hp4250-213** »
- pièce 214 : « **hp4100-214** »
- pièce 216 : « **hp4250-216** »
- pièce 217 : « **hp4250-217** »

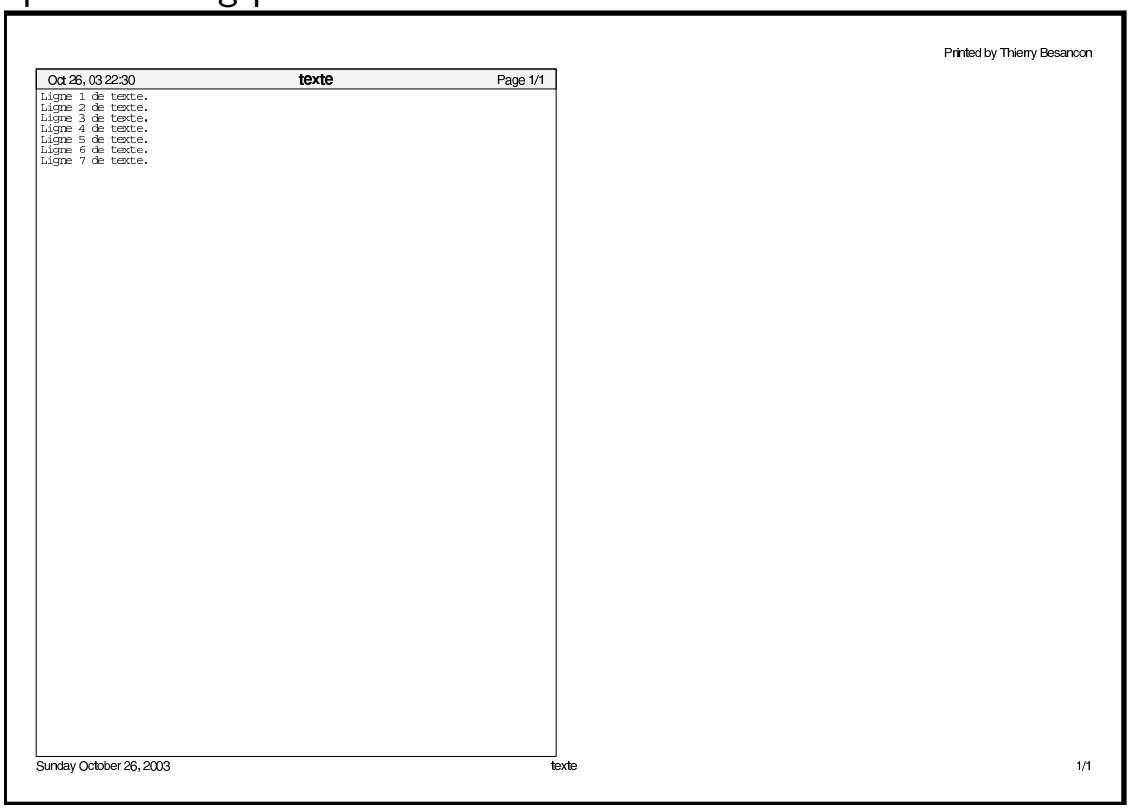
(en anglais *ascii to postscript*)
Pour convertir du texte vers le format PostScript compris par l'imprimante.
Nombreuses options de la commande « a2ps ».

Pour imprimer du texte dans la salle de TP de la Formation Permanente :

```
% a2ps -P hp4250-216 fichier.txt
[a2ps (plain): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] sent to the standard output
```

Disponible à l'URL : http://www.inf.enst.fr/~demaille/a2ps/

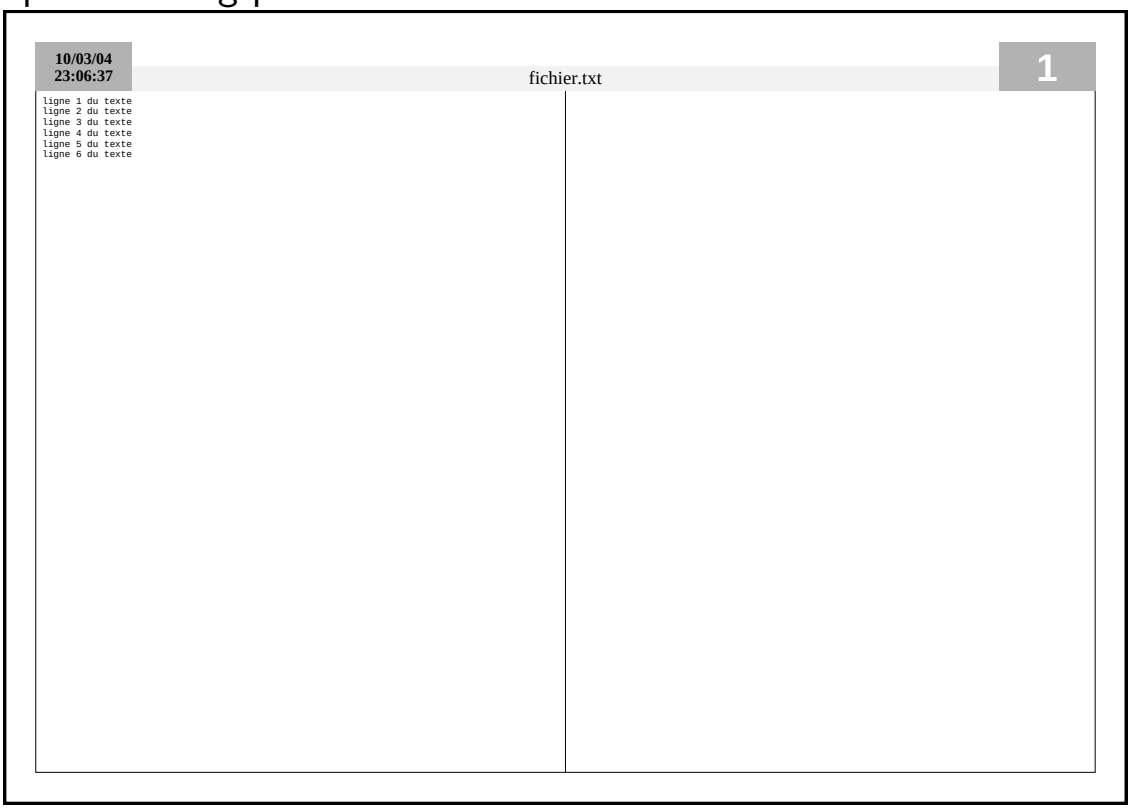
Exemple de listing produit :



(le nom provient d'un logiciel de la marque Adobe aux fonctionnalités reprises par le logiciel GNU qui a repris le nom pour marquer sa compatibilité avec le logiciel original)
Pour convertir du texte vers le format PostScript compris par l'imprimante.
Nombreuses options de la commande `enscript`.

Disponible à l'URL : `http://www.iki.fi/~mtr/genscript/`

Exemple de listing produit :



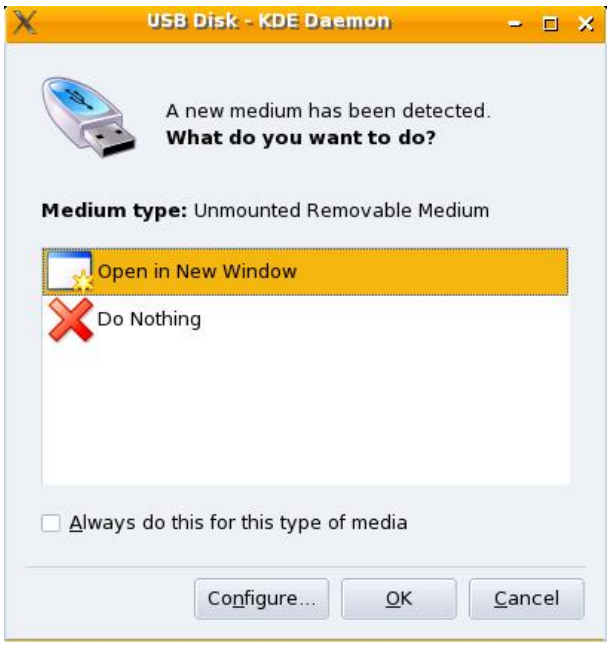
Sur des machines équipées de lecteur de disquettes, on peut transférer des fichiers depuis et vers leur lecteur de disquette. Un logiciel appelé **mtools** permet d'utiliser les disquettes en offrant des commandes UNIX avec la logique des commandes connues du DOS.
Récupérer le logiciel sur `http://mtools.linux.lu`

La commande de base à utiliser est `mcopy`.

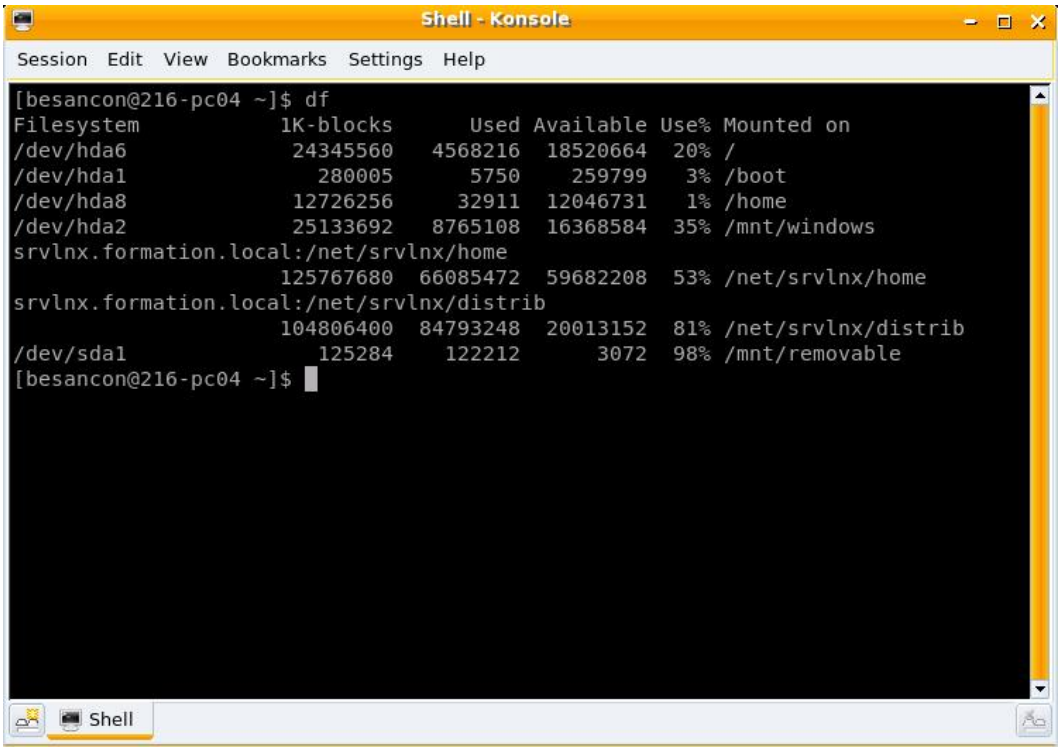
Transfert d'UNIX vers la disquette	`mcopy fichier a:`
Transfert de la disquette vers UNIX	`mcopy a:fichier .`
Affichage du contenu de la disquette	`mdir a:`

Se reporter à l'URL `http://www.loria.fr/~giese/doc/mtools.html` pour plus de détails sur les commandes disponibles.

Les clefs USB sont directement reconnues par le bureau de l'environnement de multifenêtrage KDE de la salle de TP de la Formation Permanente :



La clef apparait en tant qu'icone sur le bureau et en tant que « `/mnt/removable` » dans l'arborescence des fichiers.

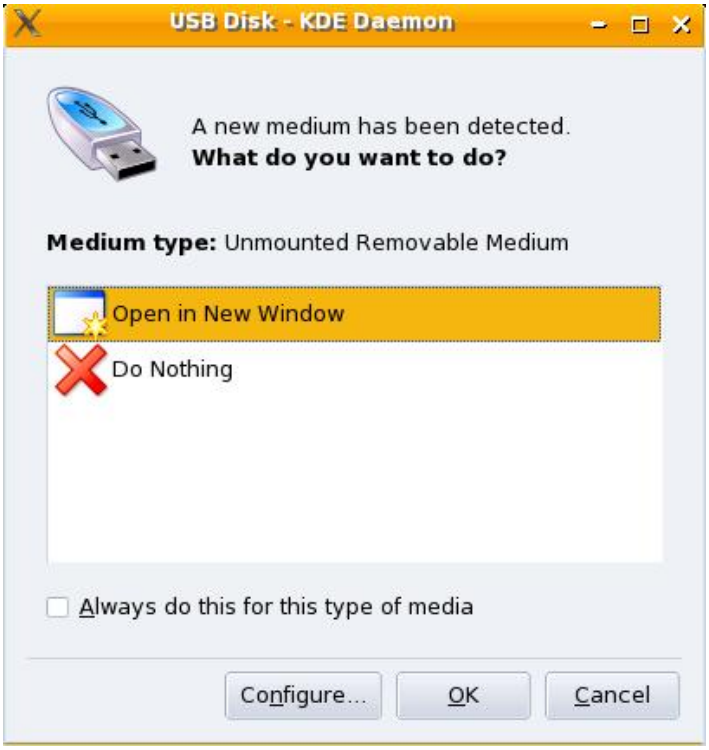


La clef USB doit être formatée en FAT ou FAT32.

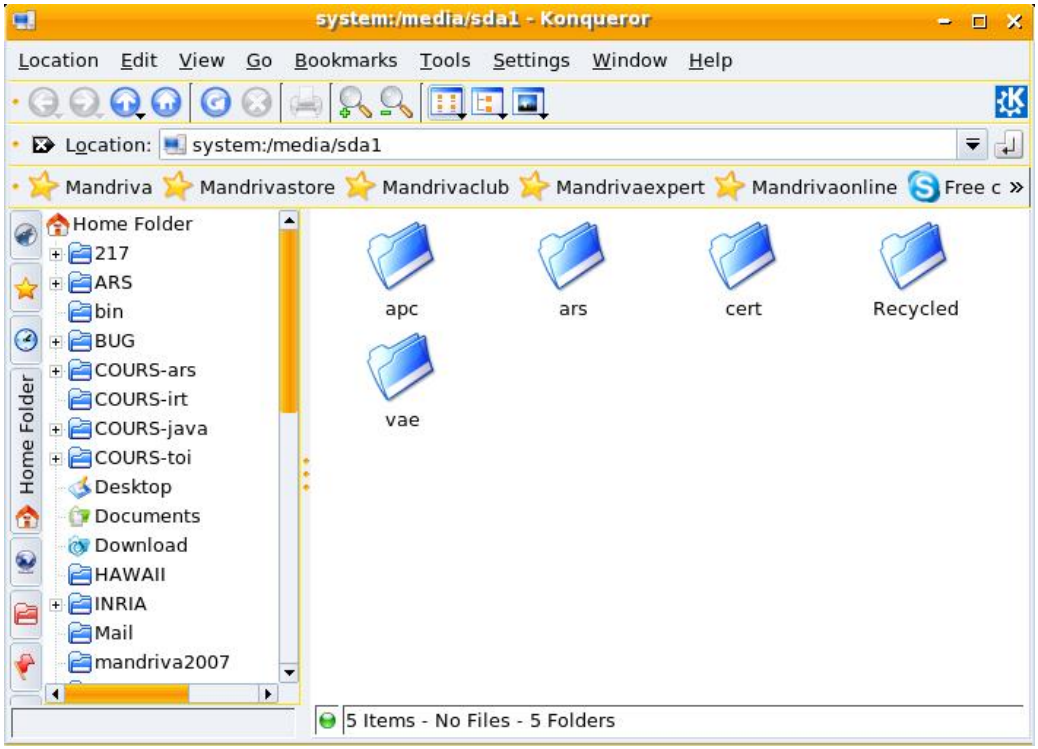
La clef USB doit être formatée en mode disque dur et doit comporter par conséquent une table de partition.

Une clef USB formatée en mode disquette (c'est-à-dire sans table de partition) ne sera pas reconnue par le système LINUX de la salle de TP de la Formation Permanente.

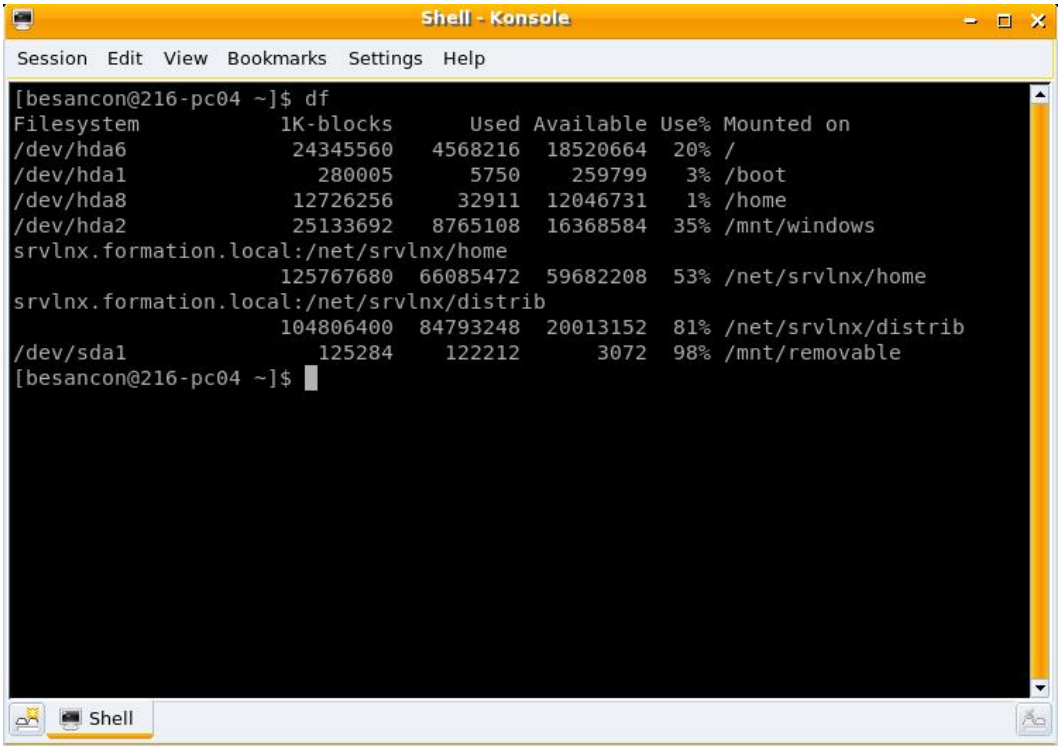
◇ Mandriva 2007 : insertion de la clef



◇ Mandriva 2007 : l'explorateur de fichiers s'ouvre

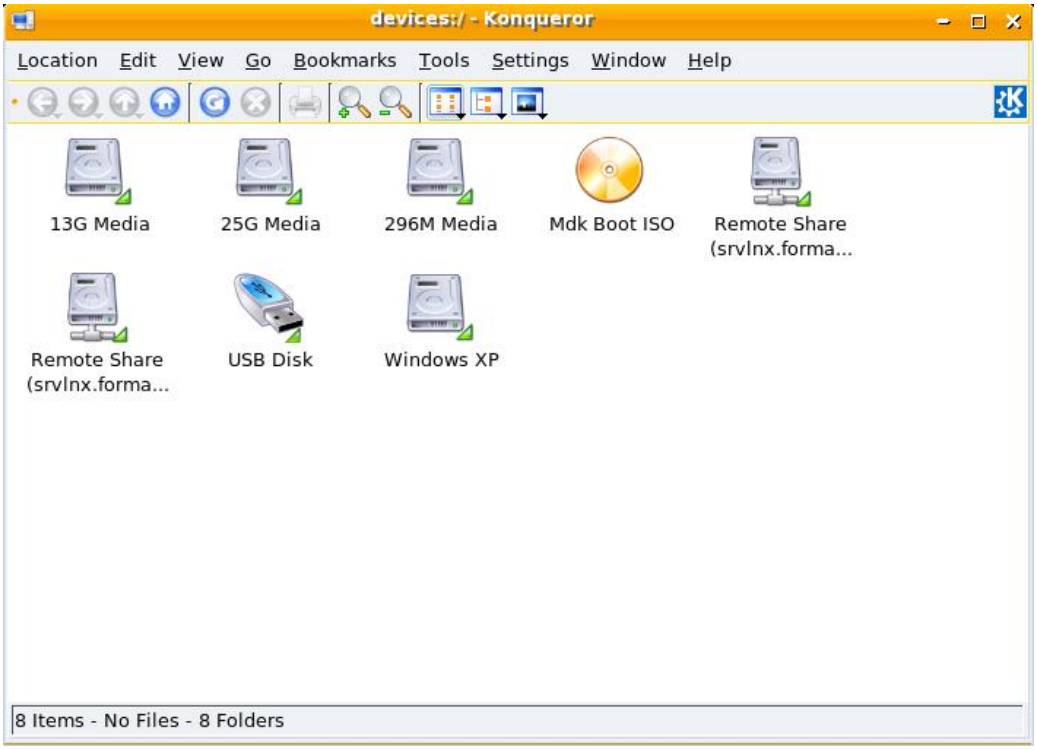


◇ Mandriva 2007 : montage automatique de la clef

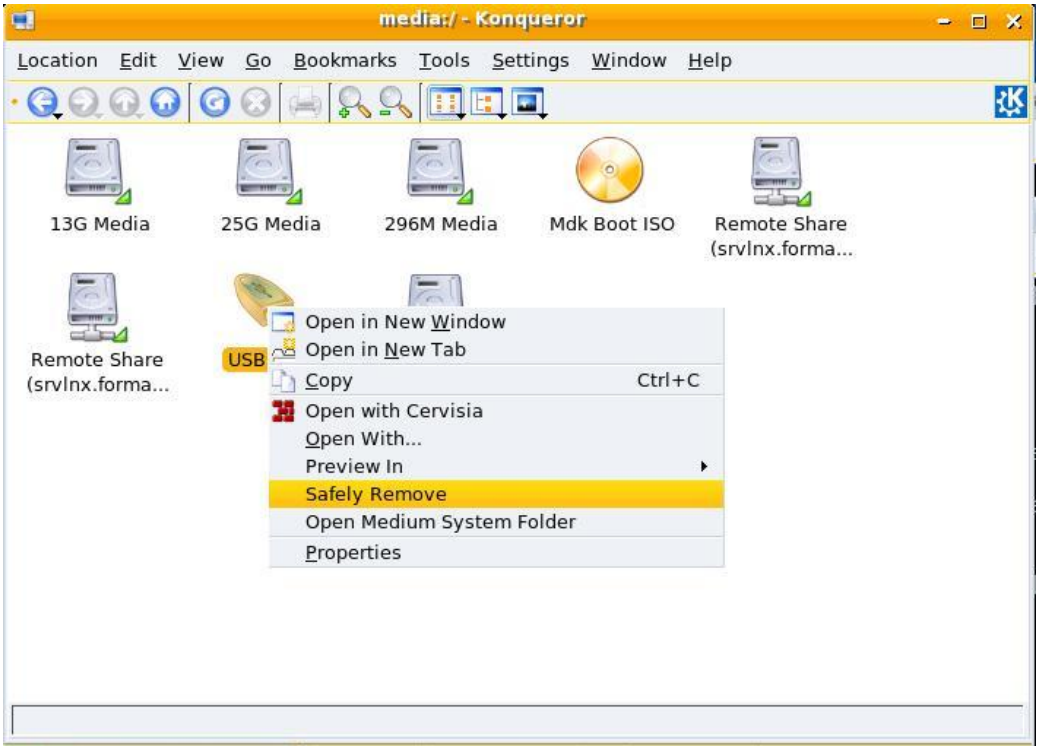


La clef est montée en tant que « `/mnt/removable` ».

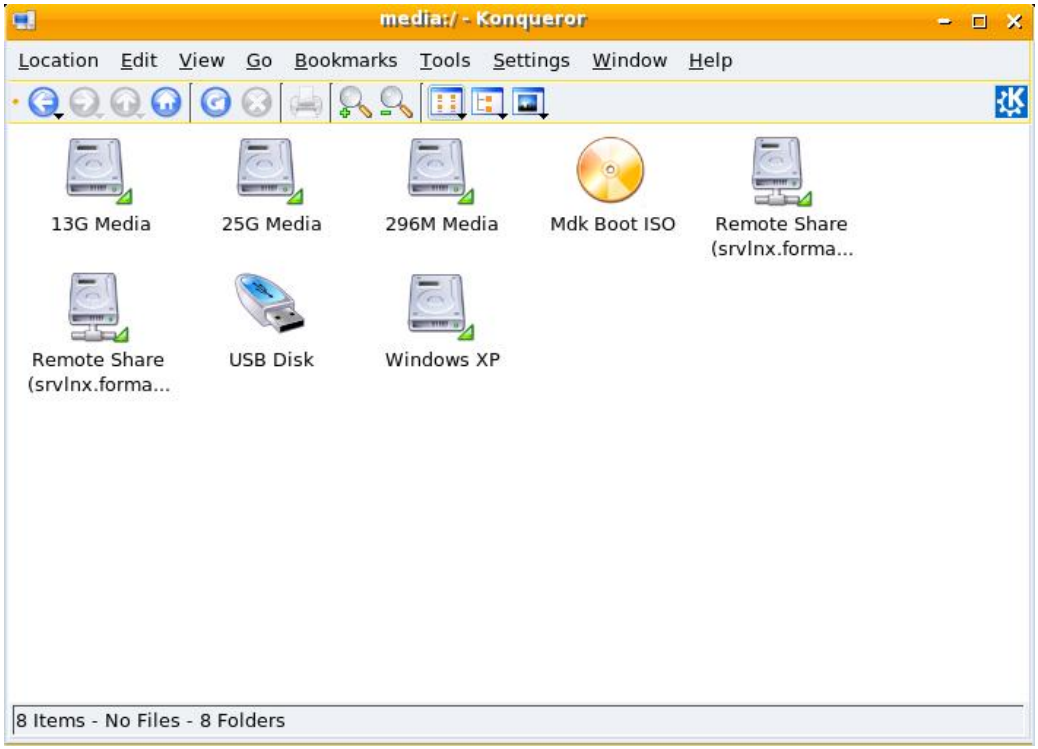
◇ Mandriva 2007 : présence de la clef dans les devices



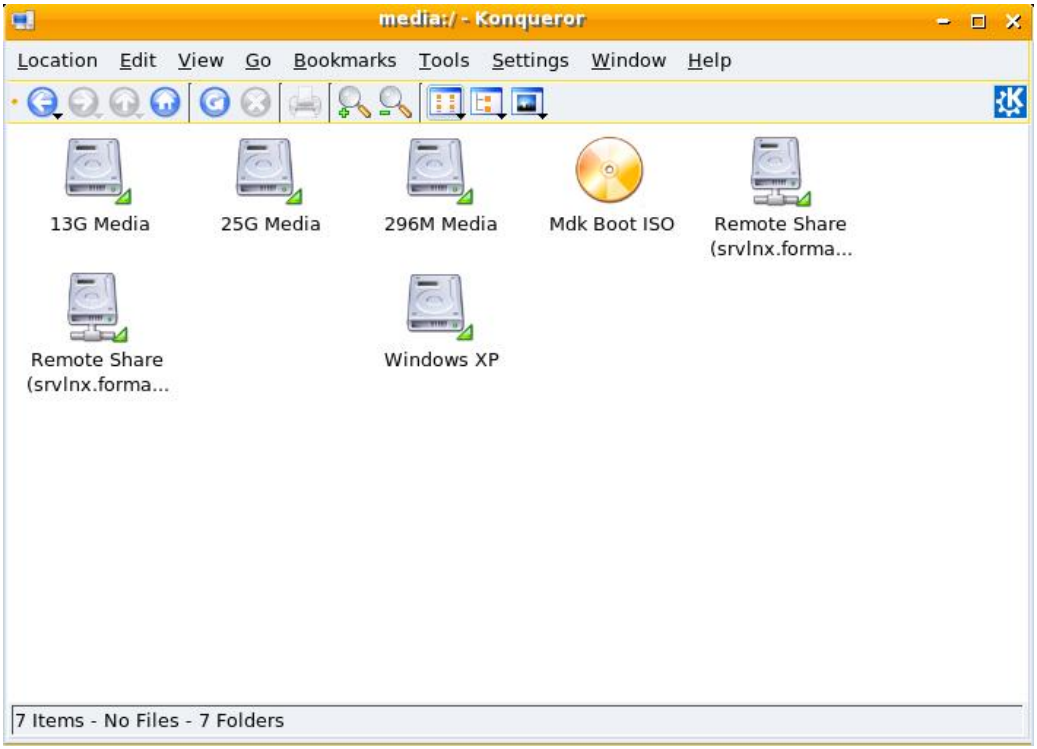
◇ Mandriva 2007 : démontage de la clef (1)



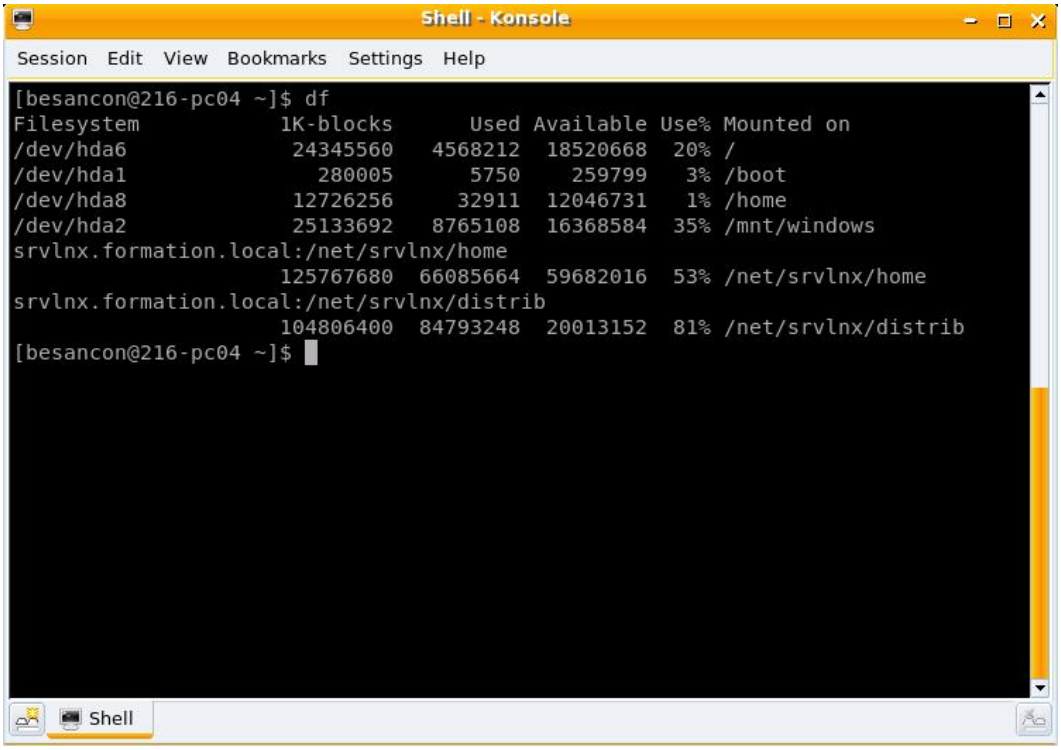
◇ Mandriva 2007 : démontage de la clef (2)



◇ Mandriva 2007 : démontage de la clef (3)



◇ Mandriva 2007 : démontage de la clef (4)



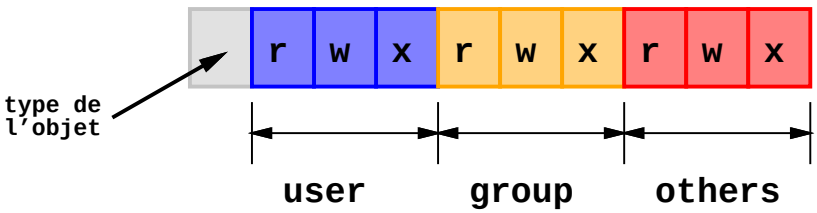
Chapitre 9

Attributs des objets UNIX

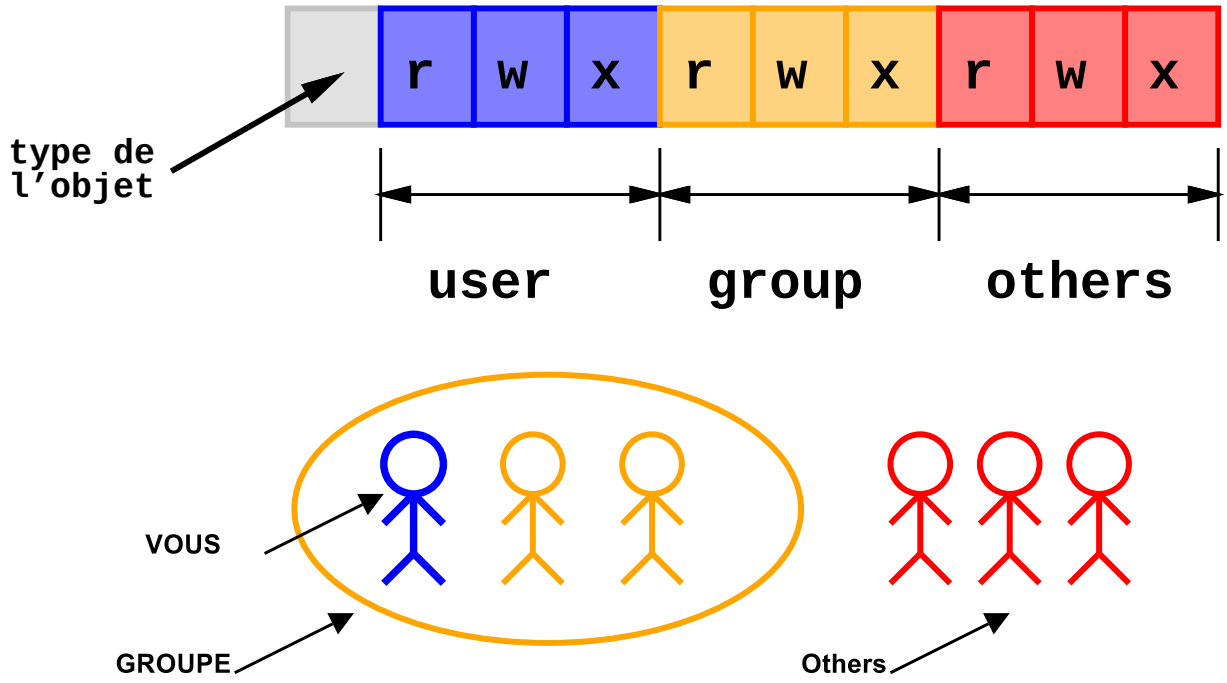
Les droits d'accès à un objet sont stockés dans une structure dite *inode*. Cette structure n'est pas manipulable directement.

Les droits d'accès des objets sont indiqués dans les 10 premiers caractères de chaque ligne affichée par « `ls -l` » :

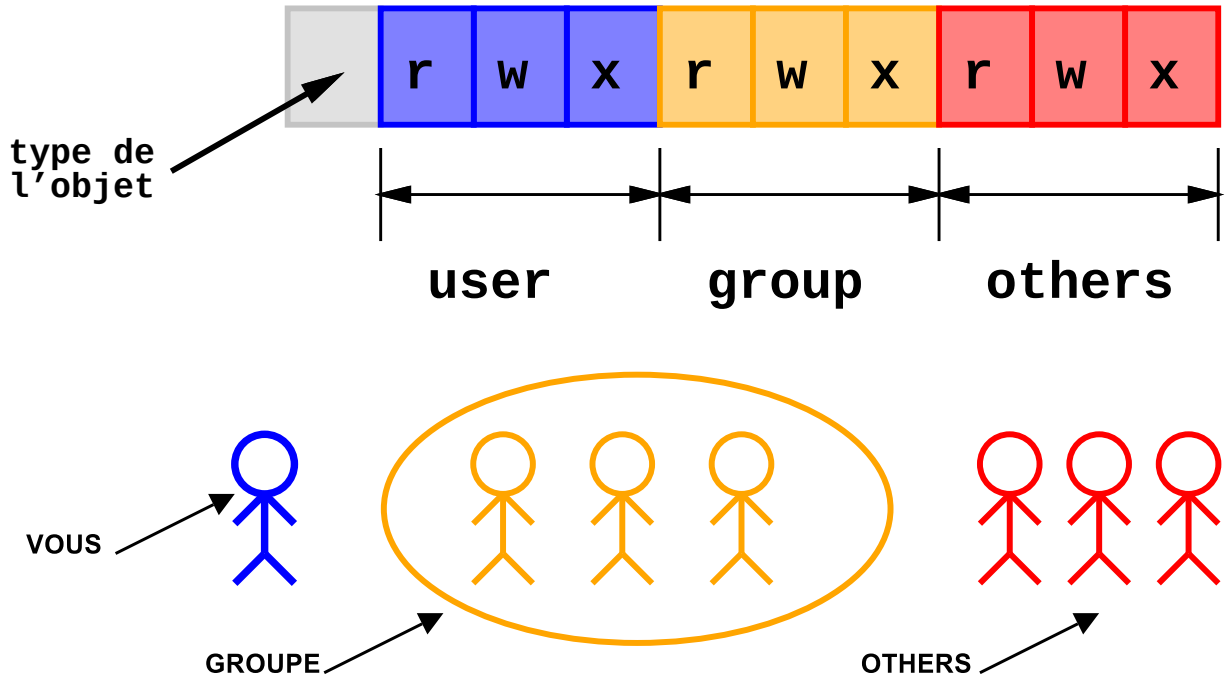
```
% ls -l
total 16
-rw-r--r--  1 besancon ars  15524 Sep 15 15:17 exemple.txt
```



Cas le plus courant :

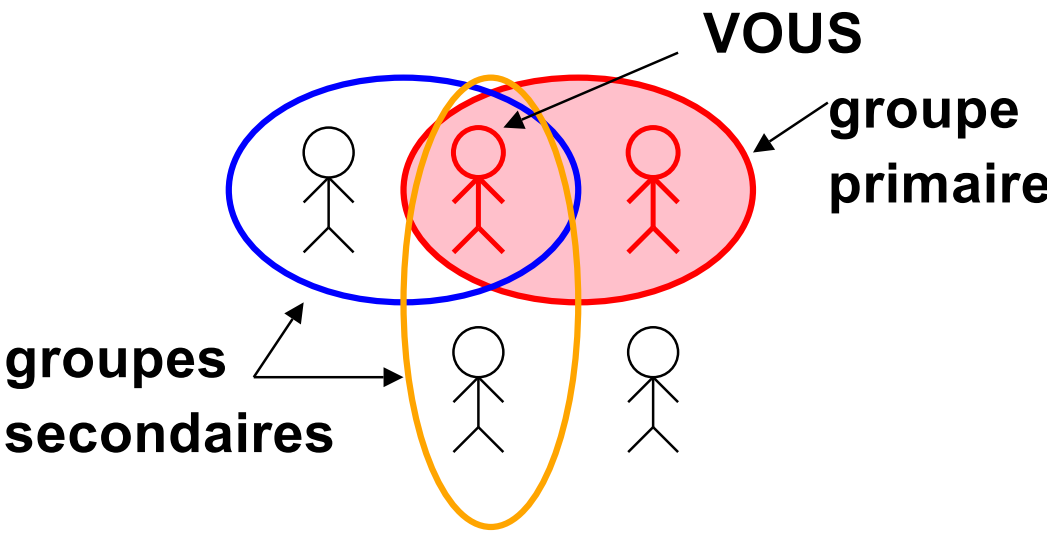


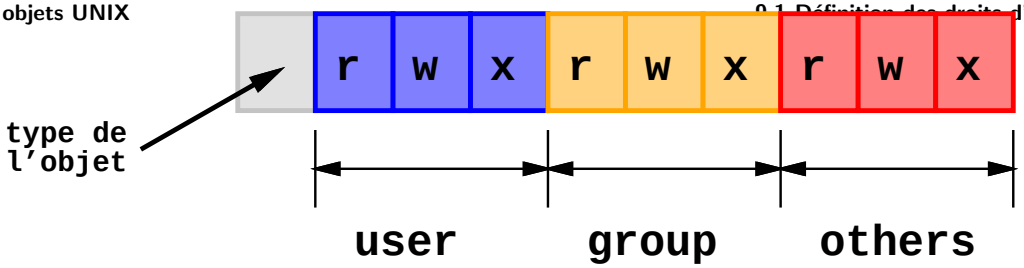
Cas moins courant mais possible :



Le principe :

- Un utilisateur appartient à un groupe primaire
- Un utilisateur peut appartenir à des groupes secondaires
- Un objet a un propriétaire utilisateur et un propriétaire groupe





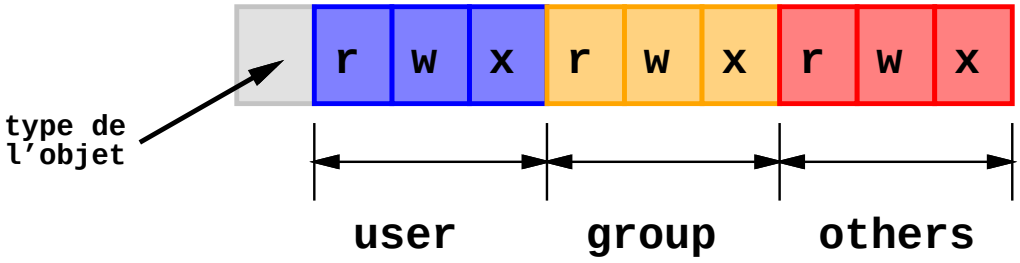
Il existe trois paquets de droits d'accès associés à chaque objet :

- droits du propriétaire (u ≡ user)
- droits des membres du groupe (g ≡ group)
- droits des autres utilisateurs (o ≡ others)

```
% ls -l
total 16
-rw-r--r--  1 besancon ars  15524 Sep 15 15:17 exemple.txt
```

Ici :

- user = utilisateur « `besancon` »
- group = utilisateurs du groupe « `ars` »
- others = les utilisateurs autres que « `besancon` » et autres que les membres du groupe « `ars` »



Il existe trois types de permissions :

- droit en lecture (r ≡ read)
- droit en écriture (w ≡ write)
- droit en exécution (x ≡ execute access)

A partir de Windows NT 4 SP4 :

	Permission Types (Windows NT)					
Permission Components	**Read (R)**	**Write (W)**	**Execute (X)**	**Delete (D)**	**Change Permissions Permissions (P)**	**Take Ownership (O)**
Traverse Folder / Execute File			✔			
List Folder / Read Data	✔					
Read Attributes	✔		✔			
Read Extended Attributes	✔					
Create Files / Write Data		✔				
Create Folders / Append Data		✔				

	Permission Types (Windows NT)					
Permission Components	**Read (R)**	**Write (W)**	**Execute (X)**	**Delete (D)**	**Change Permissions Permissions (P)**	**Take Ownership (O)**
Write Attributes		✔				
Write Extended Attributes		✔				
Delete Subfolders and Files						
Delete				✔		
Read Permissions	✔	✔	✔			
Change Permissions					✔	
Take Ownership						✔

(en anglais *change modes*)
Syntaxe : **chmod [options] modes objets**

Option « -R » pour changer récursivement les droits des objets d'une arborescence.

La précision des modes dans la commande peut prendre deux formes :

- forme symbolique :
  « u » (*user*), « g » (*group*), « o » (*others*) ou « a » (*all*)
  « + » ou « - » ou « = »
  permissions (« r », « w » ou « x »)
- forme numérique :
  Les permissions sont exprimées en base huit ou octale.
  Par exemple : rwx r-x r-x ≡ 755

Droits	Valeur base 2	Valeur base 8
`---`	000	0
`--x`	001	1
`-w-`	010	2
`-wx`	011	3
`r--`	100	4
`r-x`	101	5
`rw-`	110	6
`rwx`	111	7

C'est pourquoi on a par exemple :

`rwx r-x r-x` ≡ 755

`rw- r-- r--` ≡ 644

`rw- --- ---` ≡ 600


```
% ls -l exemple.txt
-rw-r--r--  1 besancon ars   249 Sep 20 22:43 exemple.txt

% chmod g+w exemple.txt

% ls -l exemple.txt
-rw-rw-r--  1 besancon ars   249 Sep 20 22:43 exemple.txt

% chmod o=wx exemple.txt

% ls -l exemple.txt
-rw-rw--wx  1 besancon ars   249 Sep 20 22:43 exemple.txt

% chmod 640 exemple.txt

% ls -l exemple.txt
-rw-r-----  1 besancon ars   249 Sep 20 22:43 exemple.txt
```

On peut retirer tous les droits d'accès :

```
% chmod 000 exemple.txt
---------- 1 besancon ars 249 Sep 20 22:43 exemple.txt
```

Le propriétaire de l'objet peut toujours corriger des droits d'accès (incorrects ou non).

Chapitre 9 • Attributs des objets UNIX
§9.4 • Droits interdits par défaut lors de création d'objets : umask

(en anglais *user mask*)

◇ Utilisation 1

Utilisation 1 : pour connaître les droits interdits par défaut lors de la création d'objets

Syntaxe : **umask**

```
% umask
022
```

◇ Utilisation 2

Utilisation 2 : positionner les droits interdits par défaut

Syntaxe : **umask droits-interdits**

On utilise une notation octale : on indique les bits interdits lors de la création des objets.
Voir page suivante pour le détail de la notation.

ATTENTION : le réglage du umask indique les bits interdits et en prenant le contraire, les bits qui seront peut-être obtenus.

Droits maximum par défaut	Valeur base 2	Valeur base 8
`---`	111	7
`--x`	110	6
`-w-`	101	5
`-wx`	100	4
`r--`	011	3
`r-x`	010	2
`rw-`	001	1
`rwx`	000	0

`umask 022` ≡ On aura au plus par défaut les droits « `rwx r-x r-x` »
`umask 133` ≡ On aura au plus par défaut les droits « `rw- r-- r--` »
`umask 177` ≡ On aura au plus par défaut les droits « `rw- --- ---` »
`umask 077` ≡ On aura au plus par défaut les droits « `rwx --- ---` »
(mode paranoiaque)

ATTENTION BIS :

- le réglage du umask indique les bits interdits par défaut
- le réglage du umask n'indique pas les bits qui seront obtenus
- le réglage du umask indique les bits qui seront obtenus dans le meilleur des cas

Les droits obtenus = les droits demandés par l'application moins les droits interdits (ou filtrés) par le umask

Voir les exemples ci après.

◇ Exemple 1

```
% umask 022

% vi prog.c

% ls -l prog.c
-rw-r--r--   1 besancon ars     127 Oct 12 14:45 prog.c

% gcc prog.c -o prog.exe

% ls -l prog.exe
-rwxr-xr-x   1 besancon ars    6076 Oct 12 14:45 prog.exe
```

◇ Exemple 2

```
% umask 027

% vi prog.c

% ls -l prog.c
-rw-r-----   1 besancon ars     127 Oct 12 14:45 prog.c

% gcc prog.c -o prog.exe

% ls -l prog.exe
-rwxr-x---   1 besancon ars    6076 Oct 12 14:45 prog.exe
```


◇ Exemple 3

```
% umask 000

% vi prog.c

% ls -l prog.c
-rw-rw-rw-   1 besancon ars     127 Oct 12 14:45 prog.c

% gcc prog.c -o prog.exe

% ls -l prog.exe
-rwxrwxrwx   1 besancon ars    6076 Oct 12 14:45 prog.exe
```

Conclusion des 3 exemples :

- « vi » demande les droits 666 par défaut qui sont filtrés par le umask ensuite
- « gcc » demande les droits 777 par défaut qui sont filtrés par le umask ensuite

Conclusion généralisée :

- une application générant un fichier texte demande les droits 666 par défaut qui sont filtrés par le umask ensuite
  droits-résultats = 666 - umask
- une application générant un fichier exécutable demande les droits 777 par défaut qui sont filtrés par le umask ensuite
  droits-résultats = 777 - umask

ATTENTION : un réglage de umask dure le temps d'une session shell ! (voir plus loin comment rendre le réglage permanent)

ATTENTION : le umask de l'administrateur doit être 022 au pire, 077 au mieux !

Hypothèse : on est sous Bourne Shell ou sous BASH.

Objectif : on veut régler son « `umask` » de façon permanente.

Solution :

- On utilise le fichier « `$HOME/.profile` » avec un lien symbolique « `$HOME/.bashrc` » dessus.
  On règle ainsi (sh ou bash) :
  - cas du shell interactif de login
  - cas du shell interactif non de login
  - cas du shell non interactif

  (voir page 631)
- On ajoute dans le fichier « `$HOME/.profile` »

```
umask 022
```


Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setuid** (**4000** en octal).

Avec ce bit positionné, le programme est exécuté avec les droits de l'utilisateur propriétaire.

```
% ls -lg prog1.exe prog2.exe
-rwxr-xr-x   1 besancon ars  249 Sep 20 22:43 prog1.exe
-rwxr-xr-x   1 besancon ars  249 Sep 20 22:43 prog2.exe
% chmod u+s prog1.exe
% chmod 4711 prog2.exe
% ls -lg prog1.exe prog2.exe
-rwsr-xr-x   1 besancon ars  249 Sep 20 22:43 prog1.exe
-rws--x--x   1 besancon ars  249 Sep 20 22:43 prog2.exe
```

Attention à l'affichage du bit setuid !
Classiquement :

```
% gcc prog.c -o prog.exe
% ls -l prog.exe
-rwxr-xr-x   1 besancon ars     6204 Jan 24 20:22 prog.exe
% chmod u+s prog.exe
% ls -l prog.exe
-rwsr-xr-x   1 besancon ars     6204 Jan 24 20:22 prog.exe
```


Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars     127 Jan 24 20:21 exemple.txt
% chmod u+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required
for set-ID on execution
% ls -l exemple.txt
-rw-r--r--   1 besancon ars     127 Jan 24 20:21 exemple.txt
% chmod 4644 exemple.txt
% ls -l exemple.txt
-rwSr--r--   1 besancon ars     127 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « S » ≡ « bit 04000 seul »
- affichage « s » ≡ « bit x + bit S »

ATTENTION : le bit setuid ne fonctionne pas avec un shell script. (voir page 646 pour ce qu'est un shell script)

Il ne fonctionne qu'avec un exécutable binaire.

Une solution sera proposée dans le tôme 3 (commande « `sudo` »).

Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setgid** (**2000** en octal).
Avec ce bit positionné, le programme est exécuté avec les droits du groupe propriétaire

```
% ls -lgF prog1.exe prog2.exe
-rwxr-xr-x   1 besancon ars   249 Sep 20 22:43 prog1.exe
-rwxr-xr-x   1 besancon ars   249 Sep 20 22:43 prog2.exe
% chmod g+s prog1.exe
% chmod 2711 prog2.exe
% ls -lgF prog1.exe prog2.exe
-rwxr-sr-x   1 besancon ars   249 Sep 20 22:43 prog1.exe
-rwx--s--x   1 besancon ars   249 Sep 20 22:43 prog2.exe
```

Attention à l'affichage du bit setgid !
Classiquement :

```
% gcc prog.c -o prog.exe
% ls -l prog.exe
-rwxr-xr-x   1 besancon ars     6204 Jan 24 20:22 prog.exe
% chmod g+s prog.exe
% ls -l prog.exe
-rwxr-sr-x   1 besancon ars     6204 Jan 24 20:22 prog.exe
```


Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars     127 Jan 24 20:21 exemple.txt
% chmod g+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required
for set-ID on execution
% ls -l exemple.txt
-rw-r--r--   1 besancon ars     127 Jan 24 20:21 exemple.txt
% chmod 2644 exemple.txt
% ls -l exemple.txt
-rw-r-lr--   1 besancon ars     127 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « l » ≡ « bit 02000 seul »
- affichage « s » ≡ « bit x + bit l »

ATTENTION : le bit setgid ne fonctionne pas avec un shell script. (voir page 646 pour ce qu'est un shell script)

Il ne fonctionne qu'avec un exécutable binaire.

Une solution sera proposée dans le tôme 3 (commande « `sudo` »).

Il existe un attribut spécial de répertoire réservé à la gestion du système : le **sticky** bit (**1000** en octal).

Avec ce bit positionné, on ne peut effacer d'un répertoire que ses propres fichiers et pas ceux des autres.

Exemple d'utilisation : le répertoire système de stockage des fichiers temporaires « `/tmp` »

```
% ls -ld /tmp
drwxrwxrwt  11 root root     2580 Aug  2 15:40 /tmp
% cd /tmp
% ls -l
-rw-r--r-- 1 besancon adm     90252 Jul 29 16:21 ananas.jpg
-rw-r--r-- 1 besancon adm    255596 Jul 29 16:21 banane.jpg
-rw-r--r-- 1 root     root      639 Sep  1 23:44 pear.con
-rw-r--r-- 1 root     root      614 Sep  1 23:09 php.errors
-rw-r--r-- 1 apache   apache  35383 Aug  8 01:49 ps
% rm ps
rm: ps: override protection 644 (yes/no)? y
rm: ps not removed: Permission denied
% rm php.errors
rm: php.errors: override protection 644 (yes/no)? y
rm: php.errors not removed: Permission denied
```


Historiquement : le sticky bit positionné sur un exécutable le chargeait en mémoire virtuelle et ne l'effacait pas de la zone de swap si bien que le recharger se faisait rapidement.

Mécanisme abandonné (avant 1990).

Bit libre récupéré pour le mécanisme connu maintenant.

Attention à l'affichage du sticky bit !
Classiquement :

```
% mkdir -p exemple.dir
% ls -ld exemple.dir
drwxr-xr-x   2 besancon ars   117 Jan 25 11:09 exemple.dir
% chmod 1777 exemple.dir
% ls -lgd exemple.dir
drwxrwxrwt   2 besancon ars   117 Jan 25 11:09 exemple.dir
```


Moins classiquement :

```
% mkdir -p exemple.dir
% ls -ld exemple.dir
drwxr-xr-x   2 besancon ars   117 Jan 25 11:09 exemple.dir
% chmod 1700 /tmp/exemple.dir
% ls -lgd /tmp/exemple.dir
drwx-----T   2 besancon ars   117 Jan 25 11:12 exemple.dir
```

Bref :

- affichage « T » ≡ « bit 01000 seul »
- affichage « t » ≡ « bit x + bit T »

Sur UNIX, à chaque objet sont associées 3 dates stockées dans une structure dite *inode*. Cette structure n'est pas manipulable directement.

Ces 3 dates sont :

- date de dernière modification dite *mtime* (≡ modification time)
- date de dernier accès dite *atime* (≡ access time)
- date de dernière modification des attributs dite *ctime* (≡ change time)

Ces dates sont affichables via la commande « `ls` ».

Attention : par défaut, « `ls -l` » affiche

- une date de moins de 6 mois sous la forme : « Mois Jour Heure :Minute »
- une date de plus de 6 mois sous la forme : « Mois Jour Année »

```
% ls -l
drwxr-xr-x 3 besancon ars   1024 Oct 23  2005 jardin
-rw-r--r-- 1 besancon ars  29749 Apr  5 20:50 ananas.avi
```

Sur Solaris, option « `-e` » pour un affichage normalisé :

```
% ls -e
drwxr-xr-x 3 besancon ars   1024 Oct 23 19:35:51 2005 jardin
-rw-r--r-- 1 besancon ars  29749 Apr  5 20:50:28 2006 ananas
```

Au niveau de la commande « `ls` » :

- option « `-l` » : format long (affichage du mtime par défaut)
- option « `-t` » : tri décroissant par date (mtime par défaut)
- option « `-r` » : tri par ordre inverse
- « `ls -lt` » : classement par ordre chronologique décroissant des mtimes
- « `ls -ltu` » : classement par ordre chronologique décroissant des atimes
- « `ls -ltc` » : classement par ordre chronologique décroissant des ctimes

Syntaxe : **`date [options] [+format]`**

Quelques cas utiles :

- « `date` » : la date de l'instant courant
- « `date -u` » : la date GMT de l'instant courant
- « `date '+%Y%m%d'` » : date du jour sous la forme « AAAAMMJJ ». Voir page de manuel de la fonction C « `strftime` ».

(en anglais *touch*)
Syntaxe : **touch [options] -t time objet**

Quelques options :

- option « -a » : modification de la date d'accès de l'objet (a ≡ atime)
- option « -m » : modification de la date de modification de l'objet (m ≡ mtime)
- option « -t time » : indique une date autre que la date du moment à mettre ;
  format : « AAAAMMJJhhmm.ss »

◇ Exemple

```
% ls -l exemple.txt
-rw-r--r--  1 besancon ars   49 Sep 27 13:07 exemple.txt

% touch -m -t 199901012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars   49 Jan  1  1999 exemple.txt

% touch -m -t 09012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars   49 Sep  1 22:33 exemple.txt
```

Chapitre 10

Expressions régulières et commandes UNIX associées

Expression régulière = regular expression = regexp

Besoin pratique : faire des recherches dans des fichiers d'enregistrements de base de données, d'inventaires, de comptabilité . . . Bref, des fichiers ayant souvent une **structure forte**.

Pour cela, plusieurs programmes UNIX utilisent des **critères** de reconnaissance de motifs de chaines de caractères.

Un exemple de motif :

« les lignes commencant par la lettre a »

Une expression régulière est la traduction en langage UNIX du motif de recherche, ce qui permettra de le reconnaitre au sein d'un texte dans un fichier.

Ainsi le motif précédent donne la regexp :

^a

La regexp traduit le motif à rechercher sous la forme d'une **suite de contraintes à satisfaire toutes**.

Pour construire les contraintes, on dispose de différents types d'écritures :

- écritures décrivant des valeurs de caractères
- écritures décrivant des positions de caractères
- écritures décrivant des répétitions de caractères

◇ Contraintes sur des valeurs de caractères

`caractère`	désigne la contrainte pour un caractère d'être ce caractère
`[caractères]`	désigne la contrainte pour un caractère d'être parmi la liste indiquée
`[^caractères]`	désigne la contrainte pour un caractère de ne pas figurer dans la liste indiquée
.	désigne la contrainte pour un caractère d'être quelconque

ATTENTION : les caractères « [» et «] » ne désignent pas une option ici. Ne confondez pas avec l'écriture d'une option pour une commande (voir page 54).

Rappel sur les codes ASCII et l'ordre des caractères ASCII (faire « `man ascii` ») :

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK		
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO		
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN		
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS		
32	SP	33	!	34	"	35	#	36	$	37	%	38	&		
40	(	41	)	42	*	43	+	44	,	45	-	46	.		
48	0	49	1	50	2	51	3	52	4	53	5	54	6		
56	8	57	9	58	:	59	;	60	<	61	=	62	>		
64	@	65	A	66	B	67	C	68	D	69	E	70	F		
72	H	73	I	74	J	75	K	76	L	77	M	78	N		
80	P	81	Q	82	R	83	S	84	T	85	U	86	V		
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^		
96	`	97	a	98	b	99	c	100	d	101	e	102	f		
104	h	105	i	106	j	107	k	108	l	109	m	110	n		
112	p	113	q	114	r	115	s	116	t	117	u	118	v		
120	x	121	y	122	z	123	{	124	|	125	}	126	~		

La tables ASCII indique :

- tous les chiffres sont consécutifs
- toutes les majuscules sont consécutives
- toutes les minuscules sont consécutives

D'où la notion d'intervalles notés selon la syntaxe ci-dessous :

- « `[a-z]` » : le caractère peut prendre une valeur entre « a » et « z »
- « `[A-Z]` » : le caractère peut prendre une valeur entre « A » et « Z »
- « `[0-9]` » : le caractère peut prendre une valeur entre « 0 » et « 9 »
- « `[a-zA-Z0-9]` » = « `[A-Za-z0-9]` » = « `[0-9A-Za-z]` » (8 combinaisons au total ; peu importe l'ordre des trois intervalles) : le caractère peut prendre une valeur entre « a » et « z » ou entre « A » et « Z » ou entre « 0 » et « 9 »
- « `[^a-z]` » : les valeurs entre « a » et « z » sont interdites pour le caractère ; le caractère vaudra tout sauf une minuscule

Comment indiquer les caractères spéciaux ?

- « `[a-z-]` » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « - » en plus
- « `[-a-z]` » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « - » en plus
- « `[a-z\[\]]` » : le caractère peut prendre une valeur entre « a » et « z » ou les valeurs « `[` » ou « `]` » en plus
- « `[a-z\\]` » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « \ » en plus
- « `[^a-z]` » : les valeurs entre « a » et « z » sont interdites pour le caractère ; le caractère vaudra tout sauf une minuscule
- « `[\^a-z]` » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « ^ » en plus
- « `[a-z^]` » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « ^ » en plus (la position de « ^ » est non ambigue et signifie simplement « ^ »)

◇ Contraintes sur les positions de caractères

^	désigne la contrainte d'être en début de ligne
$	désigne la contrainte d'être en fin de ligne

Rappel sur un clavier AZERTY :
La touche « ^ » sert à entrer un accent circonflexe.
Vous obtenez donc le caractère « ^ » en faisant « ^ » + « ESPACE ».

◇ Répétitions d'une contrainte

contrainte { min,max }	indique que la contrainte mentionnée juste avant doit être satisfaite entre min et max fois
contrainte*	indique que la contrainte mentionnée juste avant doit être satisfaite autant de fois que possible en pratique (de 0 à autant que l'on veut)

Par exemple :

- « `{3,7}` » : contrainte précédente répétée entre 3 et 7 fois
- « `{3,}` » : contrainte précédente répétée au moins 3 fois
- « `{5}` » : contrainte précédente répétée 5 fois
- « `{1}` » : contrainte précédente répétée 1 fois ; cette écriture n'a pas d'intérêt : autant écrire la contrainte précédente simplement toute seule

ATTENTION : principes FONDAMENTAUX des regexps

- on analyse chaque ligne indépendemment de la précédente et de la suivante
- **on analyse de gauche à droite, caractère par caractère en cherchant si l'on vérifie la regexp ; passage au caractère suivant àdroite si l'on ne vérifie pas la regexp sur la position courante**
- une contrainte peut être rendue muette selon le contexte pour satisfaire la regexp au total
- via la regexp, on essaye de vérifier (matcher) la chaine de caractères la plus longue possible

Analyse de la gauche vers la droite

Exemple : rechercher le mot « ananas » dans le texte suivant :

Application des regexps :

- sélectionner des lignes de fichiers texte qui satisfont la regexp
  ⇒ c'est l'objet de la commande UNIX « **grep** »
- faire des remplacements du texte satisfaisant la regexp par un autre texte qui peut-être déduit du texte initial
  ⇒ c'est l'objet de la commande UNIX « **sed** »

◇ Exemple 1

Le motif à satisfaire :

la ligne contient le mot elephant

Traduction en regexp :

```
elephant
```

En effet, c'est une suite de 8 contraintes à satisfaire toutes :

- contrainte 1 : un caractère doit valoir « e »
- contrainte 2 : un caractère doit valoir « l »
- contrainte 3 : un caractère doit valoir « e »
- contrainte 4 : un caractère doit valoir « p »
- contrainte 5 : un caractère doit valoir « h »
- contrainte 6 : un caractère doit valoir « a »
- contrainte 7 : un caractère doit valoir « n »
- contrainte 8 : un caractère doit valoir « t »

◇ Exemple 2

Le motif à satisfaire :

la ligne se termine par 2 caractères en minuscules suivis d'un chiffre

Démarche de la traduction :

1. « se termine » : on va procéder de droite à gauche
2. « se termine » : « `$` »
3. « un chiffre » : « `[0-9]` »
4. « une minuscule » : « `[a-z]` »
5. « une minuscule » : « `[a-z]` »

Résultat final, assemblage des résultats intermédiaires :

```
[a-z][a-z][0-9]$
```

ou

```
[a-z]{2}[0-9]$
```

◇ Exemple 3

Le motif à satisfaire :

« `mot1:mot2:mot3:` » *en début de ligne où*

- *mot1 commence par une lettre majuscule*
- *mot2 se termine par un chiffre*
- *mot3 est quelconque*

Processus de traduction :

1. « en début de ligne »
   ⇒ la regexp est de la forme « `^` »
2. « mot1 commence par une lettre majuscule »
   ⇒ mot1 commence par une lettre majuscule et le reste des lettres est quelconque
   ⇒ mot1 commence par une lettre majuscule et le reste des lettres est quelconque sans pour autant valoir le caractère : qui sépare les mots
   ⇒ la regexp est de la forme « `[A-Z][^:]*` »
3. « mot1 :mot2 »
   ⇒ la regexp est de la forme « `:` »
4. « mot2 se termine par un chiffre »
   ⇒ mot2 commence par des caractères quelconques et se termine par un chiffre
   ⇒ mot2 commence par des caractères quelconques sans pour autant valoir : qui sépare les mots et se termine par un chiffre
   ⇒ la regexp est de la forme « `[^:]*[0-9]` »

5. « mot2 :mot3 : »
   ⇒ la regexp est de la forme « `:` »
6. « mot3 est quelconque »
   ⇒ mot3 est composé de caractères quelconques
   ⇒ mot3 est composé de caractères quelconques sans pour autant valoir le caractère : qui sépare les mots
   ⇒ la regexp est de la forme « `[^:]*` »
7. « mot3 : »
   ⇒ la regexp est de la forme « `:` »

Résultat final, assemblage des résultats intermédiaires :

```
^[A-Z][^:]*:[^:]*[0-9]:[^:]*:
```


◇ Exemple 4

Soit la regexp :

```
^[^abc]*
```

Soit le fichier contenant les lignes suivantes :

```
ascenceur
berceau
chameau
elephant
```

La regexp sélectionne les lignes suivantes :

```
ascenceur
berceau
chameau
elephant
```

Pourquoi ?

Principe de rendre muette une contrainte pour satisfaire la regexp globalement

◇ Syntaxe en pratique

En pratique, avec « `grep` » ou « `sed` », écrire la regexp entre deux apostrophes.

L'explication des apostrophes sera vue au niveau du cours sur la pratique du shell. Voir page 543.

Le chapitre précédent décrit les **Basic Regular Expressions** (BRE).

Il existe des extensions au langage : les **Extended Regular Expressions** (ERE).

Par exemple la regexp :

```
(cerise|ananas)
```

Cette regexp permet de chercher les lignes contenant le mot « `ananas` » ou « `cerise` ».
Impossible à traduire avec les constructions basiques de la page 381.
Forme générale : « `(regexp1|regexp2|...|regexpN)` »

Il ne sera pas décrit davantage les ERE ici.

Le langage PERL a son propre langage de Regular Expressions.

Des programmeurs ont écrit des librairies C compatibles avec les Regular Expressions de PERL : les PCRE (PERL Compatible Regular Expressions).

Voir par exemple « `http://www.pcre.org/` ».

Il ne sera pas décrit davantage les PCRE ici.

(le mot grep vient de « `g/re/p` » dans vi)
Syntaxe : **`grep [options] regexp fichiers`**

Quelques options intéressantes :

- option « `-i` » : pas de différenciation entre lettres minuscules et majuscules
- option « `-n` » : affichage des numéros de ligne
- option « `-l` » : n'affiche que les noms de fichiers
- option « `-v` » : affichage des lignes ne contenant pas la chaîne précisée

◇ Exemple 1

Soit le fichier :

```
Ecrivons toto en minuscules ici.
Et ici ToTo en minuscules et majuscules.
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep 'toto' exemple.txt
Ecrivons toto en minuscules ici.
```


◇ Exemple 2

Soit le fichier :

```
Ecrivons toto en minuscules ici.
Et ici ToTo en minuscules et majuscules.
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep -i -n 'toto' exemple.txt
1:Ecrivons toto en minuscules ici.
2:Et ici ToTo en minuscules et majuscules.
```

◇ Exemple 3

Soit le fichier :

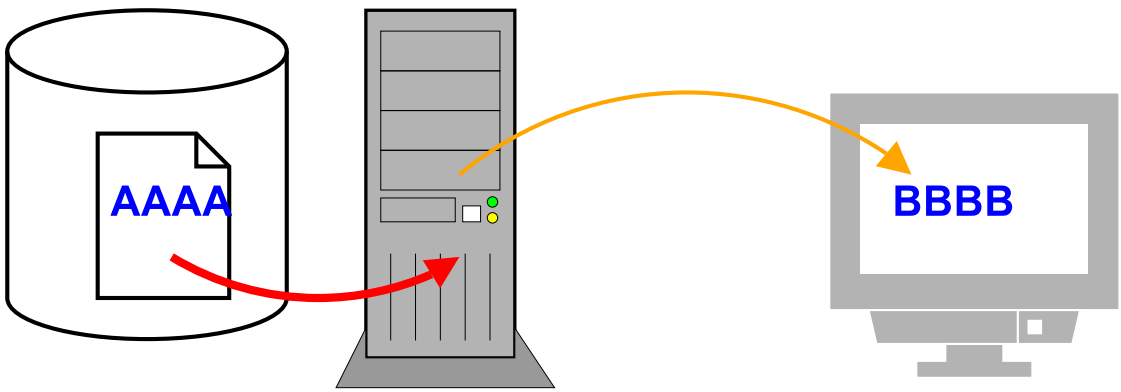
```
Ecrivons toto en minuscules ici.
Et ici ToTo en minuscules et majuscules.
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep -i -n -v 'toto' exemple.txt
3:Mais là on ne met pas la chaîne de l'exemple.
```


(en anglais *stream editor*)
Syntaxe : `sed [options] fichiers`

La commande « `sed` » agit sur un **flux**.



sed -e 's/AAAA/BBBB/' exemple.txt

Qu'est qu'un flux ?

Un flux est une quantité de texte envoyé à l'affichage par une commande.

Exemples typiques de flux :

- le résultat d'une commande « `ls` »
- le résultat d'un « `cat exemple.txt` »

sed modifie un flux, pas un contenu de fichier : après l'application de la commande, le fichier utilisé est inchangé.

Quelques options intéressantes :

- option « `-e commande` » : commande à exécuter sur chaque ligne du flux
- option « `-f script` » : précision d'un fichier dans lequel prendre les commandes à appliquer sur chaque ligne de texte

Une commande prend l'une des formes suivantes :

1. `[adresse1[,adresse2]]fonction[argument]`
2. `[/regexp1/[,/regexp2/]]fonction[argument]`

(avec les crochets « `[]` » désignant l'aspect facultatif de l'objet)

On construit une adresse de ligne grâce à :

- son numéro de ligne
- le caractère « `$` » pour désigner la dernière ligne

Les fonctions proposées dans sed :

- suppression de ligne : « `d` » (en anglais *delete*)
- affichage de ligne : « `p` » (en anglais *print*)
  La fonction « `p` » n'est intéressante que couplée à l'option « `-n` » de « `sed` » car l'option « `-n` » supprime l'affiche par défaut des lignes.
- substitution au sein des lignes :
  « `s/regexp/remplacement/modifiers` » (en anglais *substitute*)

Des compléments au fonctionnement de la commande sed seront donnés en TP.

◇ Exemple 1

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

On veut supprimer les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -e '1,2d' exemple.txt
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```


◇ Exemple 1 bis

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

On veut supprimer les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -n -e '3,$p' exemple.txt
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

◇ Exemple 2

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

On ne veut garder que les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -e '3,$d' exemple.txt
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
```


◇ Exemple 2bis

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

On ne veut garder que les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -n -e '1,2p' exemple.txt
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
```

◇ Exemple 3

Soit le fichier « exemple.txt » :

```
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

On veut remplacer le mot « epuise » du fichier par 0 **lors de son affichage** :

```
% sed -e 's/epuise/0/' exemple.txt
moteur;ferrari;30
moteur;porsche;0
carrosserie;porsche;0
moteur;ford;40
moteur;skoda;0
```


◇ Exemple 4

Soit le fichier « exemple.txt » :

```
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

On veut remplacer le mot « epuise » du fichier par 0 pour les lignes parlant de moteur **lors de son affichage** :

```
% sed -e '/moteur/s/epuise/0/' exemple.txt
moteur;ferrari;30
moteur;porsche;0
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;0
```

ATTENTION : dans les deux pages précédentes, la solution est **APPROXIMATIVE** mais suffit !

Que se passerait-il si le mot « epuise » apparaissait ailleurs ? ? ?

Exemple : inventaire d'un marchand d'articles de pêche :

```
flotteur;30
epuisette;10
hameçon;epuise
canne à pêche;10
```

```
% sed -e 's/epuise/0/' exemple.txt
flotteur;30
0tte;10
hamecon;0
canne à pêche;10
```


Soit le fichier « `exemple.txt` » **EN ANGLAIS** :

```
motor;ferrari;30
wheel;general motors;5
motor;ford;40
```

Si on remplace le mot « motor » sans réfléchir :

```
% sed -e 's/motor/XXXXX/' exemple.txt
XXXXX;ferrari;30
wheel;general XXXXXs;5
XXXXX;ford;40
```

◇ Exemple 5

Soit le fichier « exemple.txt » :

```
Les courses de chevals se font a Vincennes.
Les chevals ont 4 jambes. Les chevals sont des equides.
```

On veut remplacer le mot « chevals » du fichier par « chevaux » **lors de son affichage** :

```
% sed -e 's/chevals/chevaux/' exemple.txt
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevals sont des equides.
```

⇒ Le remplacement n'a pas lieu sur tous les mots « chevals » au sein d'une ligne !

◇ Exemple 5 (suite)

Soit le fichier « exemple.txt » :

```
Les courses de chevals se font a Vincennes.
Les chevals ont 4 jambes. Les chevals sont des equides.
```

On veut remplacer le mot « chevals » du fichier par « chevaux » **lors de son affichage** :

```
% sed -e 's/chevals/chevaux/g' exemple.txt
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

⇒ Le remplacement a lieu sur tous les mots « chevals » au sein d'une ligne.

Pour utiliser les Extended Regular Expressions (ERE) au niveau des commandes « grep » ou « sed », il faut indiquer des options supplémentaires :

- sur Solaris :
  « **egrep [options] 'regexp' fichiers** »
- sur LINUX :
  « **grep -E [options] 'regexp' fichiers** »
- sur LINUX :
  « **sed -r '...' fichiers** »

Variante de SED : le GNU sed.
Voir « http://www.gnu.org/ »

Option intéressante « --color » (surtout pour les débutants).
Soit le fichier « exemple.txt » :

```
% cat exemple.txt
nanananas

% grep --color -E 'ananas' exemple.txt
nananananas
```

Soit le fichier « exemple.txt » :

```
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

```
% grep --color -E 'porsche' exemple.txt
moteur;porsche;epuise
carrosserie;porsche;epuise
```


Soit le fichier « exemple.txt » :

```
aaaaab;ccccc
aaaaa--aaaaa
```

```
% grep --color -E 'a{2,4}' exemple.txt
aaaaab;ccccc
aaaaa--aaaaa
```

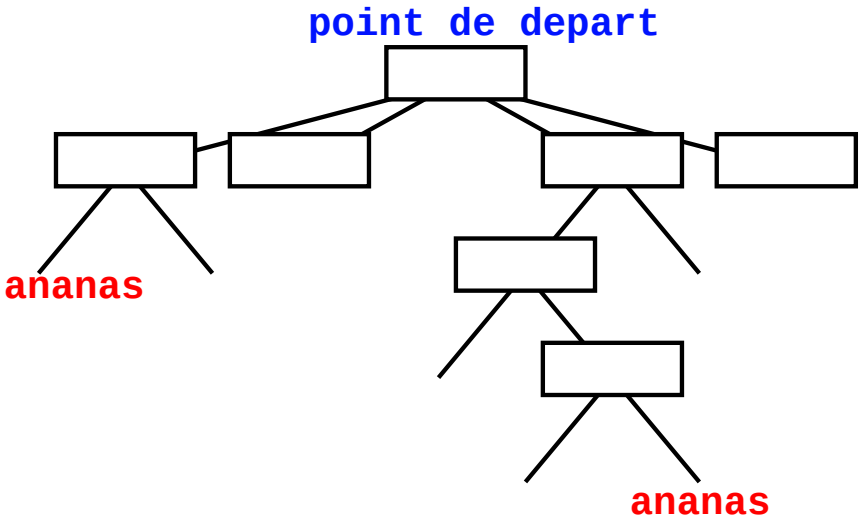
Quelques informations :

- « `http://www.regular-expressions.info/` »
- « Expressions régulières » de Bernard Desgraupes, aux éditions Pearson (16 EUR)

Chapitre 11

Commande de recherche d'objets : `find`

(en anglais *find*)
On recherche **à partir des répertoires indiqués** les objets répondant aux critéres exprimés par des expressions.



Syntaxe : **find répertoires [expressions]**

Les expressions indiquent :

- des conditions
- des actions à effectuer

◇ Options pour rechercher sur un nom :

Syntaxe : « `-name nom` »
Le nom sera spécifié avec des métacaractères du shell (voir page 581)

Autre syntaxe possible (seulement LINUX) : « `-iname nom` » (lettres minuscules ou majuscules indifférenciées)

Le nom sera spécifié avec des métacaractères du shell (voir page 581)

Exemple : rechercher des objets d'extension « `.c` » :

```
% find . -name \*.c -print
% find . -name '*.c' -print
```

◇ Options pour rechercher sur des droits d'accès :

Plusieurs syntaxes possibles :

- Syntaxe : « `-perm permissions` »
  Les permissions doivent être strictement celles indiquées.
- Syntaxe : « `-perm -permissions` »
  Les bits indiqués doivent tous exister parmi les permissions des objets recherchés.
- Syntaxe (seulement LINUX) : « `-perm +permissions` »
  Au moins l'un des bits indiqués doit exister parmi les permissions des objets recherchés.

Exemple : rechercher des objets de permission 755 :

```
% find . -perm 755 -print
```


- « `-perm 123` » :
  Les droits doivent être exclusivement 123 = « `-x -w- -wx` ».
- « `-perm -123` » :
  Les droits doivent être au moins 123 = « `-x -w- -wx` ».
  Les droits 763 = 123 + 640 sont OK par exemple : « `rwx rw- -wx` »
- « `-perm +123` » :
  Au moins l'un des bits 123 = « `-x -r- -rx` » doit exister parmi les permissions des objets.
  Les droits 700 sont OK par exemple : « `rwx --- ---` »

◇ Options pour rechercher sur le propriétaire d'objets :

Plusieurs syntaxes possibles :

- Syntaxe : « `-user login` »
  Le propriétaire de l'objet doit être le login indiqué
- Syntaxe : « `-nouser` »
  Le fichier doit appartenir à un utilisateur non défini sur le système, c'est-à-dire l'UID n'a pas de login associé dans « `/etc/passwd` »

Exemple : rechercher les objets de l'utilisateur « besancon » :

```
% find . -user besancon -print
```


◇ Options pour rechercher sur types d'objets :

Syntaxe : « `-type X` »
avec X une lettre indiquant la nature de l'objet :

- « d » pour dossier (en anglais *directory*)
- « f » pour fichier (en anglais *file*)
- « l » pour lien symbolique (en anglais *link*)
- « c » pour fichier character (voir tôme 2)
- « b » pour fichier bloc (voir tôme 2)
- « s » pour socket (voir tôme 3)

(d'autres lettres existent pour des objets plus sophistiqués qui ne seront pas abordés en cours)

Exemple : rechercher des répertoires :

```
% find . -type d -print
```

◇ Options pour rechercher sur des tailles d'objets :

Plusieurs syntaxes possibles :

- Syntaxe : « `-size nombre` »
- Syntaxe : « `-size -nombre` »
- Syntaxe : « `-size +nombre` »

Si le nombre indique « c », alors il exprime des octets sinon ce sont des blocs de 512 octets.

Exemple : rechercher des objets de plus de 1000000 caractères :

```
% find . -size +1000000c -print
```


◇ Options pour rechercher sur des dates :

Plusieurs syntaxes :

- Syntaxe : « `-newer fichier` »
- Syntaxe : « `-atime nombre` »
- Syntaxe : « `-mtime nombre` »
- Syntaxe : « `-ctime nombre` »

Sur SOLARIS, l'unité du nombre est en jours.

Sur LINUX, plusieurs unités sont possibles : « d » pour jour, « h » pour heure, « m » pour minute, « s » pour seconde

Exemple : rechercher des objets moins vieux de 3 jours

```
% find . -mtime -3 -print
```

◇ Composition d'options :

Plusieurs syntaxes possibles :

- Syntaxe : OU logique entre expressions :
  « `condition1 -o condition2` »
  (attention : OU entre **expressions** ci-dessus)
- Syntaxe : ET logique entre expressions :
  « `condition1 -a condition2` »
  (attention : ET entre **expressions** ci-dessus) ; le `-a` est facultatif

Exemples :

```
% find . -name fruits -type d -print
% find . -name ananas -o -name cerise -print
```


Plusieurs syntaxes possibles (suite) :

- Syntaxe : groupement d'expressions :
  « `( expression1 -[ao] expression2 )` »
  ATTENTION : nécessiter de protéger du shell les parenthèses !
  (voir page 519)
  ⇒ écriture en pratique :
  « `\( expression1 -[ao] expression2 \)` »

◇ Options pour affichage du nom des objets trouvés :

Plusieurs options possibles :

- Syntaxe : « `-print` »
- Syntaxe : « `-ls` »

Exemples :

```
% find . -type d -print
% find . -type f -ls
```


◇ Option d'exécution d'une commande :

Syntaxe : « `-exec commande {} ;` »
ATTENTION : nécessiter de protéger du shell le point-virgule !
(voir page 517)
⇒ écriture en pratique : « `-exec commande {} \;` »

Exemple : rechercher tous les objets s'appelant `a.out` ou s'appelant avec une extension '`.o`', non utilisés depuis plus de 7 jours et on appliquera la commande d'effacement aux objets trouvés :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

◇ Option de recherche des liens d'un inode

Syntaxe : « -inum numero-inode »

Exemples :

```
% mkdir jardin
% touch jardin/ananas
% mkdir zoo
% ln jardin/ananas zoo/lion

% ls -i jardin/ananas
  11854169 jardin/ananas

% find . -inum 11854169 -print
./jardin/ananas
./zoo/lion

% find . -inum 11854169 -ls
11854169 0 -rw-r--r-- 2 besancon ars 0 Oct 22 00:21 ./jardin/anana
11854169 0 -rw-r--r-- 2 besancon ars 0 Oct 22 00:21 ./zoo/lion
```


◇ Retour sur l'arborescence UNIX

Le répertoire courant est noté « . ».
Lancer une recherche à partir de l'endroit où l'on est :

```
% find . -name ananas.txt -print
```

Ne pas confondre la commande find et la commande ls

Rechercher un fichier nommé « ananas.txt » dans une arborescence :

- OUI : « `find . -name ananas.txt -print` »
- NON : « `ls -Rl | grep ananas.txt` »

Pourquoi ?
→ La commande `ls` pourrait renvoyer des données parasites.

Dans le répertoire courant, chercher les fichier commencant par « banane » :

- NON : « `find . -name 'banane*' -print` »
- OUI : « `ls banane*` »

Pourquoi ?
→ La commande `find` va faire une recherche récursive et dépasser le niveau du répertoire courant en descendant plus bas.

La directive -exec n'est pas très pratique.

Exemple où -exec n'est pas adapté :

Rechercher les objets commencant par banane et les renommer en 2003-banane...

→ La directive -exec symbolise le fichier à afficher par « {} » mais ne permet pas de construire des commandes non basiques :

```
% ls banane*
banane1   banane2

% find . -name 'banane*' -exec echo {} {} \;
./banane1 ./banane1
./banane2 ./banane2

% find . -name 'banane*' -exec echo {} 2003-{} \;
./banane1 2003-{}
./banane2 2003-{}
```


Les objets trouvés partent du point de recherche et le mentionnent.

```
% ls banane*
banane1   banane2

% find . -name 'banane*' -print
./banane1
./banane2
```

Attention si vous devez donc retraiter les noms des objets trouvés (ici parasitage du « ./ »).

Chapitre 12

Commandes UNIX réseau de base

(en anglais *host name*)
Syntaxe : **`hostname`**

```
% hostname
serveur.formation.jussieu.fr
```

(la commande sert aussi à baptiser une machine. Syntaxe :
`hostname nom-de-machine`)

(en anglais *? ? ? name*)
Syntaxe : **uname [options]**

```
% uname -n
serveur.formation.jussieu.fr

% uname -a
Linux serveur.formation.jussieu.fr 2.4.20-28.7smp #1 SMP Thu Dec
18 11:18:31 EST 2003 i686 unknown

% uname -s
Linux

% uname -m
i686

% uname -r
2.4.20-28.7smp
```


Autre exemple :

```
% uname -n
solaris.example.com

% uname -a
SunOS solaris.example.com 5.10 Generic_118822-20 sun4u sparc
SUNW,Sun-Blade-100

% uname -s
SunOS

% uname -m
sun4u

% uname -r
5.10
```

(en anglais *? ? ? ping*)
La commande « ping » permet de tester si une machine est joignable.

Principe : envoi d'un paquet réseau spécial à une machine (effet « ping »)
qui répond par un autre paquet réseau spécial (effet « pong »)

Syntaxe : **ping [options]**|PARAMSnom-de-machine

```
% ping host1.example.com
sendto: Network is unreachable
% ping host2.example.com
host2.example.com is alive
```

Humour :

```
% ping elvis
elvis is alive
```


Autre type d'affichage :

```
% ping www.lemonde.fr
PING a245.g.akamai.net (81.52.207.14) from 134.157.46.137 :
56(84) bytes of data
.
64 bytes from 81.52.207.14: icmp_seq=0 ttl=53 time=11.834 ms
64 bytes from 81.52.207.14: icmp_seq=1 ttl=53 time=11.499 ms
64 bytes from 81.52.207.14: icmp_seq=2 ttl=53 time=11.103 ms
64 bytes from 81.52.207.14: icmp_seq=3 ttl=53 time=11.651 ms
64 bytes from 81.52.207.14: icmp_seq=4 ttl=53 time=10.817 ms
64 bytes from 81.52.207.14: icmp_seq=5 ttl=53 time=11.354 ms
^C
--- a245.g.akamai.net ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max/mdev = 10.817/11.376/11.834/0.349 ms
```

Commande « ping.exe »

(en anglais *trace route*)
La commande traceroute permet de tester si une machine est joignable. Elle renvoie les intermédiaires réseau qui route notre acheminement vers la machine distante.
Syntaxe : **traceroute machine**

```
% traceroute ftp.lip6.fr
traceroute to nephtys.lip6.fr (195.83.118.1), 30 hops max, 40 byte
packets
 1  yacht (129.199.96.254)  0 ms  0 ms  0 ms
 2  renater (129.199.1.10)  2 ms  1 ms  1 ms
 3  195.221.127.61 (195.221.127.61)  3 ms  1 ms  1 ms
 4  195.221.126.1 (195.221.126.1)  2 ms  1 ms  1 ms
 5  195.221.126.78 (195.221.126.78)  2 ms  1 ms  1 ms
 6  jussieu.rap.prd.fr (195.221.126.33)  2 ms  2 ms  2 ms
 7  nephtys.lip6.fr (195.83.118.1)  2 ms  2 ms  2 ms
```

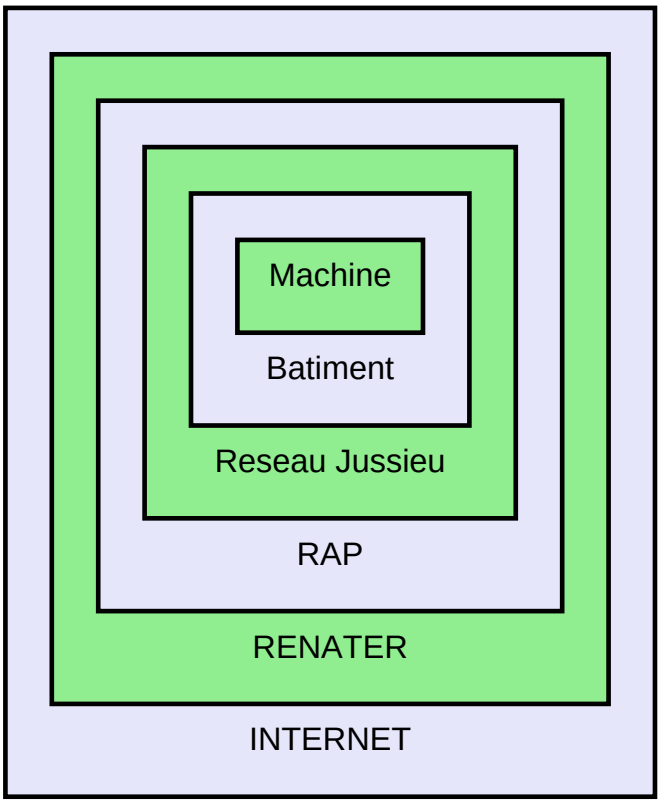
Le nombre d'intermédiaires n'est pas proportionnel à l'éloignement géographique de la machine destination.

En cas de soucis :

```
% traceroute serveur.formation.jussieu.fr
traceroute to serveur.formation.jussieu.fr (134.157.46.129), 30 hops
max, 40 byte packets
 1  yacht (129.199.96.254)  0 ms  0 ms  0 ms
 2  renater (129.199.1.10)  2 ms  1 ms  1 ms
 3  195.221.127.61 (195.221.127.61)  2 ms  2 ms  1 ms
 4  195.221.126.1 (195.221.126.1)  2 ms  1 ms  1 ms
 5  195.221.126.78 (195.221.126.78)  2 ms  1 ms  2 ms
 6  jussieu.rap.prd.fr (195.221.126.33)  3 ms  2 ms  2 ms
 7  134.157.254.123 (134.157.254.123)  3 ms  2 ms  2 ms
 8  * * *   <- symptôme de problème
^C
```

Attention : ne pas oublier les filtrages réseau des firewalls qui peuvent se traduire au niveau de ping comme un problème (étoiles).

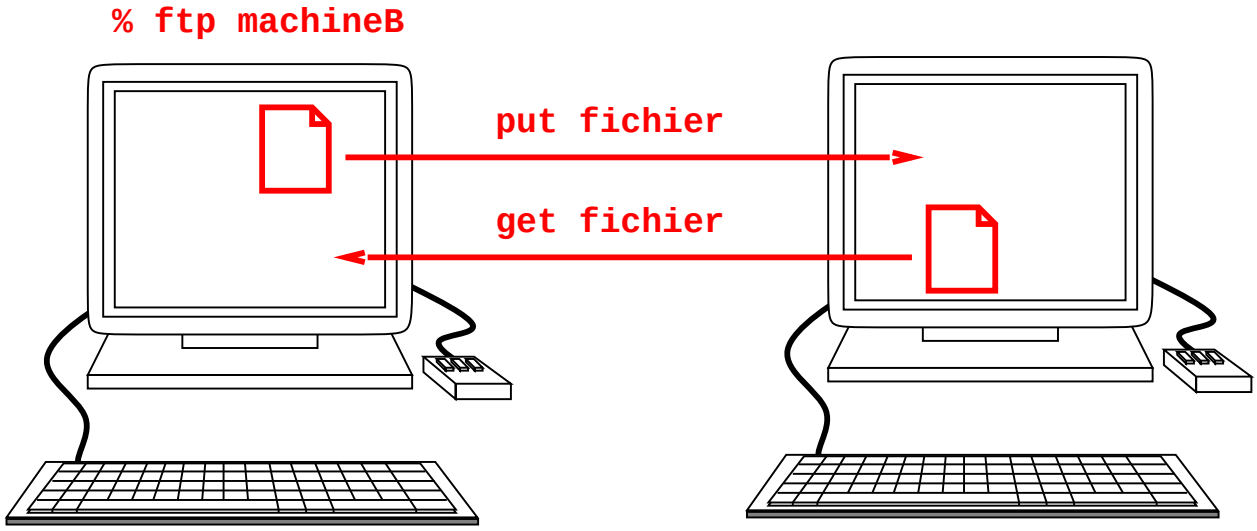
Utile de connaitre les FAI (*Fournisseur d'Accès à Internet*) que vous utilisez :



Commande « tracert.exe »

```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\besancon>tracert serveur.formation.jussieu.fr

Tracing route to serveur.formation.jussieu.fr [134.157.46.129]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  192.168.1.254
  2    40 ms    40 ms    41 ms  [illegible]
  3    42 ms     *      135 ms  th2-6k-2-a5.routers.proxad.net [213.228.7.254]
  4     *        *        *     Request timed out.
  5    39 ms    40 ms     *     p19-6k-2-po4.intf.routers.proxad.net [212.27.50.14]
  6    39 ms    42 ms    39 ms  aub-6k-1-v806.routers.proxad.net [212.27.50.162]
  7    41 ms    40 ms    41 ms  renater.sfinx.tm.fr [194.68.129.102]
  8    43 ms    40 ms    39 ms  nri-c-gi3-0-0-12.cssi.renater.fr [193.51.179.154]
  9    42 ms    40 ms    39 ms  jussieu-pos4-0.cssi.renater.fr [193.51.180.157]
 10    41 ms    45 ms    39 ms  gw-rap.rap.prd.fr [193.50.20.73]
 11    59 ms    43 ms    40 ms  jussieu-rap.rap.prd.fr [195.221.127.182]
 12    42 ms    49 ms    42 ms  r-intercon3.reseau.jussieu.fr [134.157.254.26]
 13    44 ms    41 ms    43 ms  serveur.formation.jussieu.fr [134.157.46.129]

Trace complete.

C:\Documents and Settings\besancon>_
```


(en anglais *file transfer protocol*)
Syntaxe : **ftp machine**



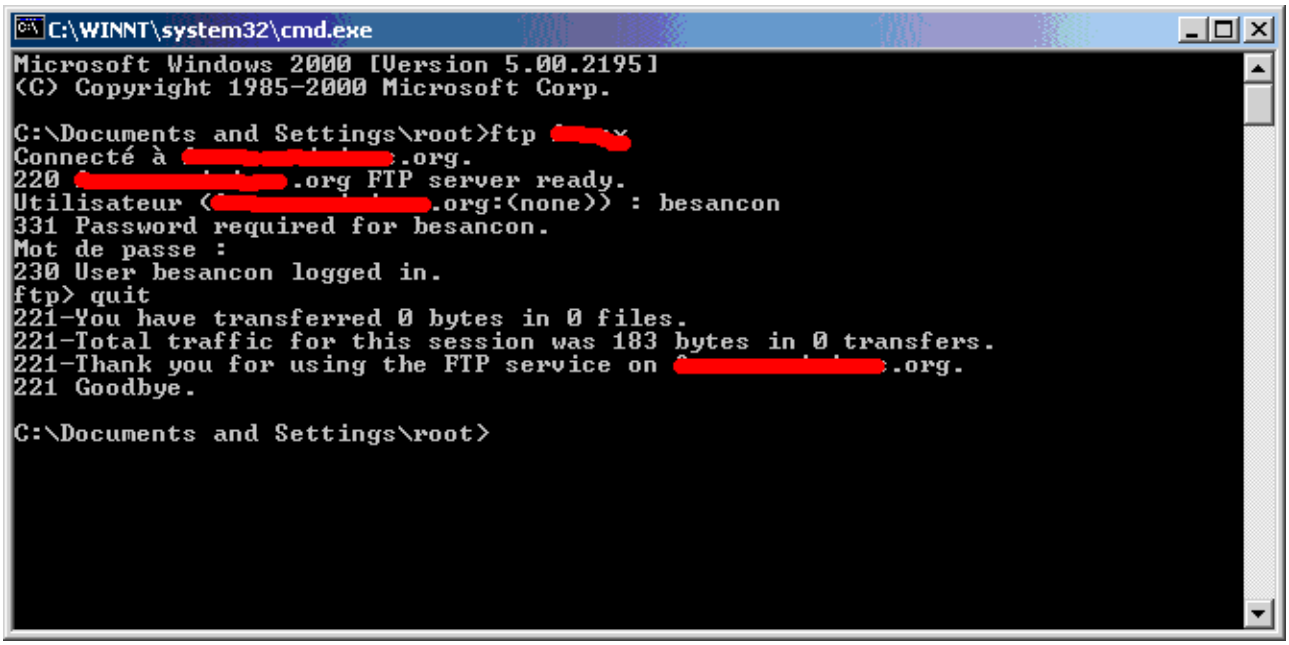
La commande `ftp` vous place dans une espèce de shell dans lequel vous disposez des commandes suivantes (ce sont les plus importantes à votre niveau) :

- commande « `binary` » : à utiliser si le fichier à transmettre contient des caractères non texte
- commande « `dir` » : pour lister les fichiers sur la machine distante
- commande « `!ls` » : pour lister les fichiers sur la machine locale
- commande « `cd directory` » : pour changer de répertoire sur la machine distante
- commande « `lcd directory` » : pour changer de répertoire sur la machine locale
- commande « `get fichier` » : pour récupérer sur la machine distante un fichier
- commande « `put fichier` » : pour déposer sur la machine distante un fichier
- commande « `quit` » : pour se déconnecter

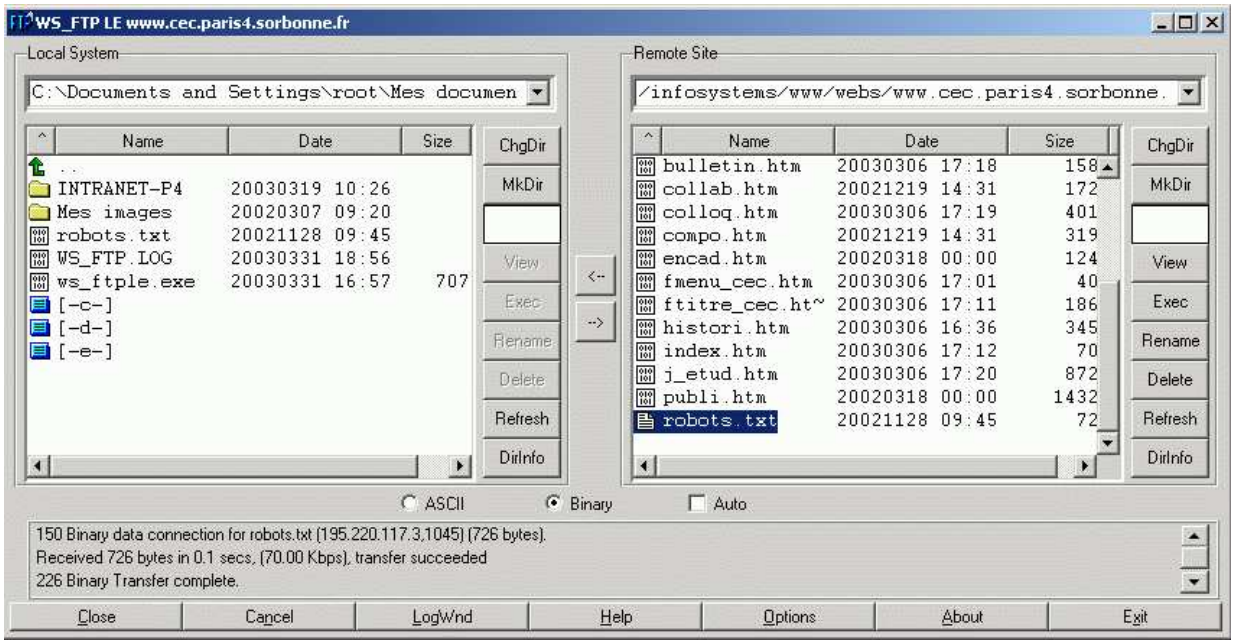

```
% ftp unix.example.com
Connected to unix.example.com.
220 unix.example.com FTP server (SunOS 4.1) ready.
Name (unix:besancon):
331 Password required for besancon.
Password:
230 User besancon logged in.
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (192.168.0.1,3945) (0 bytes).
total 307
-rw-r--r--  1 besancon software       69 Mar 20  1995 .Xdefaults
...
ftp> get fichier [nouveau nom]
...
ftp> put fichier [nouveau nom]
200 PORT command successful.
150 ASCII data connection for fichier (192.168.0.1,3947).
226 ASCII Transfer complete.
local: fichier remote: fichier
802 bytes sent in 0.0042 seconds (1.9e+02 Kbytes/s)
ftp> quit
221 Goodbye.
```

Multiples utilitaires pour faire du FTP sous WINDOWS.

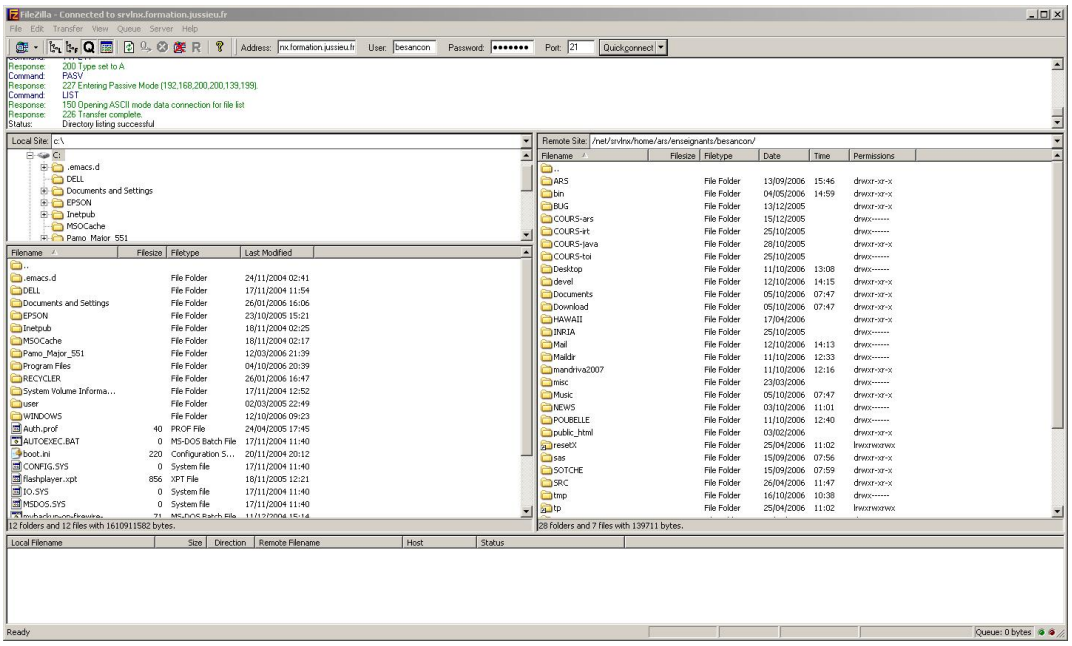
Utilitaire « ftp.exe » fourni par WINDOWS :



Exemple d'utilitaire graphique archi classique : WS_FTP LE (site http://www.ipswitch.com)



Exemple d'utilitaire graphique : `filezilla` (site `http://filezilla.sourceforge.net`, disponible UNIX, WINDOWS, MacOS)



(en anglais *secure shell*)

Le protocole SSH chiffre la communication avec la machine distante.
⇒ protection contre les piratages du type mouchard réseau qui récupére les mots de passe

Plusieurs syntaxes de la commande « `ssh` » :

- **`ssh [-l utilisateur] nom-de-machine commande`**
- **`ssh utilisateur@nom-de-machine [commande]`**

Si l'on ne précise pas de commande, on lancera un shell en interactif.

Site pour récupérer un logiciel UNIX connaissant le protocole SSH : `http://www.openssh.org`

◇ Exemple 1 :

```
% ssh server.example.com
besancon@server.example.com's password: XXXXXXXX
Last login: Sun Oct 12 15:20:16 2003 from ppp-3
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995
No mail.
server%
```

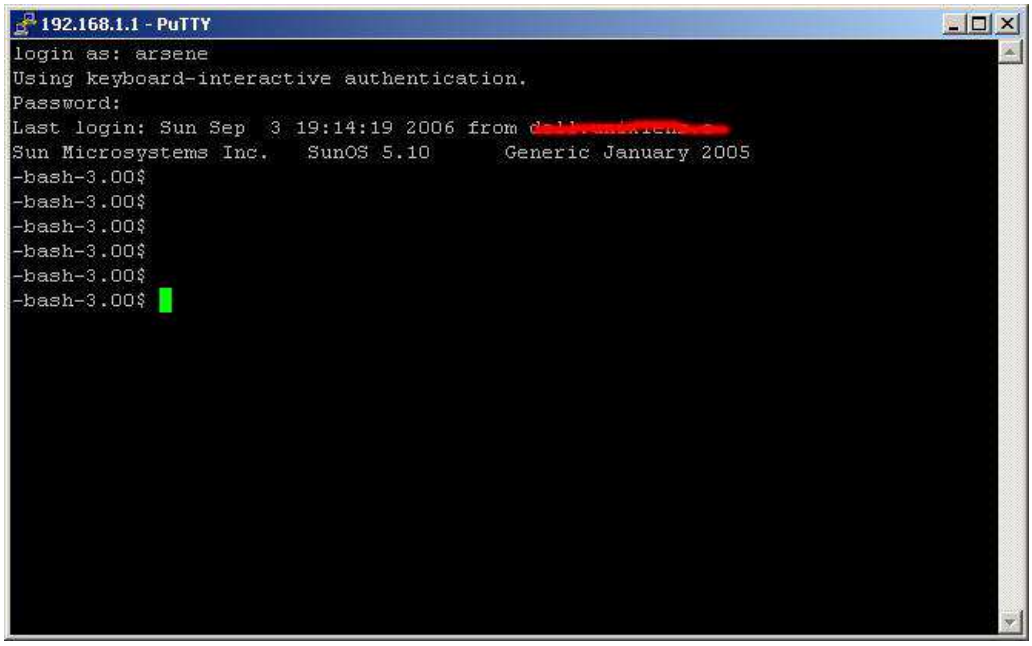
◇ Exemple 2 :

```
% ssh server.example.com date
Password: XXXXXXXX
Fri Jun  9 21:27:34 MEST 2006
```


Chapitre 12 • Commandes UNIX réseau de base
§12.10 • (Windows : : Connexion à distance interactive SSH : putty.exe)

L'utilitaire PUTTY fournit les fonctionnalités du protocole SSH pour Windows.

Site : http://www.chiark.greenend.org.uk/~sgtatham/putty/



(en anglais *secure copy*)
Plusieurs syntaxes :

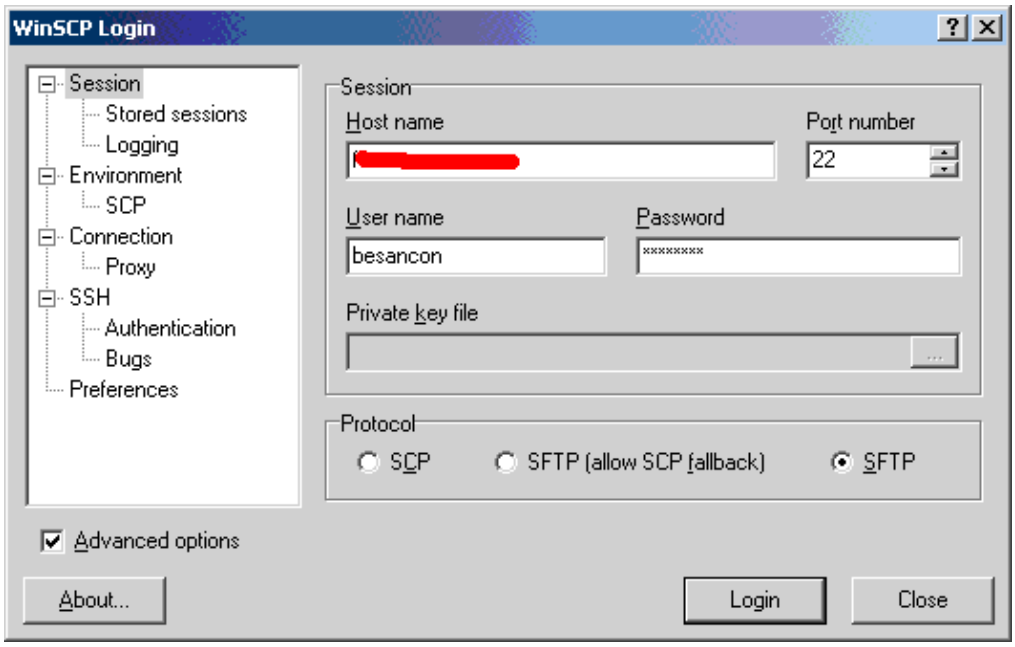
- machine distante à machine locale :
  scp user@machine :chemin-fichier1 chemin-fichier2
- machine locale à machine distante : **scp chemin-fichier1 user@machine :chemin-fichier2**

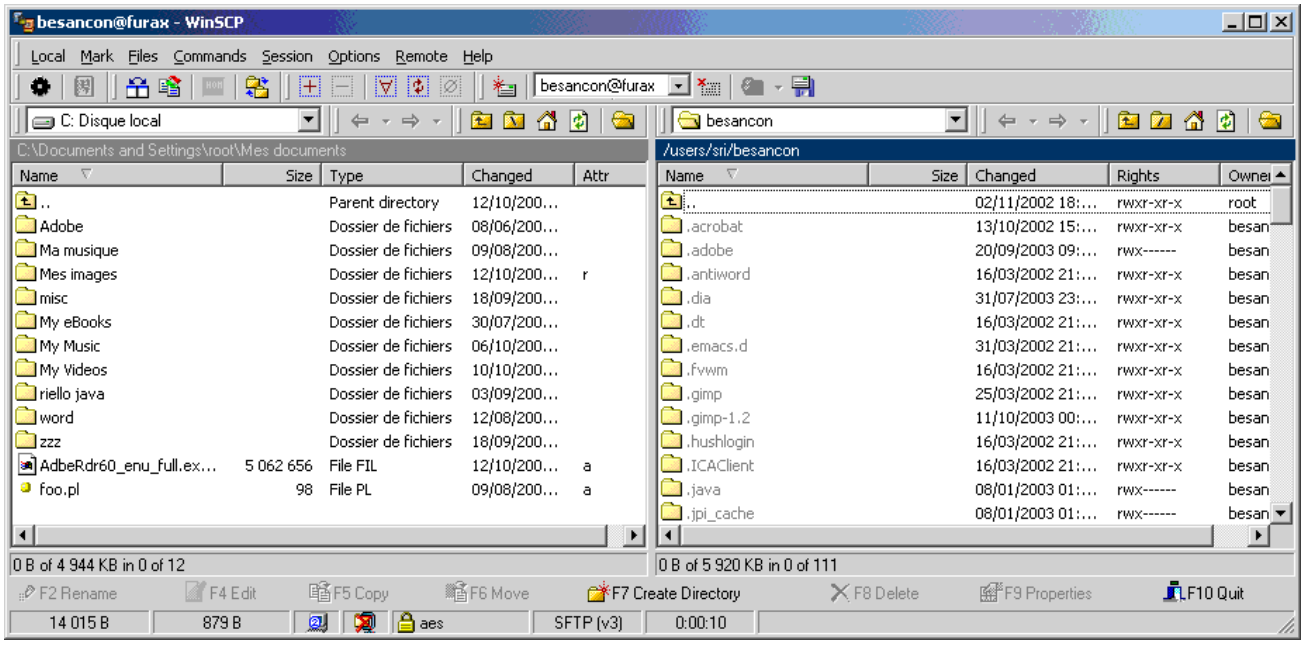
```
% scp ananas.txt besancon@server.example.com:/tmp/cerise.txt
besancon@server.example.com's password: XXXXXXXX
ananas.txt          100% |*****************************|    15        00:00

% ssh besancon@server.example.com ls -l /tmp/cerise.txt
besancon@server.example.com's password: XXXXXXXX
-rw-r--r--   1 besancon ars        15 Oct 12 15:24 /tmp/cerise.txt
```

Les logiciels implémentant SCP sont livrés avec ceux implémentant SSH
⇒ sur UNIX se reporter à http://www.openssh.org

L'utilitaire WINSCP fournit les fonctionnalités du protocole SSH pour Windows.

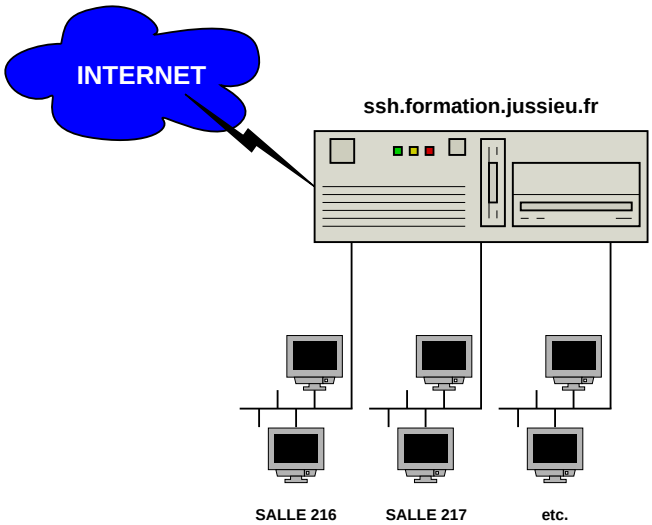
Site : http://winscp.sourceforge.net/eng/





Machine « `ssh.formation.jussieu.fr` » accessible via SSH/SCP **exclusivement**

Point de passage obligatoire depuis internet vers les machines des salles



Syntaxe : `users`

```
% users
aidan besancon fouquet kahn vogel
```


Syntaxe : **who [options]**

```
% who
kahn            ttyp0    Jul 11 22:48 (1.2.3.4)
vogel           ttyp1    Jul 10 20:33 (1.2.3.5)
aidan           ttyp2    Jul 12 00:35 (1.2.3.6)
besancon        ttyp3    Jul 12 00:46 (1.2.3.7)
fouquet         ttyp7    Jul 10 14:30 (1.2.3.8)
```

Syntaxe : **w [options]**

```
% w
USER             TTY      FROM              LOGIN@  IDLE WHAT
kahn             p0       x.example.com    Sun10PM  1:00 -bash (bash)
vogel            p1       y.example.com    Sat08PM  2:00 mutt -f mai.04
aidan            p2       z.example.com    12:35AM     9 pine
besancon         p3       t.example.com    12:46AM     - w
fouquet          p7       q.example.com    Sat02PM  1:06 -bash (bash)
```


URL = Universal Resource Locator

> *URL : Dénomination unique à caractère universel qui permet de localiser une ressource, un document, sur l'Internet, et qui indique la méthode pour y accéder, le nom du serveur et le chemin à l'intérieur du serveur.*

Typiquement :

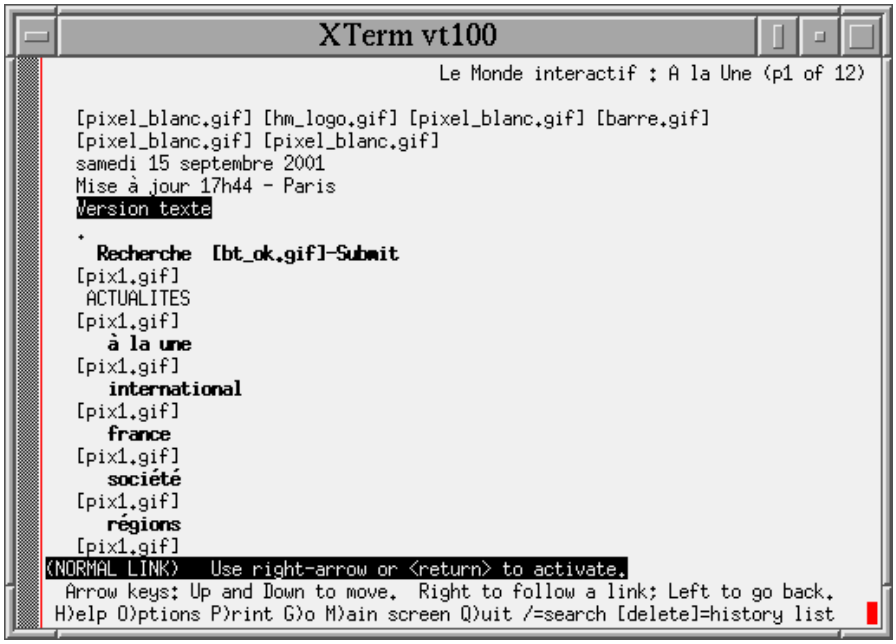
`protocole://serveur/chemin`

Les types les plus répandus :

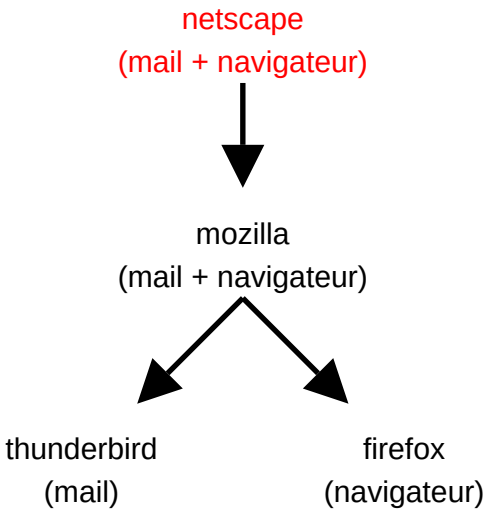
- URL de page web : « `http://www.lemonde.fr/` »
  ouvrable par Mozilla, Firefox, Opera
- URL de page ftp : « `ftp://ftp.jussieu.fr/` »
  ouvrable par Mozilla, Firefox, Opera
- URL d'adresse email :
  « `mailto:Thierry.Besancon@formation.jussieu.fr` »
  ouvrable par Mozilla, Thunderbird

`http://www.lynx.org/`

Syntaxe : **`lynx URL`**



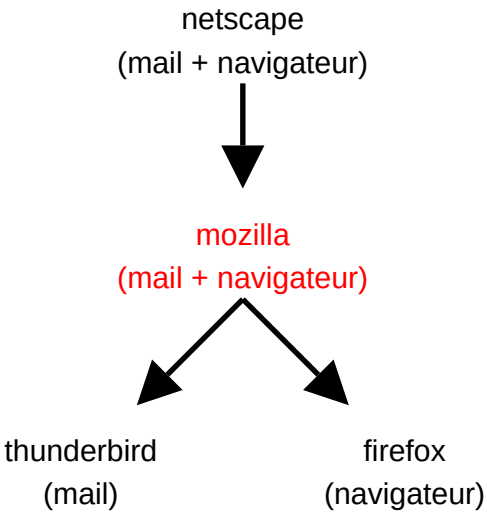
http://www.netscape.com/



Obsolète

http://www.mozilla.org/

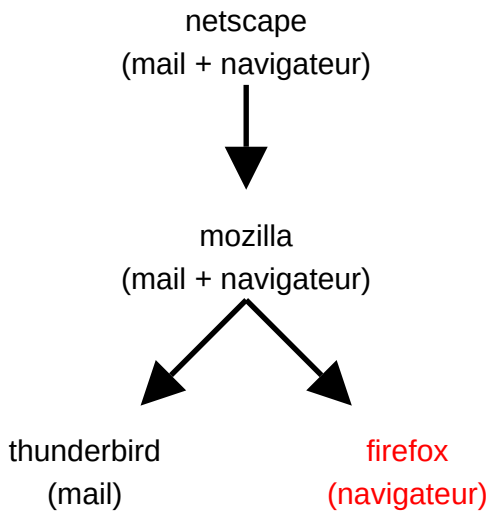
Syntaxe : **mozilla URL**



Obsolète

`http://www.getfirefox.org/`

Syntaxe : **`firefox URL`**

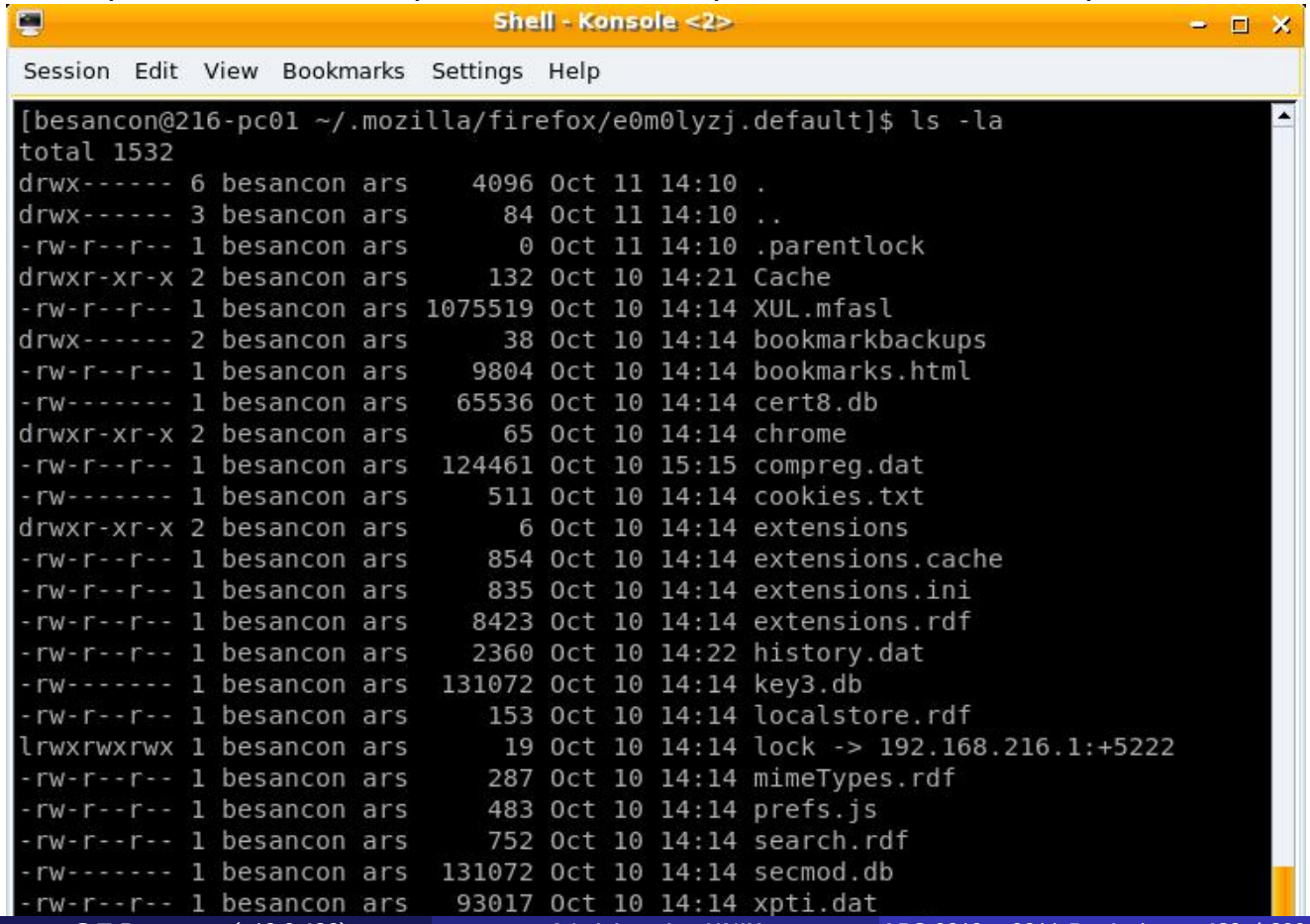


Parfois des problèmes liés à une mauvaise sortie antérieure de « `firefox` » :

Solution :

```
% rm -f $HOME/.mozilla/firefox/*/lock
% rm -f $HOME/.mozilla/firefox/*/.parentlock
```

Lorsque « firefox » quitte, les fichiers précédents doivent disparaitre :



Chapitre 12 • Commandes UNIX réseau de base

§12.22 • Navigateur Web : opera

http://www.opera.com/

Utilitaire de récupération de page web via ligne de commande.

Intérêt : permet d'automatiser des opérations de récupération de fichiers mis à disposition sur des serveurs web.

Syntaxe : **wget URL**

Site http://www.gnu.org/software/wget

Exemple : récupération d'un fichier PDF sur un site web :

```
% wget http://www.example.com/document.pdf
--01:46:59--  http://www.example.com/document.pdf
           => 'document.pdf'
Resolving www.example.com... 192.168.0.1
Connecting to www.example.com[192.168.0.1]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4,241 [application/pdf]

100%[====================================>] 4,241       --.--K

01:46:59 (28.28 MB/s) - 'document.pdf' saved [4241/4241]
```

Utilitaire de récupération de page web via ligne de commande.

Intérêt : permet d'automatiser des opérations de récupération de fichiers mis à disposition sur des serveurs web.

Plus puissant que « wget » (supporte les connexions HTTPS en particulier, les formulaires, etc.)

Syntaxe : **curl URL**

Site http://curl.haxx.se

Exemple : récupération d'un fichier de backup sur un site sécurisé avec présentation d'un certificat d'authentification :

```
% curl \
        --show-error \
        --cacert ${HOME}/certificats/ca.crt \
        --key-type PEM \
        --key ${HOME}/certificats/thierry.besancon@example.com.key \
        --cert-type PEM \
        --cert ${HOME}/certificats/thierry.besancon@example.com.crt \
        --url https://www.example.com/backup/20060804.sql.gz \
        --output backup-20060804.sql.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   296  100   296    0     0    361      0 --:--:-- --:--:-- --:--:-- 12869
   2 backup-20060804.sql.gz
```

Votre adresse de courrier électronique est du type :

`login@formation.jussieu.fr`

Même mot de passe que le login UNIX !

Attention : votre compte électronique expirera le 1er octobre de l'an prochain.

Informations pour utiliser la messagerie de la Formation Permanente :

- protocole IMAP + sécurisation par SSL
- serveur depuis lequel lire le courier :
  « `mailhost.formation.jussieu.fr` »
  ATTENTION : valable depuis tout internet
- serveur vers lequel envoyer le courier :
  « `mailhost.formation.jussieu.fr` »
  ATTENTION : uniquement valable depuis les salles de TP de la Formation Permanente
  Utiliser le serveur SMTP de votre FAI.
- nom de connexion : login UNIX
- mot de passe : celui du login UNIX

Le fichier « `$HOME/.forward` » sert au renvoi du courrier électronique vers une autre adresse. On y précise les adresses vers lesquelles renvoyer.

◇ Exemple 1 :

Pour rediriger par exemple vers `besancon@example.com` :

```
besancon@example.com
```

◇ Exemple 2 :

Pour garder une copie locale (par exemple je suis `besancon@formation.jussieu.fr`) et quand même renvoyer vers une autre adresse (par exemple `besancon@example.com`) :

```
\besancon
besancon@example.com
```


Commandes de base historiques : « `mail` » ou « `mailx` » ou « `Mail` »

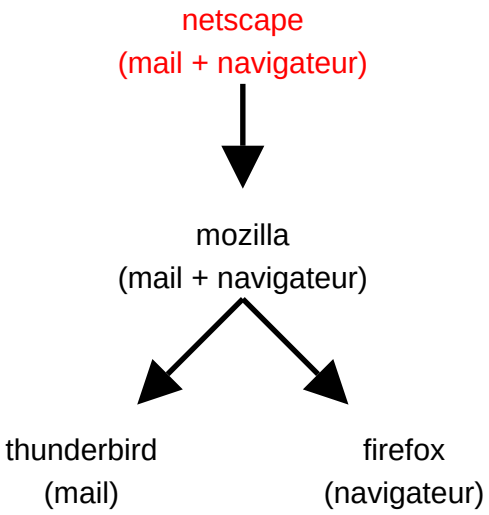
```
% Mail Thierry.Besancon@formation.jussieu.fr
Subject: test
Cc:
test
.
EOT
```

Utilisation pratique : envoi d'un fichier texte à quelqu'un :

```
mail -s "sujet du mail" adresses < contenu-mail.txt
```

(voir page 562 pour la redirection)

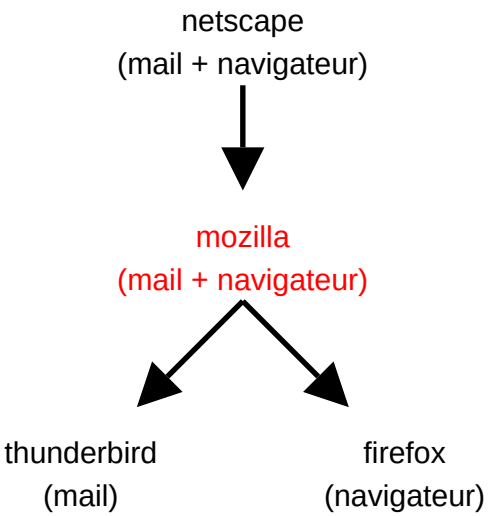
http://www.netscape.com/



Affichage graphique de vos mails. Consommateur de ressources.

Obsolète

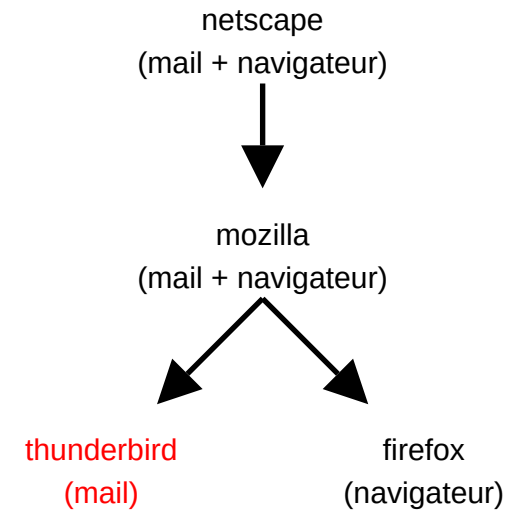
http://www.mozilla.org/



Affichage graphique de vos mails. Consommateur de ressources.

Obsolète

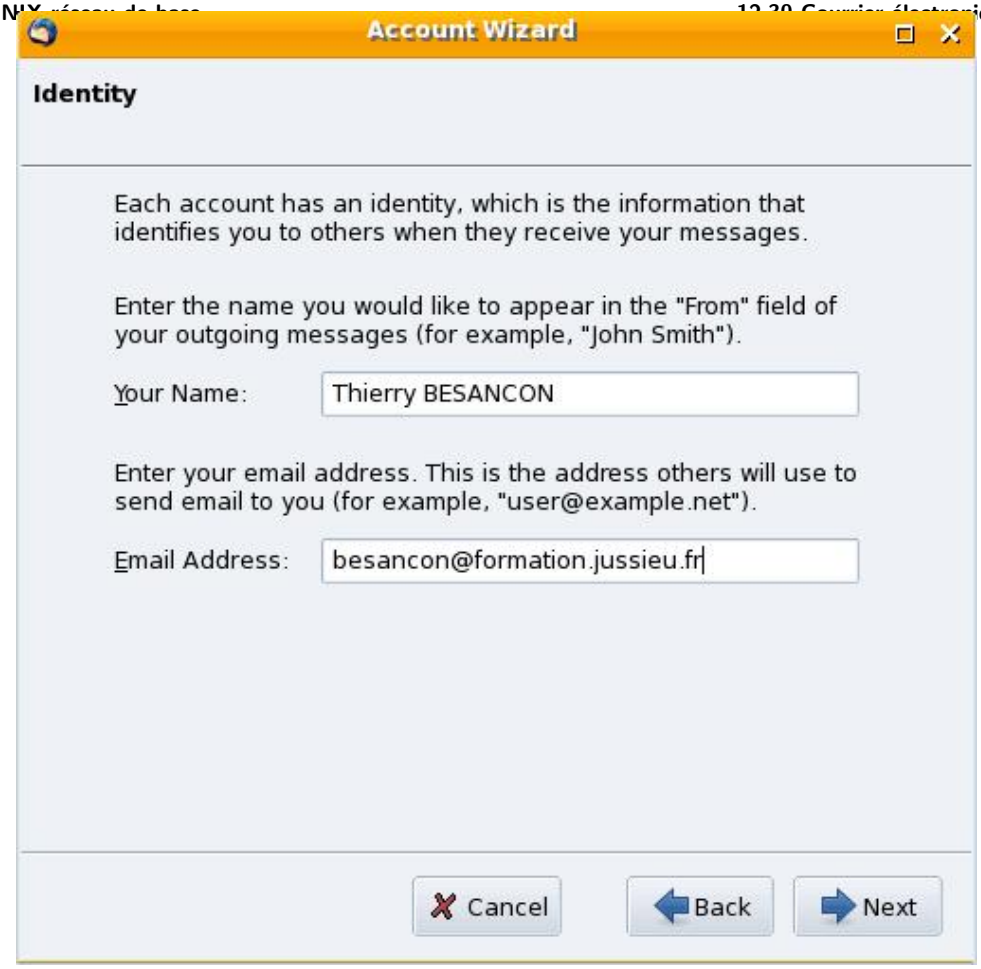
http://www.getthunderbid.org/

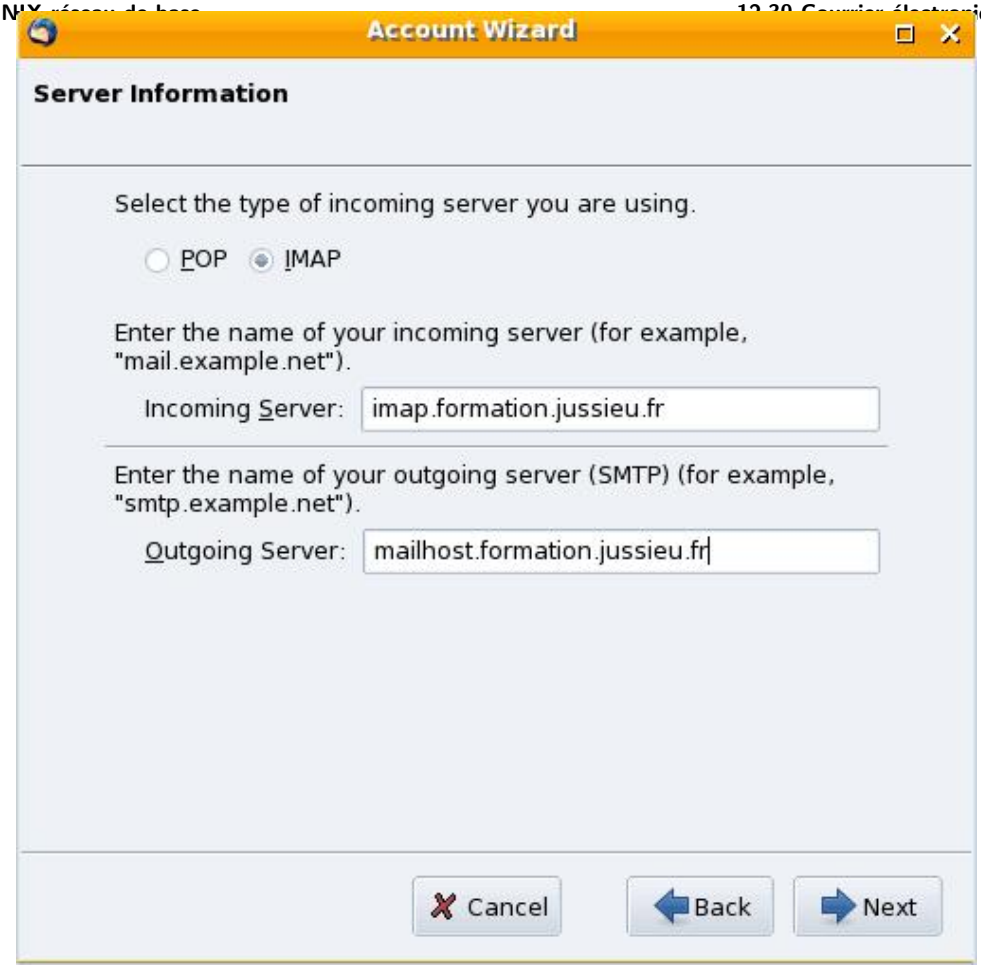


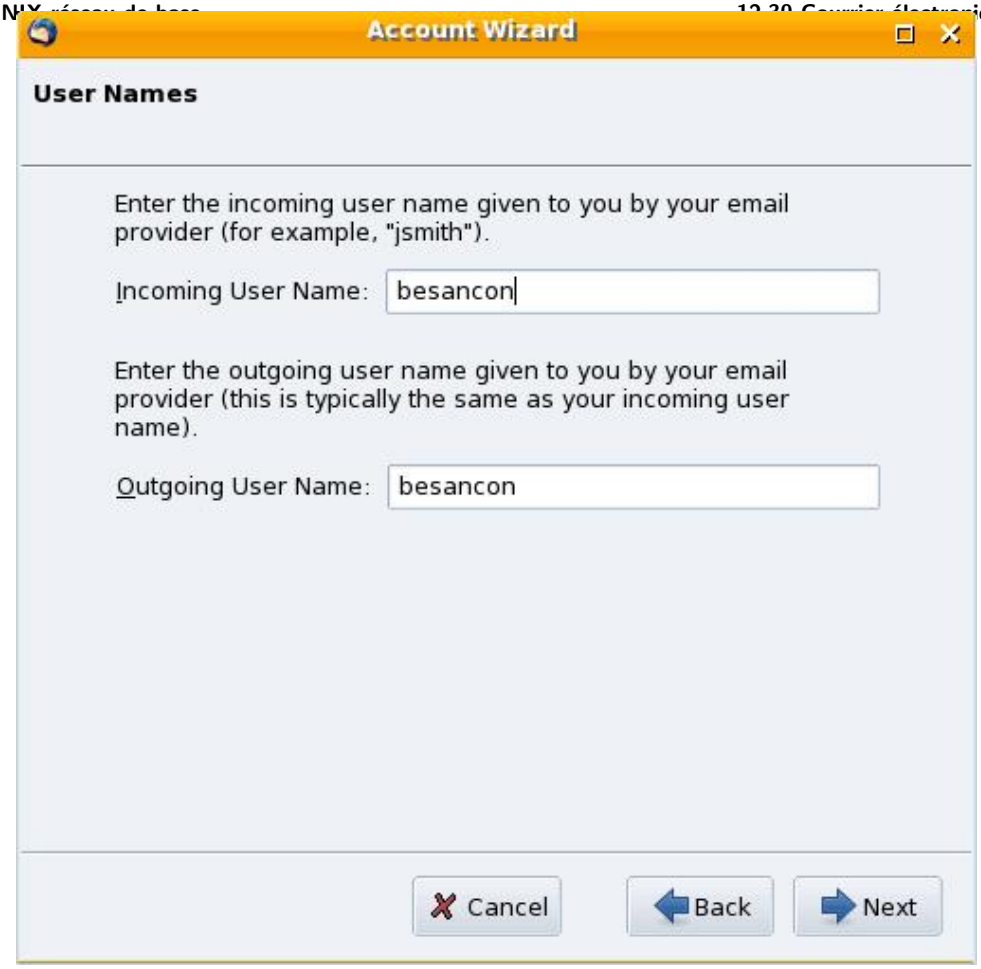
Affichage graphique de vos mails. Consommateur de ressources.

Préférer « thunderbird » à « mozilla » à « netscape ».

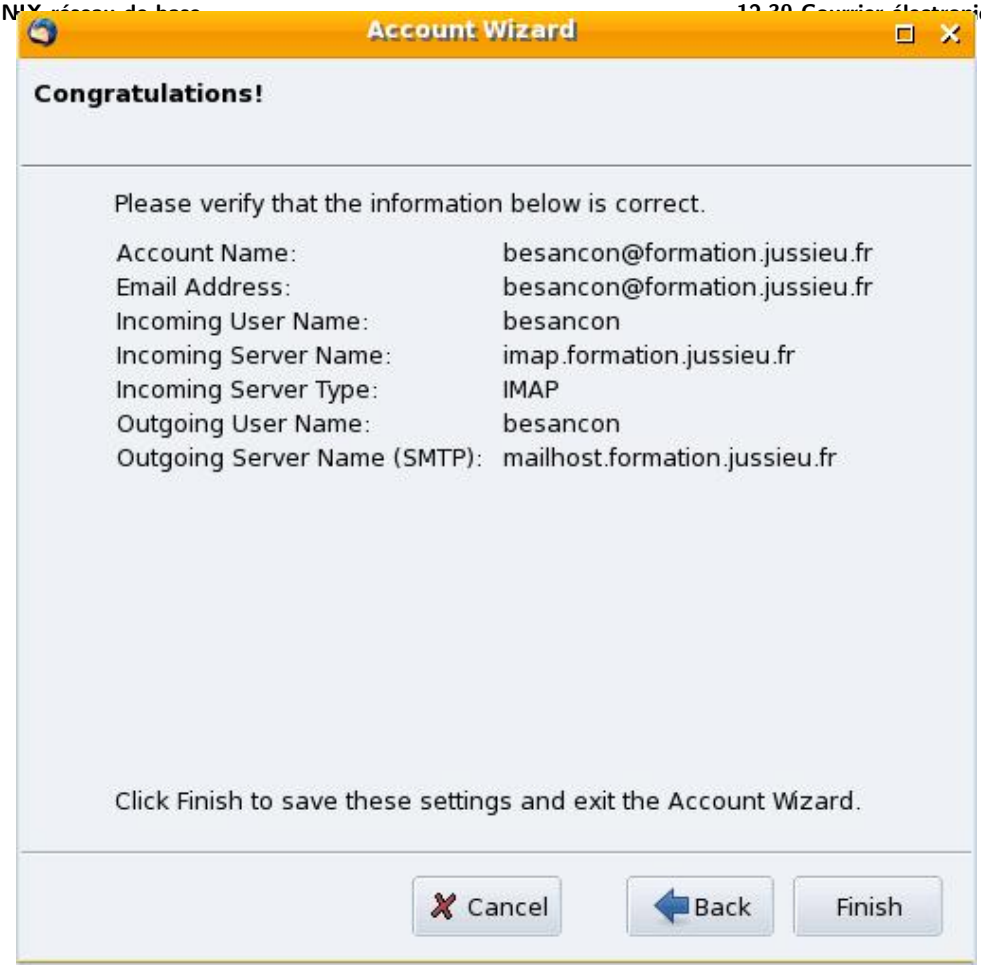
◇ Configuration de thunderbird


Account Wizard
Identity
Each account has an identity, which is the information that identifies you to others when they receive your messages.
Enter the name you would like to appear in the "From" field of your outgoing messages (for example, "John Smith").
Your Name:
Thierry BESANCON
Enter your email address. This is the address others will use to send email to you (for example, "user@example.net").
Email Address:
besancon@formation.jussieu.fr
Cancel
Back
Next

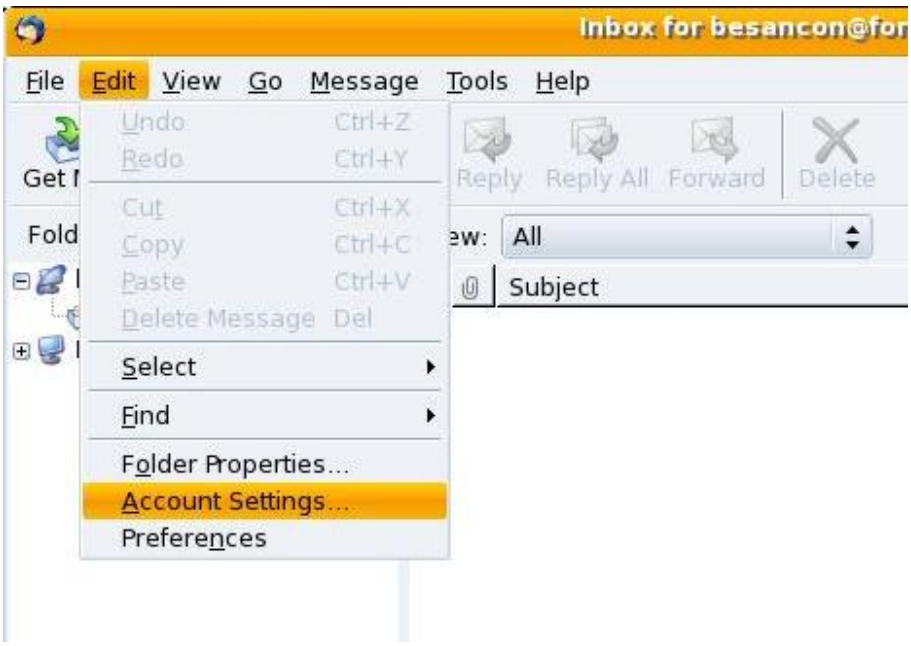

Account Wizard
Server Information
Select the type of incoming server you are using.
POP
IMAP
Enter the name of your incoming server (for example, "mail.example.net").
Incoming Server:
imap.formation.jussieu.fr
Enter the name of your outgoing server (SMTP) (for example, "smtp.example.net").
Outgoing Server:
mailhost.formation.jussieu.fr
Cancel
Back
Next

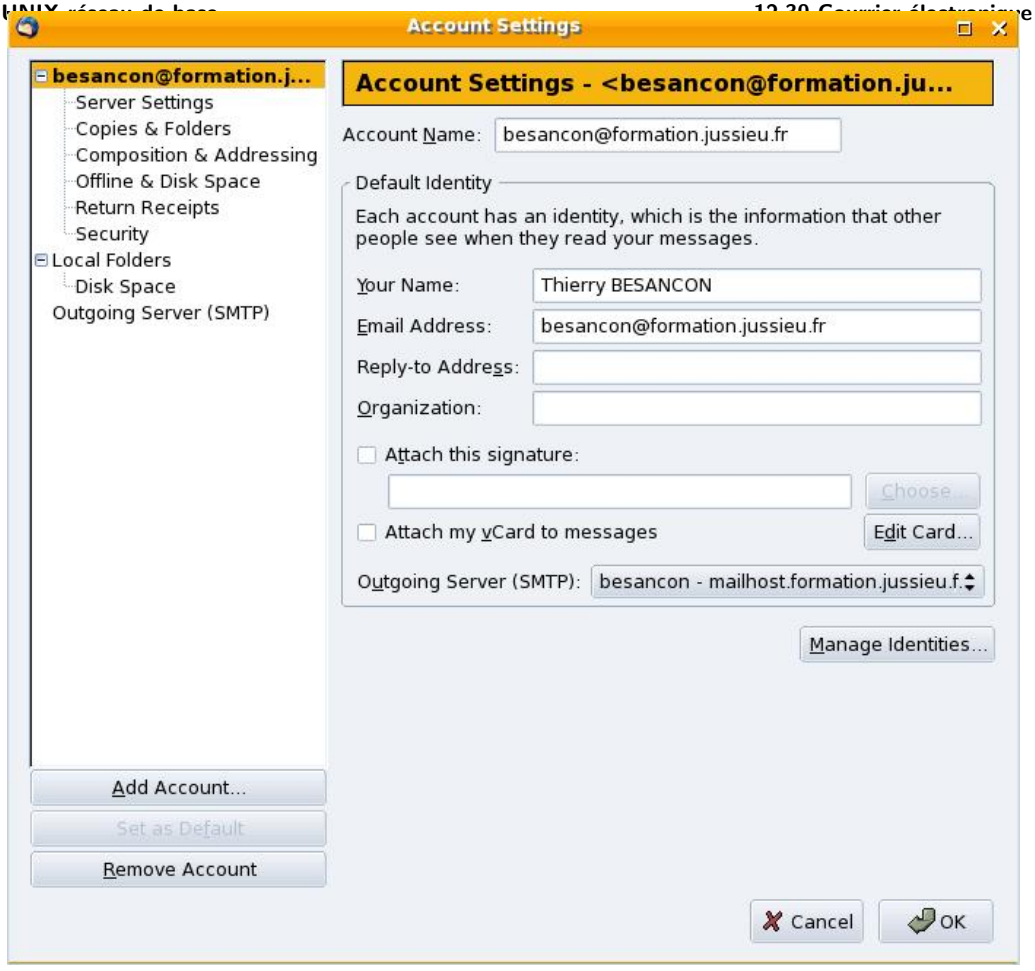

Account Wizard
User Names
Enter the incoming user name given to you by your email provider (for example, "jsmith").
Incoming User Name: besancon
Enter the outgoing user name given to you by your email provider (this is typically the same as your incoming user name).
Outgoing User Name: besancon
Cancel
Back
Next

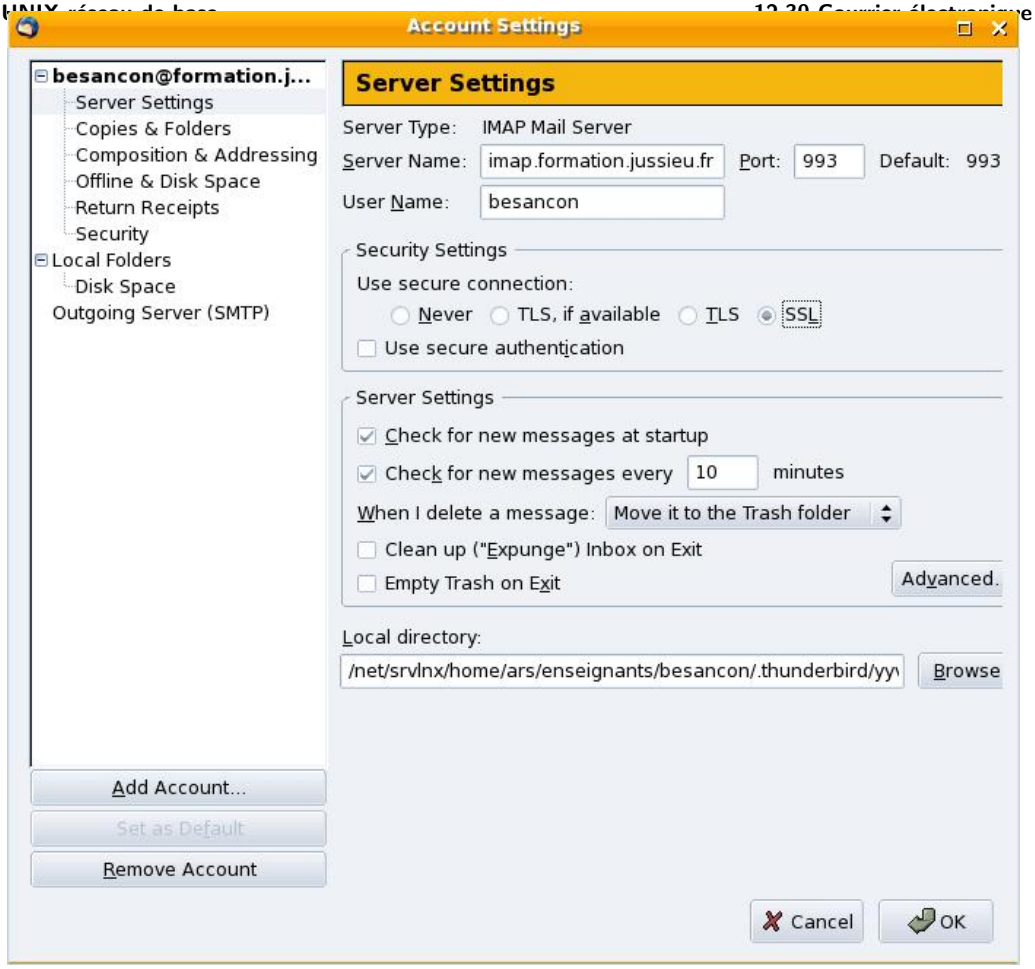

Account Wizard
Account Name
Enter the name by which you would like to refer to this account (for example, "Work Account", "Home Account" or "News Account").
Account Name: besancon@formation.jussieu.fr
Cancel
Back
Next


Account Wizard
Congratulations!
Please verify that the information below is correct.
Account Name: besancon@formation.jussieu.fr
Email Address: besancon@formation.jussieu.fr
Incoming User Name: besancon
Incoming Server Name: imap.formation.jussieu.fr
Incoming Server Type: IMAP
Outgoing User Name: besancon
Outgoing Server Name (SMTP): mailhost.formation.jussieu.fr
Click Finish to save these settings and exit the Account Wizard.
Cancel
Back
Finish


Alert
Could not connect to mail server imap.formation.jussieu.fr; the connection was refused.
OK

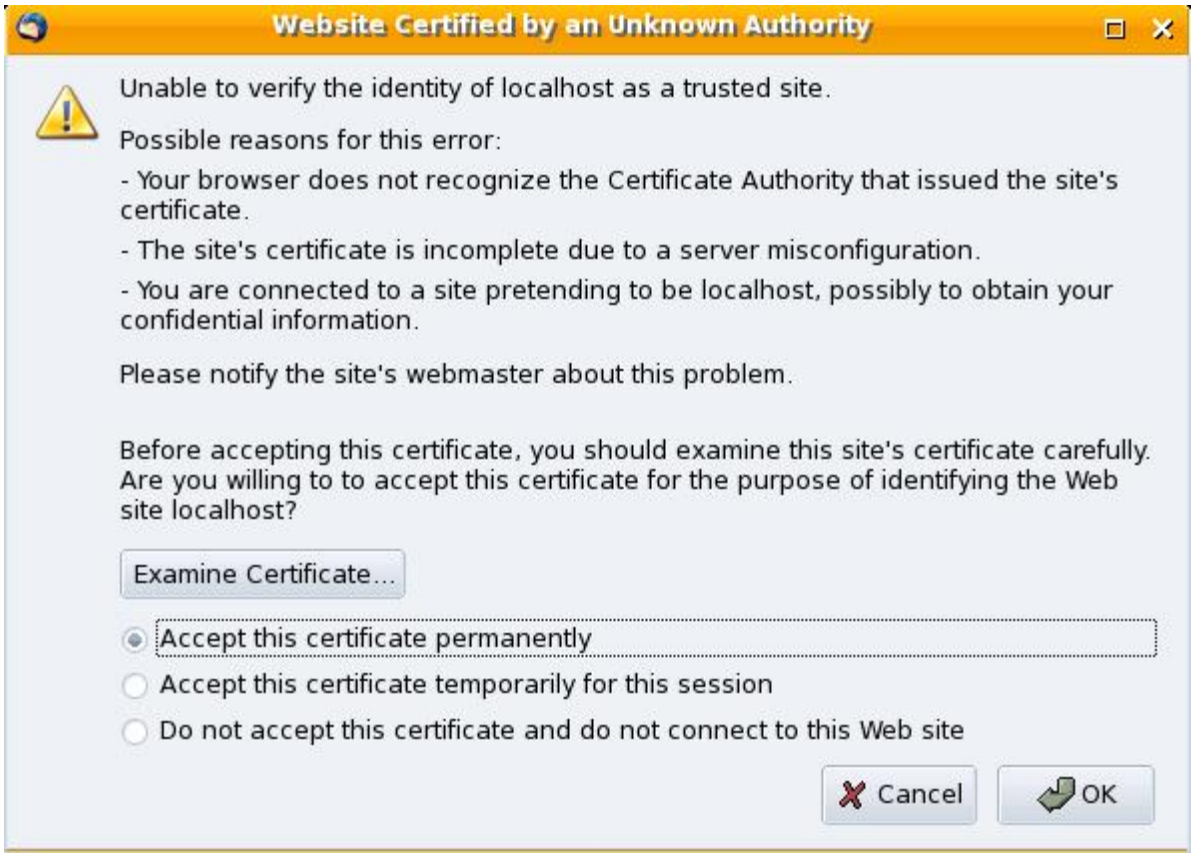

Inbox for besancon@for
File Edit View Go Message Tools Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Message Del
Select
Find
Folder Properties...
Account Settings...
Preferences
Reply
Reply All
Forward
Delete
All
Subject


Account Settings
besancon@formation.j...
Server Settings
Copies & Folders
Composition & Addressing
Offline & Disk Space
Return Receipts
Security
Local Folders
Disk Space
Outgoing Server (SMTP)
Add Account...
Set as Default
Remove Account
Account Settings - <besancon@formation.ju...
Account Name: besancon@formation.jussieu.fr
Default Identity
Each account has an identity, which is the information that other people see when they read your messages.
Your Name: Thierry BESANCON
Email Address: besancon@formation.jussieu.fr
Reply-to Address:
Organization:
Attach this signature:
Choose...
Attach my vCard to messages
Edit Card...
Outgoing Server (SMTP): besancon - mailhost.formation.jussieu.f...
Manage Identities...
Cancel
OK


Account Settings
besancon@formation.j...
Server Settings
Copies & Folders
Composition & Addressing
Offline & Disk Space
Return Receipts
Security
Local Folders
Disk Space
Outgoing Server (SMTP)
Add Account...
Set as Default
Remove Account
Server Settings
Server Type: IMAP Mail Server
Server Name: imap.formation.jussieu.fr
Port: 993
Default: 993
User Name: besancon
Security Settings
Use secure connection:
Never
TLS, if available
TLS
SSL
Use secure authentication
Server Settings
Check for new messages at startup
Check for new messages every 10 minutes
When I delete a message: Move it to the Trash folder
Clean up ("Expunge") Inbox on Exit
Empty Trash on Exit
Advanced...
Local directory:
/net/srvlnx/home/ars/enseignants/besancon/.thunderbird/yy
Browse
Cancel
OK

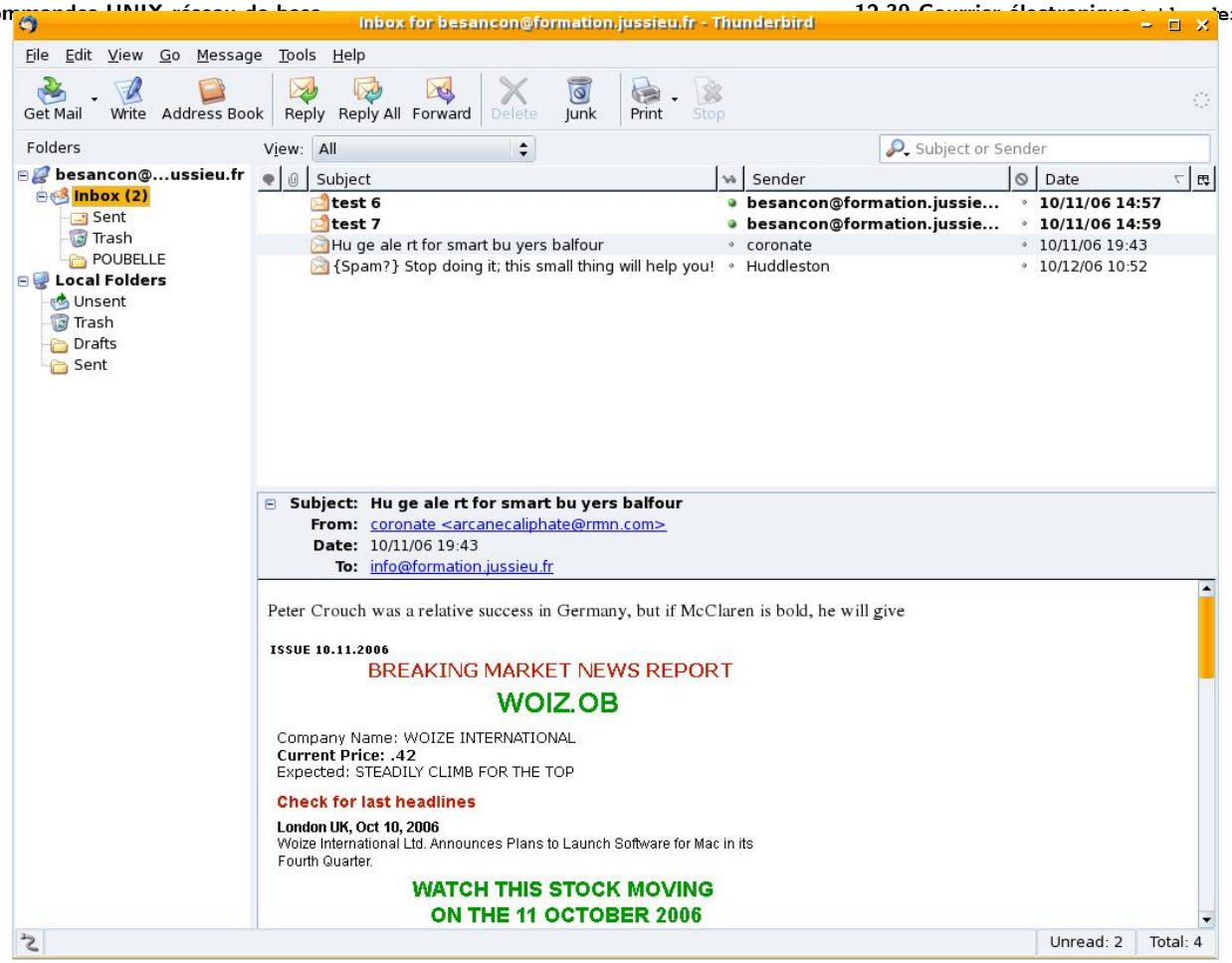

Account Settings
besancon@formation.j...
Server Settings
Copies & Folders
Composition & Addressing
Offline & Disk Space
Return Receipts
Security
Local Folders
Disk Space
Outgoing Server (SMTP)
Add Account...
Set as Default
Remove Account
Outgoing Server (SMTP) Settings
Although you can specify more than one outgoing server (SMTP), this is only recommended for advanced users. Setting up multiple SMTP servers can cause errors when sending messages.
besancon - mailhost.formation.jussieu.fr (Default)
Add...
Edit...
Remove
Set Default
Description: <not specified>
Server Name: mailhost.formation.jussieu.fr
Port: 0
User Name: besancon
Secure Connection: None
Cancel
OK


SMTP Server
Settings
Description:
Server Name: mailhost.formation.jussieu.fr
Port: 25 Default: 25
Security and Authentication
Use name and password
User Name: besancon
Use secure connection:
No TLS, if available TLS SSL
Cancel
OK


Website Certified by an Unknown Authority
Unable to verify the identity of localhost as a trusted site.
Possible reasons for this error:
- Your browser does not recognize the Certificate Authority that issued the site's certificate.
- The site's certificate is incomplete due to a server misconfiguration.
- You are connected to a site pretending to be localhost, possibly to obtain your confidential information.
Please notify the site's webmaster about this problem.
Before accepting this certificate, you should examine this site's certificate carefully. Are you willing to to accept this certificate for the purpose of identifying the Web site localhost?
Examine Certificate...
Accept this certificate permanently
Accept this certificate temporarily for this session
Do not accept this certificate and do not connect to this Web site
Cancel
OK


Security Error: Domain Name Mismatch
You have attempted to establish a connection with "imap.formation.jussieu.fr". However, the security certificate presented belongs to "localhost". It is possible, though unlikely, that someone may be trying to intercept your communication with this web site.
If you suspect the certificate shown does not belong to "imap.formation.jussieu.fr", please cancel the connection and notify the site administrator.
View Certificate
Cancel
OK


Mail Server Password Required
Enter your password for besancon@imap.formation.jussieu.fr:
****
Use Password Manager to remember this password.
Cancel
OK



Chapitre 12 • Commandes UNIX réseau de base

§12.31 • Courrier électronique : webmail de la Formation Permanente

Le site web de la formation permanente offre un service de webmail.

Se reporter à « `http://www.formation.jussieu.fr/` » ou à « `https://webmail.formation.jussieu.fr/` »

Chapitre 13

Pratique du Bourne shell

(en anglais *echo*)
Syntaxe : « `echo caractères` »

◇ Exemples 1

```
% echo quelquechose
quelquechose

% echo "quelquechose"
quelquechose
```

```
% echo "un deux"
un deux

% echo "un  deux"     "trois"
un  deux trois
```

◇ Exemples 2

```
% echo ""
<- ligne vide

% echo
<- ligne vide
```


1. attente d'une entrée de commande ;
2. traitement des caractères spéciaux de la commande ;
3. recherche de l'exécutable ;
   s'il n'est pas trouvé, on affiche un message d'erreur et le shell reprend à l'étape 1 ;
4. `fork()` + `exec()` de la commande à lancer ;
5. le shell fait un `wait()` de la commande ;
6. une fois la commande terminée, le shell reprend à l'étape 1.

Voir page 553 et page 554 aussi.

Caractères	**Signification**
tabulation, espace	appelés *white characters* ; délimiteurs des mots ; un tel caractère au minimum
retour charriot	fin de la commande à exécuter
`;`	séparateur de commandes sur une seule ligne
`( )`	exécution des commandes dans un sous shell
`{ }`	exécution des commandes en série
`&`	lance une commande en tâche de fond
`' " \`	appelés *quote characters* ; changent la façon dont le shell interpréte les caractères spéciaux
`< > << >> ` ` | `	caractères de redirection d'entrées/sorties
`* ? [ ] [^ ]`	caractères de substitution de noms de fichiers
`$`	valeur d'une variable

La difficulté liée aux métacaractères : étape 2 de la page 513 :

- Que garde le shell pour lui ?
- Que donner à l'application ?

Délimiteurs des mots de la ligne de commande : espace, tabulation

Un caractère délimiteur au minimum entre chaque mot de la ligne de commande.

Le retour charriot valide la ligne entrée.

La présence du « \ » neutralise (voir page 543) le sens spécial du métacaractère retour charriot.
Exemple :

```
% echo ananas \
>>banane \
>>cerise
ananas banane cerise
```


Le point-virgule sert à lancer plusieurs commandes UNIX sur la même ligne de commande.

Exemple :

```
% date ; ls -l exemple.txt
Thu Jul  8 19:49:19 MEST 2004
-rw-r--r--   1 besancon ars    87 Jul  6 19:25 exemple.txt
```

◇ Retour sur find :

Suite au sens de métacaractères de « ; » dans le shell, on écrit :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

La présence du « \ » neutralise (voir page 543) le sens spécial du métacaractère « ; ».

Les parenthèses délimitent un bloc de commandes exécutées dans un second shell.
Exemple 1 :

```
% pwd ; ( cd /tmp ; ls -l ; pwd ) ; pwd
/home/besancon
total 2560
-rw-r--r--  1 besancon adm    7824 Oct  9 08:53 20060476.pdf
-rw-------  1 besancon adm     419 Sep 12 00:55 Acro000F5aivb
-rw-r--r--  1 besancon adm   10799 Oct  5 00:04 fvwmrcICqrb
drwxr-xr-x  2 besancon adm     117 Sep 12 00:46 hsperfdata_besanc
-rw-------  1 root     root   3533 Sep 11 21:13 pg_hba.conf
-rw-r--r--  1 root     root   1914 Sep 14 00:09 php.errors
-rw-------  1 root     root  13666 Sep 11 21:13 postgresql.conf
drwx------  2 besancon adm     183 Jul 27 19:43 ssh-yXbXQ635
/tmp
/home/besancon
```

Le changement de directory se fait dans le second shell.

Exemple 2 :

```
% a=3 ; echo $a ; ( a=5 ; echo $a ) ; echo $a
3
5
3
```

La valeur 5 est affectée à la variable « a » du second shell et disparait avec lui.

Voir les explications sur les variables page 586.

◇ Retour sur find :

Suite au sens de métacaractères de « ( ) » dans le shell, on écrit :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

La présence du « \ » neutralise (voir page 543) le sens spécial des métacaractères « ( ) ».

◇ Avant plan

Lorsqu'une commande est en train de s'exécuter, le shell ne rend pas la main et attend que la commande se termine (correctement ou incorrectement).

On parle ainsi d'avant plan (en anglais *foreground*).

Pour interrompre prématurément une commande : taper sur la touche Ctrl **et aussi** sur la touche C du clavier. Cela tue la commande qui tournait.

On notera l'appui sur ces 2 touches par « `Ctrl-C` » ou par « `^C` ».

◇ Arrière plan

Si l'on veut une lancer une commande et récupérer la main tout de suite, avant même que la commande ait fini de s'exécuter, il faut lancer la commande par :

```
% commande &
```

On parle ainsi d'arrière plan (en anglais *background*).

Le signe « `&` » signifie de lancer en **tâche de fond**, en **background** la commande. Sans ce signe, la commande est lancée en **premier plan**, en **foreground**.

◇ Passage d'avant plan en arrière plan

Pour passer en background une commande lancée en foreground :

1. Figer la commande en cours.
   Taper sur la touche « Control » **et aussi** sur la touche « Z », soit « Ctrl-Z » ou « ^Z » :

```
% commande
...
^Z
[1]+  Stopped                 commande
```

2. Indiquer de l'exécuter dorénavant en background.
   Taper la commande « **bg** » (en anglais *background*) :

```
% bg
[1]+ commande &
```


D'une façon générale, un débutant UNIX doit proscrire l'utilisation de « Ctrl-Z ».

Dans 9 cas sur 10, c'est « Ctrl-C » qu'il doit employer. On évitera ainsi une saturation de la machine avec des commandes suspendues et en attente d'être tuées ou relancées.

◇ Passage d'arrière plan en avant plan

Pour passer en foreground une commande lancée en background :

1. La commande est lancée.
   On a la main :

   ```
   % commande &
   ...
   ```

2. Indiquer de l'exécuter dorénavant en foreground.
   Taper la commande « **fg** » (en anglais *foreground*) :

   ```
   % fg
   ```

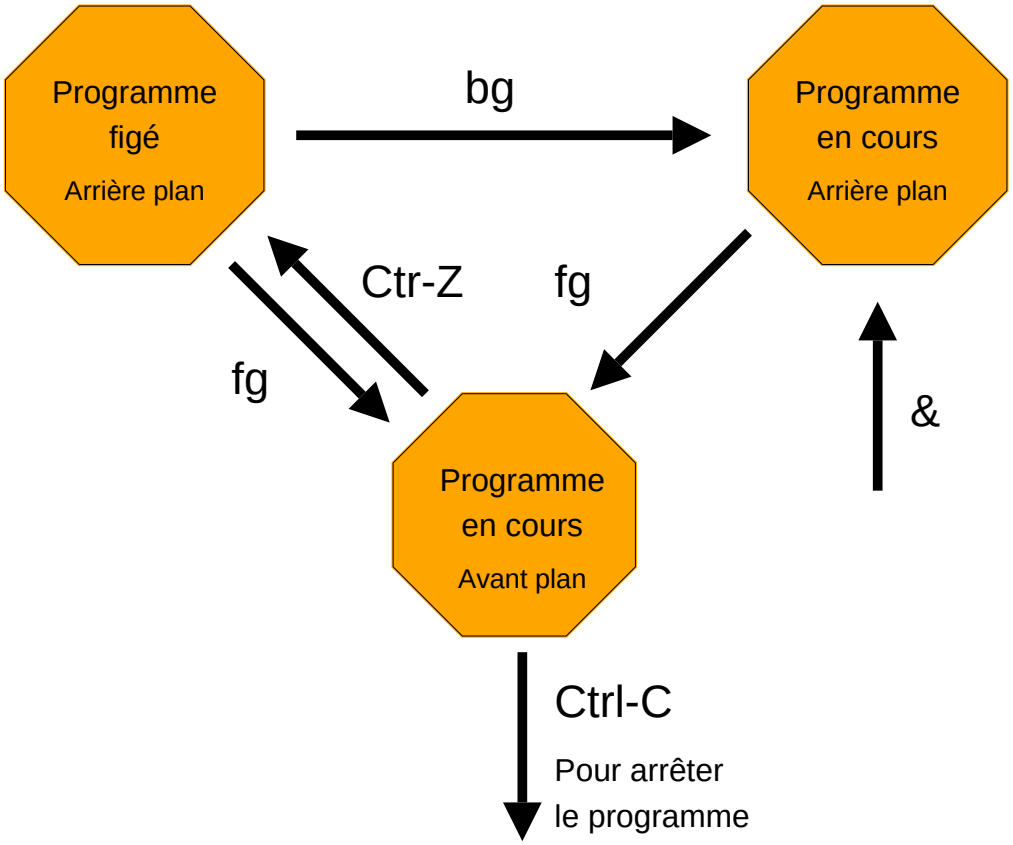

◇ Tuer un processus en arrière plan

Pour tuer la commande en background : « `kill %1` »

```
% commande &
% kill %1
[1]+  Terminated                 commande &
```

En fait, voir la forme générale page 531.

◇ En résumé

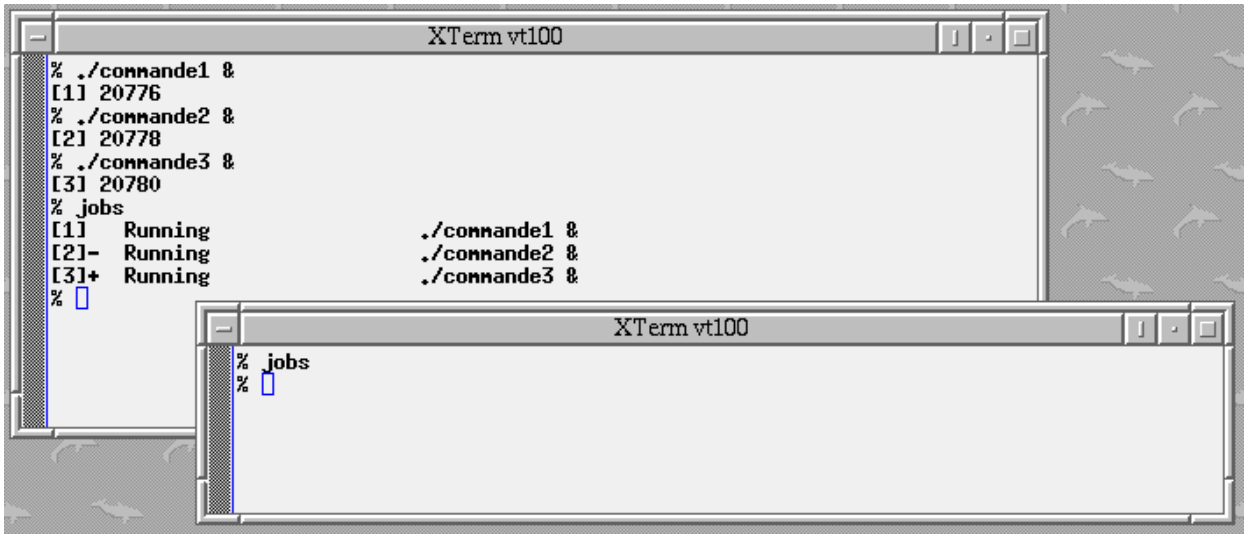


◇ Liste des processus en arrière plan

Pour connaitre la liste des commandes en background :

```
% jobs
[1]   Running                 commande1 &
[2]   Running                 commande2 &
[3]   Running                 commande3 &
[4]-  Running                 commande4 &
[5]+  Running                 commande5 &
```

On peut avoir plusieurs commandes en background. D'où une numérotation des commandes qui sont affichées. Ce numéro peut être repris dans les commandes « `fg` » et « `bg` » ainsi que dans la commande suivante, « `kill` ».



◇ Comment manipuler les processus en background quand il y en a plusieurs ?

Syntaxes générales :

- Syntaxe « `fg %n` »
- Syntaxe « `bg %n` »
- Syntaxe « `kill %n` »

avec n un des numéros trouvés grâce à « `jobs` ».

Par exemple, pour tuer une commande en background :

```
% jobs
[1]   Running                 commande1 &
[2]   Running                 commande2 &
[3]   Running                 commande3 &
[4]-  Running                 commande4 &
[5]+  Running                 commande5 &
% kill %3
[3]   Terminated              commande3 &
% jobs
[1]   Running                 commande1 &
[2]   Running                 commande2 &
[4]-  Running                 commande4 &
[5]+  Running                 commande5 &
```


(en anglais *processus*)

Les commandes « `fg` », « `bg` », « `jobs` » ne fonctionnent que sur les processus lancés par le shell courant. Les commandes vues précédemment peuvent donc être inutilisables si vous avez quitté votre shell.

La commande « `ps` » plus générale permet d'avoir des informations sur tous les processus de la machine.

2 syntaxes selon l'UNIX de la machine :

- syntaxe de la famille BSD ; FreeBSD, OpenBSD, NetBSD, etc.
- syntaxe de la famille System-V ; SUN, HP, IBM, LINUX, etc.

◇ Syntaxe de la famille d'UNIX BSD

- les processus associés à son terminal : « `ps` »
- tous ses processus : « `ps -x` »
- tous les processus de la machine : « `ps -ax` »
- tous les processus de la machine avec les noms de login associés : « `ps -aux` »

Exemple (partiel) de « `ps -aux` » :

```
% ps -aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  1130   52 ?        S    Oct23   0:06 init
root         2  0.0  0.0     0    0 ?        SW   Oct23   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   Oct23   0:01 [kupdate]
...
nobody     476  0.0  0.1  1300   44 ?        S    Oct23   0:01 [identd]
daemon     490  0.0  0.0  1144    0 ?        SW   Oct23   0:00 [atd]
xfs        636  0.0  0.3  2820  120 ?        S    Oct23   0:18 xfs -droppriv -da
root     14703  0.0  0.0  2256    0 tty1     SW   Oct25   0:00 [login]
root      9813  0.0  0.0  6912    0 ?        SW   Oct31   0:09 [kdm]
idiri    20810  0.0  0.0  6552    0 ?        SW   15:13   0:01 [kwm]
idiri    20863  0.0  0.0  1996    0 pts/0    SW   15:13   0:00 [tcsh]
besancon 21785  0.0  1.3  1732  416 pts/1    S    15:25   0:00 -bash
idiri    23660  0.2  1.5  1860  472 tty2     S    16:39   0:01 vi probleme6.c
...
```


◇ Syntaxe de la famille d'UNIX System-V

- les processus associés à son terminal : « `ps` »
- tous les processus de la machine avec les noms de login associés : « `ps -edf` »

Exemple (partiel) de « `ps -edf` » :

```
% ps -edf
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0 09:09:47 ?        0:01 sched
    root     1     0  0 09:09:47 ?        0:02 /etc/init -
    root     2     0  0 09:09:47 ?        0:00 pageout
    root     3     0  0 09:09:47 ?        0:52 fsflush
    root   181     1  0 09:12:07 ?        0:06 /usr/lib/autofs/automountd
...
  daemon   283     1  0 09:12:12 ?        0:11 /usr/sbin/lpd
    root   291     1  0 09:12:13 ?        0:00 /usr/local/apache/bin/httpd
    root   296     1  0 09:12:14 ?        0:00 /usr/local/admin/lib/idled
  nobody 15130   291  0 23:30:56 ?        0:00 /usr/local/apache/bin/httpd
besancon 16463 16461  0 00:12:26 pts/0    0:00 -csh
...
```

(en anglais *kill*)

La commande « `kill` » sert à communiquer avec des processus :

- arrêt de processus
- demande au processus de se reconfigurer
- passage en mode verbeux du processus
- etc.

La commande « `kill` » existe sur tous les UNIX et il n'y a pas de différence de fonctionnement selon les UNIX.

2 syntaxes possibles :

- syntaxe numérique : « `kill -9 2878` »
- syntaxe symbolique : « `kill -KILL 2878` » (voir pages suivantes 538 et**??** pour les noms symboliques utilisables)

Nom en langage C	Nom pour la commande « kill »	Valeur	Comportement	Sens
SIGHUP	HUP	1	Exit	Hangup
SIGINT	INT	2	Exit	Interrupt
SIGQUIT	QUIT	3	Core	Quit
SIGILL	ILL	4	Core	Illegal Instruction
SIGTRAP	TRAP	5	Core	Trace or Breakpoint Trap
SIGABRT	ABRT	6	Core	Abort
SIGEMT	EMT	7	Core	Emulation Trap
SIGFPE	FPE	8	Core	Arithmetic Exception
SIGKILL	KILL	9	Exit	Killed
SIGBUS	BUS	10	Core	Bus Error
SIGSEGV	SEGV	11	Core	Segmentation Fault
SIGSYS	SYS	12	Core	Bad System Call
SIGPIPE	PIPE	13	Exit	Broken Pipe
SIGALRM	ALRM	14	Exit	Alarm Clock
SIGTERM	TERM	15	Exit	Terminated
SIGUSR1	USR1	16	Exit	User Signal 1

Nom en langage C	Nom pour la commande kill	Valeur	Comportement	Sens
SIGUSR2	USR2	17	Exit	User Signal 2
SIGCHLD	CHLD	18	Ignore	Child Status Changed
SIGPWR	PWR	19	Ignore	Power Fail or Restart
SIGWINCH	WINCH	20	Ignore	Window Size Change
SIGURG	URG	21	Ignore	Urgent Socket Condition
SIGPOLL	POLL	22	Exit	Pollable Event
SIGSTOP	STOP	23	Stop	Stopped (signal)
SIGTSTP	TSTP	24	Stop	Stopped (user)
SIGCONT	CONT	25	Ignore	Continued
SIGTTIN	TTIN	26	Stop	Stopped (tty input)
SIGTTOU	TTOU	27	Stop	Stopped (tty output)
SIGVTALRM	VTALRM	28	Exit	Virtual Timer Expired
SIGPROF	PROF	29	Exit	Profiling Timer Expired
SIGXCPU	XCPU	30	Core	CPU time limit exceeded
SIGXFSZ	XFSZ	31	Core	File size limit exceeded
SIGWAITING	WAITING	32	Ignore	Concurrency signal reserved by threads library

Les signaux les plus utiles sont :

- SIGHUP
  Cela envoie l'équivalent du « Ctrl-C » du clavier.
- SIGKILL
  Cela envoie un signal que le processus est obligé de suivre et qui se traduira inéluctablement par la mort du processus.

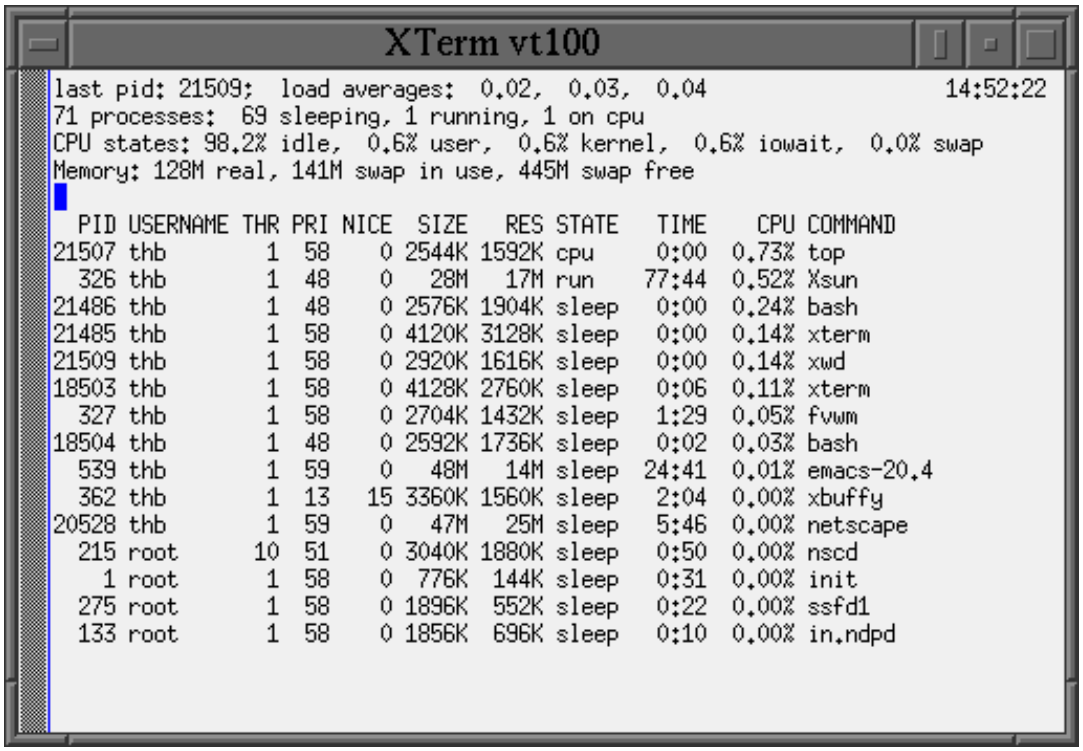
Inconvénient de « ps » : c'est la liste des processus à un instant t.

On ne pourra jamais sous UNIX avoir la liste des processus en cours : le temps de chercher les processus et de faire le rapport, certains processus peuvent avoir disparu.

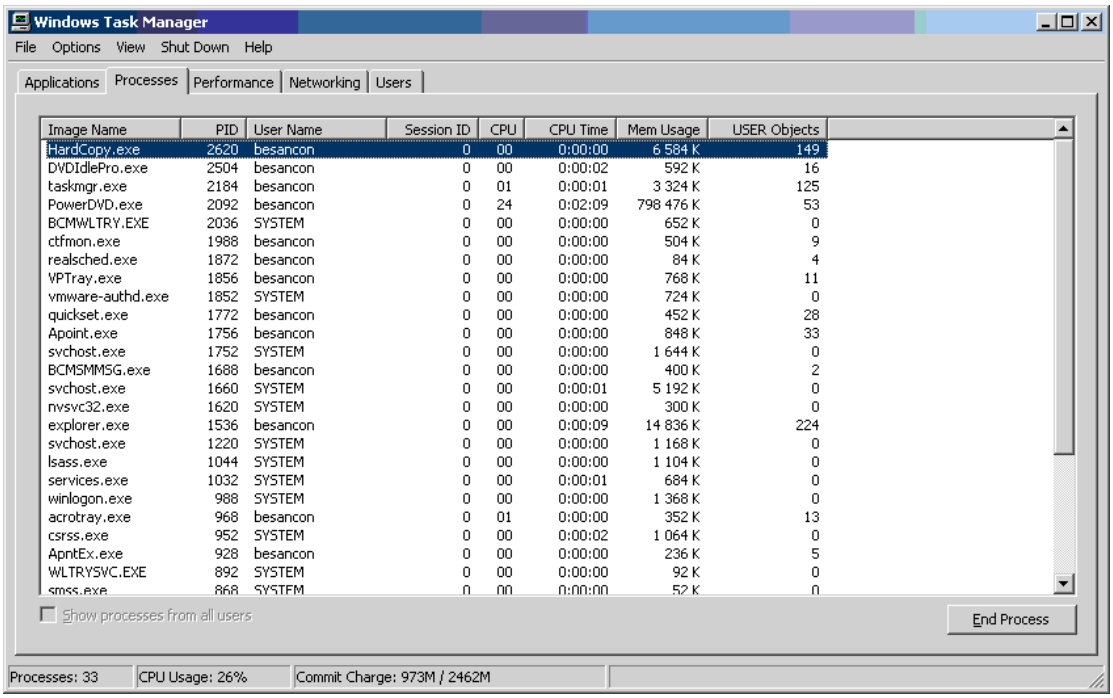
Amélioration de « ps » : la commande « top » (n'est cependant pas standard sur tous les UNIX)

Son intérêt : elle affiche une liste des processus toutes les *n* secondes

URL : ftp://ftp.groupsys.com/pub/top/



Le programme « `taskmgr.exe` » offre une fonctionnalité du genre de « `top` ».



Caractères	Nom	Description
'	single quote	le shell n'interprète aucun caractère spécial entre deux '
"	double quote	le shell n'interprète aucun caractère spécial à l'exception de « $ », « ` » et « \ »
\	backslash ou antislash	caractère d'échappement dont le sens dépend du contexte (voir page 549)

Exemples d'utilisation pour conserver les espaces dans les chaines de caractères :

1. `% echo un deux`
   `un deux`
2. `% echo "un deux"`
   `un deux`
3. `% echo 'un deux'`
   `un deux`
4. `% echo un\ \ \ \ \ deux`
   `un deux`

Exemples d'utilisation pour afficher des apostrophes :

1. `% echo c'est aujourd'hui`
   `cest aujourdhui` **<- pas d'apostrophe**
2. `% echo c\'est aujourd\'hui`
   `c'est aujourd'hui`
3. `% echo "c'est aujourd'hui"`
   `c'est aujourd'hui`

Exemples d'utilisation pour afficher des guillemets :

1. `% echo \"ananas\"`
   `"ananas"`
2. `% echo '"ananas"'`
   `"ananas"`
3. `% echo ""ananas"" <- attention aux paires de guillemets`
   `ananas`

En résumé :

1. `% echo '"'`
   `"`
2. `% echo "'"`
   `'`

Exemples de ce qui est interprété ou pas selon que l'on a des guillemets ou des apostrophes ou des backslashes :

1. `% echo "$HOME"`
   `/net/serveur/home/ars/enseignants/besancon`
2. `% echo '$HOME'`
   `$HOME`
3. `% echo "\$HOME"`
   `$HOME`
4. `% echo '\$HOME'`
   `\$HOME`

(voir page 586 pour les explications sur le symbole « $ »)

◇ Caractère « \ »

Le caractère « \ » sert à neutraliser le caractère suivant.

Les caractères neutralisés dépendront du contexte :

- hors guillemets
- entre guillemets

Hors guillemets : Le caractère « \ » neutralise **tous** les caractères.

```
% echo x\yz
xyz

% echo x\#y
x#y

% echo x\'y
x'y
```

```
% echo x\$y
x$y

% echo x\`y
x`y

% echo x\"y
x"y

% echo x\\y
x\y

% echo x\
y
xy
```


Entre guillemets : Le caractère « \ » neutralise **uniquement** les caractères « $ », « ` », « " » et « \ » et retour (« \n »).

```
% echo "x\yz"
x\yz

% echo "x\#y"
x\#y

% echo "x\'y"
x\'y
```

```
% echo "x\$y"
x$y

% echo "x\`y"
x`y

% echo "x\"y"
x"y

% echo "x\\y"
x\y

% echo "x\
y"
xy
```

◇ Retour sur grep :

Pour neutraliser tout caractère dans une regexp que le shell pourrait vouloir interpréter, on écrit :

```
% grep -E '^[a-e]' exemple.txt
```

◇ Retour sur sed :

Pour neutraliser tout caractère dans une regexp que le shell pourrait vouloir interpréter, on écrit :

```
% sed -e 's/^[a-e]//' exemple.txt
```


En interne, le shell lance une commande par la fonction « `execl()` » du langage C (après un « `fork()` ») :

```
% ls -l -----> execl("/bin/ls",
                    "ls",       <--> argv[0] <--> $0
                    "-l",       <--> argv[1] <--> $1
                    NULL
                   ) ;
```

Se reporter à la page de manuel de « `exec(2)` ».

La difficulté :

- certains caractères sont pour le shell
- certains caractères sont pour la commande invoquée
- le shell se sert toujours en premier

Conséquence :
Si un caractère spécial a un sens pour le shell et pour la commande, il faut l'écrire de façon non ambigue. Le shell doit comprendre si le caractère spécial est pour lui ou pour la commande.

◇ Exemple 1

```
% rm exemple.txt    --> execl("/bin/rm",
% ...                        "rm",
                            "exemple.txt",
                            NULL
                           );
```

◇ Exemple 2

```
% ls
fichier1.doc   fichier2.doc
% rm *.doc     --> execl("/bin/rm",
% ...                    "rm",
                        "fichier1.doc",
                        "fichier2.doc",
                        NULL
                       );
```


◇ Exemple 3

```
% ls
ananas  cerise
% expr 2 * 3  -------> execl("/usr/bin/expr",
expr: syntax error           "expr",
% ...                        "2",
                             "ananas",
                             "cerise",
                             "3",
                             NULL
                            );
```

◇ Exemple 4

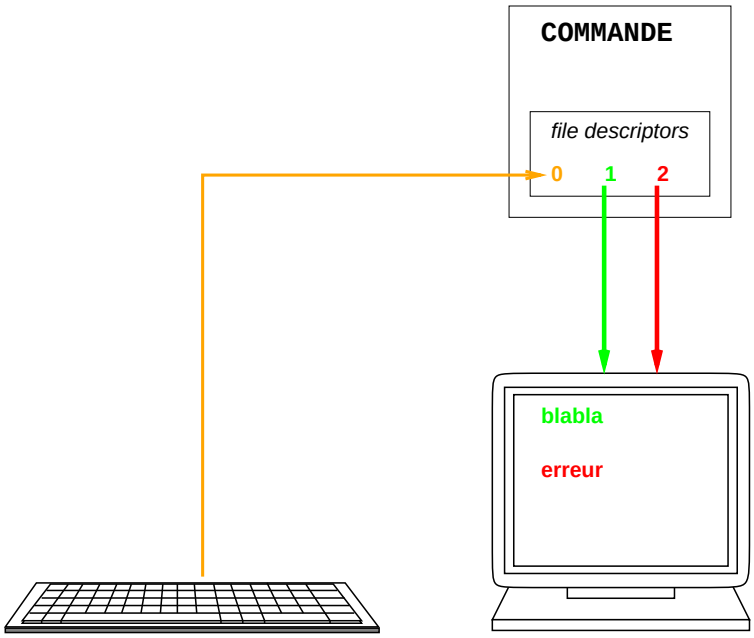
```
% ls
ananas  cerise
% expr 2 \* 3  -------> execl("/usr/bin/expr",
6                         "expr",
% ...                     "2",
                          "*",
                          "3",
                          NULL
                         );
```


Toutes les entrées/sorties d'UNIX sont réalisées au moyen de fichiers. Chaque processus ouvre donc un certain nombre de fichiers. Ces fichiers sont référencés en interne par une table d'entiers dits **file descriptors**.

Nom	File descriptor	Destination par défaut
standard input (**stdin**)	0	clavier
standard output (**stdout**)	1	écran
standard error (**stderr**)	2	écran

Les file descriptors existent en langage C et sont profondément ancrés dans le fonctionnement interne d'UNIX.

Les file descriptors par défaut se présentent ainsi :



Le fichier système de programmation C « /usr/include/stdio.h » indique :

```
#define stdin   (&__sF[0])
#define stdout  (&__sF[1])
#define stderr  (&__sF[2])
```

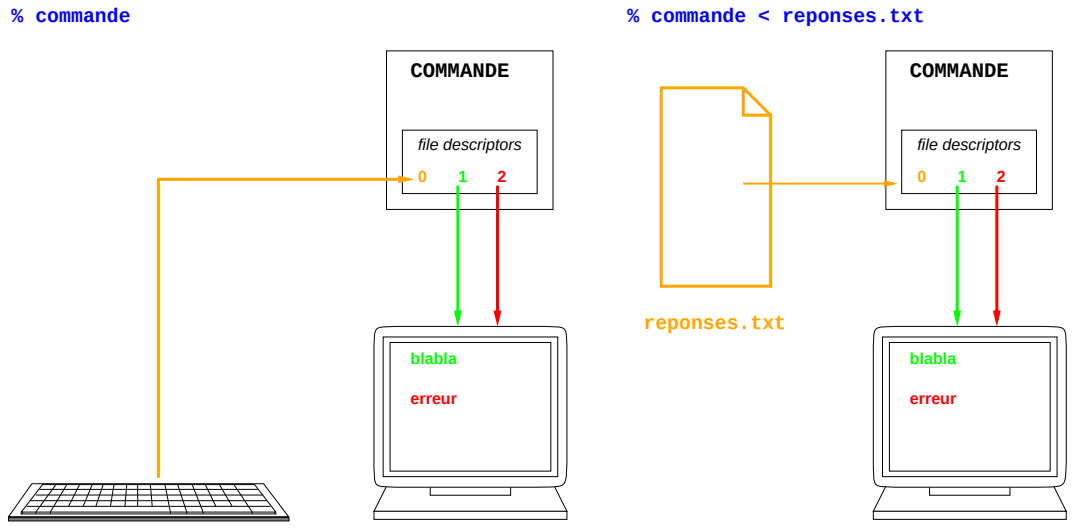
Quelques petits exemples C de démonstration :

```
#include<stdio.h>
main()
{
  char line[1024];
  (void)fgets(line, 1024, stdin);

  fprintf(stdout, "Bonjour !\n");

  fprintf(stderr, "Enfer et damnation !\n");
}
```

Canal	Format	Description
0	`commande < fichier.txt`	stdin de `commande` provient de `fichier.txt`
1	`commande > fichier.txt`	stdout de `commande` placé dans `fichier.txt` dont le contenu précédent est écrasé
1	`commande >> fichier.txt`	stdout de `commande` placé en fin de `fichier.txt`
1	`variable=`commande``	remplace `commande` par le résultat de l'exécution de `commande`
1 + 0	`commande1 | commande2`	passe le stdout de `commande1` comme stdin de `commande2`
2	`commande 2> fichier.txt`	redirige stderr de `commande` dans `fichier.txt`
2	`commande 2>> fichier.txt`	stderr de `commande` placé en fin de `fichier.txt`

◇ Redirection du file descriptor 0

Comment s'organisent les file descriptors lors de « `commande < reponses.txt` » ?



Exemples :

```
% application.exe < reponses.txt
```

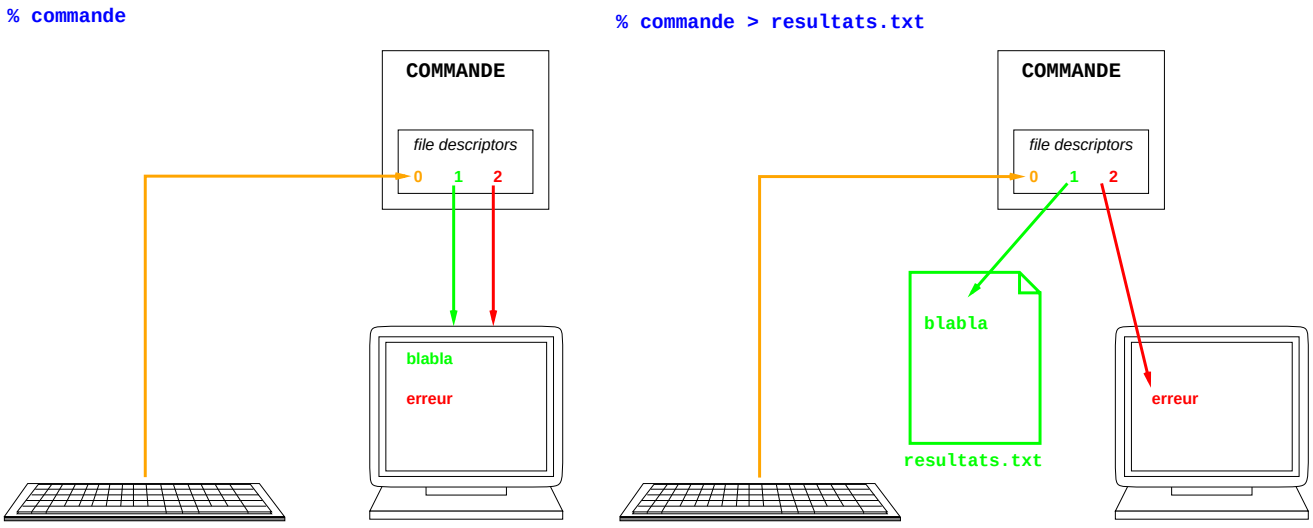

Exemple : extrait d'un script qui partitionne un disque dur en 2 partitions de 128 MB et 128 MB respectivement.

```
% cat partitions.txt
n
p
1

+127
n
p
2

+127
w
% fdisk /dev/sdb < partitions.txt
```

La commande « `fdisk` » est en principe interactive. Ici le script proposera les réponses lui même sans passer par un fichier de réponse annexe.

◇ Redirection du file descriptor 1

Comment s'organisent les file descriptors lors de « `commande > resultats.txt` » ?

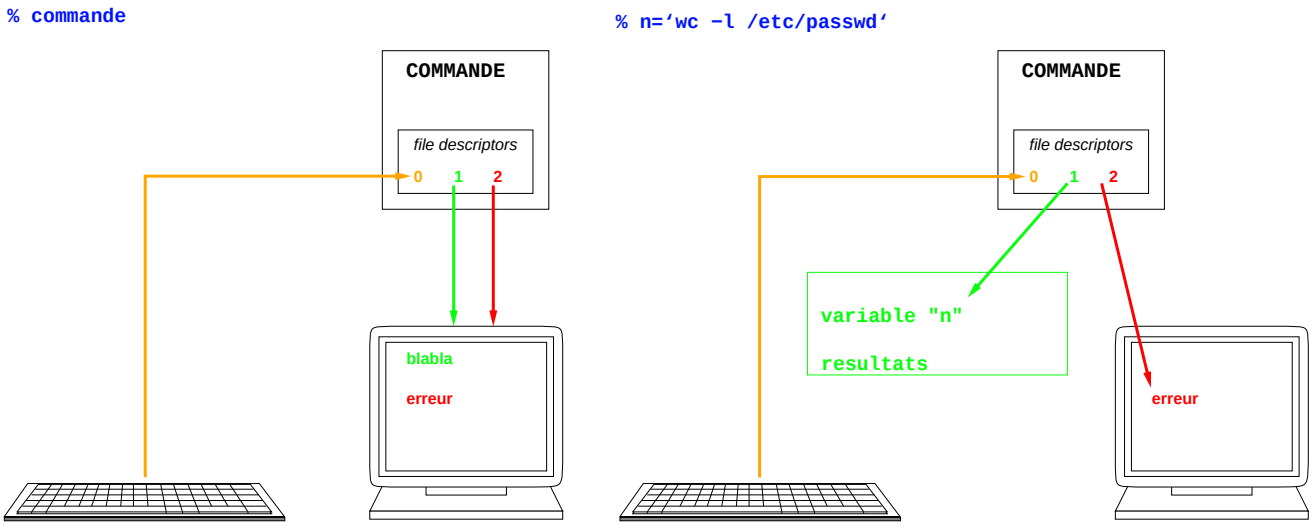


Exemples :

1. `% ls > /tmp/exemple.txt`
2. `% ls /etc >> /tmp/exemple.txt`

◇ Redirection du file descriptor 1 (suite)

Comment s'organisent les file descriptors lors de « `commande > resultats.txt` » ?

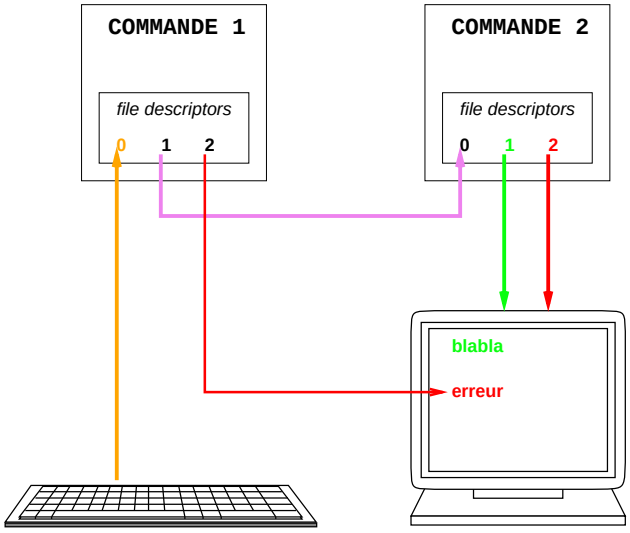


Exemples :

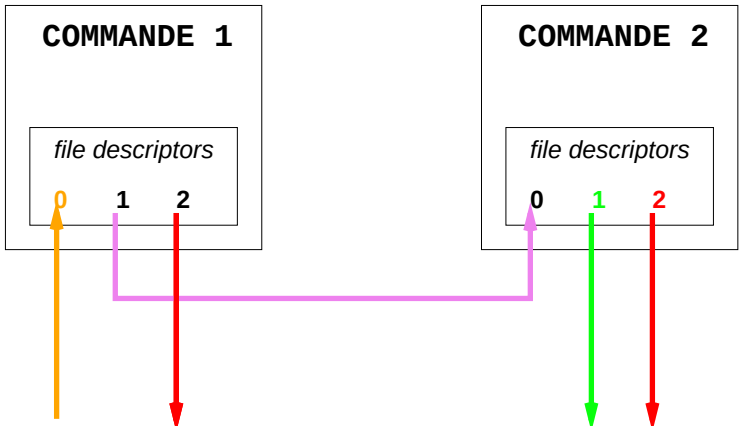
1
```
% n=`wc -l /etc/passwd`
% echo $n
170 /etc/passwd
```

◇ Redirection avec un pipe

Comment s'organisent les file descriptors lors de « `commande1 | commande2` » ?



Pour que « `commande1 | commande2` » fonctionne, il faut que la commande2 lise sur stdin (filedescriptor 0) !



Ce n'est pas le cas de toutes les commandes !
Certaines commandes UNIX ne pourront jamais être utilisées dans un pipe !

Exemples qui fonctionnent :

1. `% ls -l | more`

2. `% cat /etc/group | more`

   qui équivaut à

   `more /etc/group`

Exemples qui ne fonctionnent pas :

1. `% echo exemple.txt | ls`

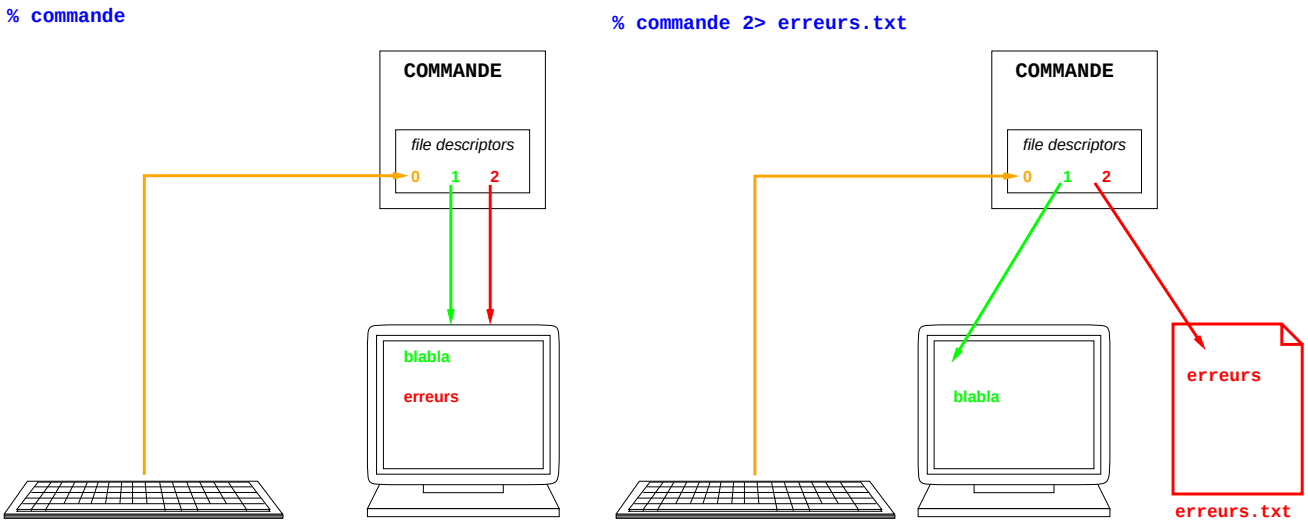
   ```
   ananas cerise
   ```

2. `% echo exemple.txt | rm`

   ```
   usage: rm [-fiRr] file ...
   ```


◇ Redirection du file descriptor 2

Comment s'organisent les file descriptors lors de « `commande 2> erreurs.txt` » ?



Exemple :

```
% ls fichier-inexistant.txt
ls: fichier-inexistant.txt: No such file or directory

% ls fichier-inexistant.txt > erreurs.txt
ls: fichier-inexistant.txt: No such file or directory

% ls fichier-inexistant.txt 2> erreurs.txt
% cat erreurs.txt
ls: fichier-inexistant.txt: No such file or directory
```


Exemple compliqué qui montre que l'ordre des redirections est important.

Où est le message d'erreur ? :

```
% ls exemple.txt inexistant.txt >&2 2> erreurs.txt
exemple.txt
```

Où est le message normal affiché ? :

```
% ls exemple.txt inexistant.txt 2> erreurs.txt >&2
% ...
```

Explications :

- Le shell évalue la ligne de commande de gauche à droite.
- Dans la commande « `ls exemple.txt inexistant.txt >&2 2> erreurs.txt` », stdout est redirigé sur l'écran puisque stderr équivaut à l'écran à cet instant, puis stderr est redirigé sur `erreurs.txt`. D'où le résultat.

◇ Moralité :

- Forme générale de lancement d'une commande en background :

```
% commande < reponses.txt > resultats.txt 2> erreurs.txt &
```

- Suivi de l'avancée du calcul : le processus tourne-t-il encore ?
  - si l'on n'a pas terminé le shell dans lequel on a lancé le calcul
    ⇒ utiliser « `jobs` »
  - on a terminé le shell dans lequel on a lancé le calcul
    ⇒ utiliser « `ps` » :

    ```
    % ps -edf | grep commande
    ```


◇ Moralité (suite) :

- Suivi de l'avancée du calcul via l'inspection des fichiers « `resultats.txt` » et « `erreurs.txt` » :
  - consultation des fichiers par « `cat` » ou « `more` » ou autre commande de ce type :

    ```
    % more resultats.txt
    % more erreurs.txt
    ```
  - consultation des fichiers par « `tail -f` » :

    ```
    % tail -f resultats.txt
    % tail -f erreurs.txt
    ```

    Terminer un « `tail -f` » par Ctrl-C (voir page 522).

◇ Protection contre l'écrasement de fichiers lors de redirection

Sous le shell BASH de LINUX on peut se protéger contre l'écrasement intempestif de fichier lors d'une redirection.
Pour cela, il faut créer une variable d'environnement BASH spéciale (se reporter page 596 pour les explications sur les variables d'environnement) :

```
% ls -l
-rw-r--r--   1 besancon ars      0 Aug  7 20:39 ananas.txt
-rw-r--r--   1 besancon ars      0 Aug  7 20:40 banane.txt
-rw-r--r--   1 besancon ars      0 Aug  7 20:40 cerise.txt
% set -o noclobber
% ls > cerise.txt
bash: cerise.txt: cannot overwrite existing file
```

Voir page 631 pour rendre ce réglage permanent.

On peut vouloir se débarrasser d'une partie de l'affichage.

Solution inefficace :

```
% application > /tmp/resultats
...
% rm /tmp/resultats
```

La solution est de rediriger vers « /dev/null » :

```
% application > /dev/null
...
```

« /dev/null » est indispensable dans la vie de l'administrateur système.

◇ Forme générale de lancement d'une commande en background :

• On garde tous les messages émis par le calcul :

```
% calcul < reponses.txt > resultats.txt 2> erreurs.txt &
```

• On garde les messages normaux émis par le calcul mais pas les messages d'erreur :

```
% calcul < reponses.txt > resultats.txt 2> /dev/null &
```

• On garde les messages d'erreur émis par le calcul mais pas les messages normaux :

```
% calcul < reponses.txt > /dev/null 2> erreurs.txt &
```

• On ne garde aucun message émis par le calcul :

```
% calcul < reponses.txt > /dev/null 2> /dev/null &
```

« `jobs` », « `ps` », « `more` », « `cat` », « `tail -f` » reste vrai)

Ces metacharacters servent à construire des noms de fichiers.

Caractère	Description
`*`	0 ou plus caractères quelconques
`?`	**un et un seul** caractère quelconque exactement
`[ ]`	1 caractère énumré dans l'ensemble entre crochets
`[^ ]`	1 caractère non énuméré dans l'ensemble entre crochets

Exemples :

1. `% ls *`
```
fichier1.txt  fichier2.txt  fichier3.txt  fichier4.txt
```
2. `% ls /etc/*.??`
```
/etc/locate.rc    /etc/pwd.db        /etc/spwd.db
/etc/mail.rc      /etc/sendmail.cf
```
3. `% ls /var/log/[lp]*`
```
/var/log/lastlog  /var/log/lpd-errs  /var/log/ppp.log
```
4. `% ls /var/log/[^mw]*`
```
/var/log/dmesg                /var/log/ppp.log
/var/log/dmesg.today          /var/log/sendmail.st
/var/log/dmesg.yesterday      /var/log/setuid.today
/var/log/lastlog              /var/log/setuid.yesterday
/var/log/lpd-errs             /var/log/slip.log
```


Attention : le shell remplace les metacharacters s'il le peut et uniquement s'il le peut. Sinon il renonce au sens spécial du metacharacter et laisse le metacharacter tel que au milieu des autres caract'eres.

◇ Exemples (1)

```
% ls
banane.txt   cerise.txt

% echo b*
banane.txt

% echo a*
a*   <- le shell laisse « * »
```

◇ Exemples (2) : le répertoire est vide

```
% echo *
*  <- le shell laisse « * »

% ls *
*: No such file or directory <- pas d'objet « * »

% rm *
*: No such file or directory <- pas d'objet « * »

% rmdir *
rmdir: directory "*": Directory does not exist <- pas
d'objet « * »
```


◇ Exemples (3) : le répertoire est vide

```
% echo [a-z]*
[a-z]*   <- le shell laisse « [a-z]* »

% ls [a-z]*
[a-z]*: No such file or directory  <- pas d'objet « [a-z]* »

% rm [a-z]*
[a-z]*: No such file or directory  <- pas d'objet « [a-z]* »

% rmdir [a-z]*
rmdir: directory "[a-z]*": Directory does not exist  <- pas
d'objet « [a-z]* »
```

◇ Noms de variables

Les noms des variables suivent les mêmes règles que celles du langage C.

```
temp
1abc
abc_d_e
temp.doc
abc#1
avc-d
TeMpVaR
a$3
```


◇ Liste des variables définies dans la session shell

```
set
```

Attention : les variables ne seront pas classées.

Conseil : faire « `set | sort` »

◇ Assignation de variable

```
variable=valeur
```

Rappel : l'espace est un caractère spécial donc pas d'espace de part et d'autre du signe « = ».

Sinon grosses erreurs :

```
% a =3
a: not found
% a= 3
3: not found
% a = 3
a: not found
```


◇ Consultation de variable

Au choix :

- écriture « `$variable` »
- écriture « `${variable}` »

Préférer la seconde écriture. Pour la raison suivante :

```
% a=ananas
% echo $a33

% echo ${a}33
ananas33
```

Rappel :

```
% echo "j'ai 5 \$"
j'ai 5 $
```

Mais aussi :

```
% echo "j'ai 5 $"
j'ai 5 $
```


◇ Suppression de variable

```
unset variable
```

◇ Types des variables

LES VARIABLES SONT DE TYPE CARACTÉRES.

Il n'y a pas d'autre type.
Il n'y a pas de type numérique.

Il est donc impossible de faire :

```
compteur=1
compteur=$compteur + 1
```


◇ Manipulations numériques sur les variables : `expr`

(en anglais *expression*)

Pour faire des opérations numériques sur les variables, on fait ainsi : « `expr` » :

```
compteur=1
compteur=`expr $compteur + 1`
```

Se reporter à la page de manuel de « `expr` » pour les autres opérations mathématiques réalisables.

ATTENTION à la multiplication ! : conflit avec le métacaractère « ⋆ » du shell :
Il faut écrire :

```
% expr 2 \* 3
6
```

```
% expr 2 "*" 3
6
```

```
% expr 2 '*' 3
6
```

mais la forme ci-dessous est FAUSSE (voir page 557) :

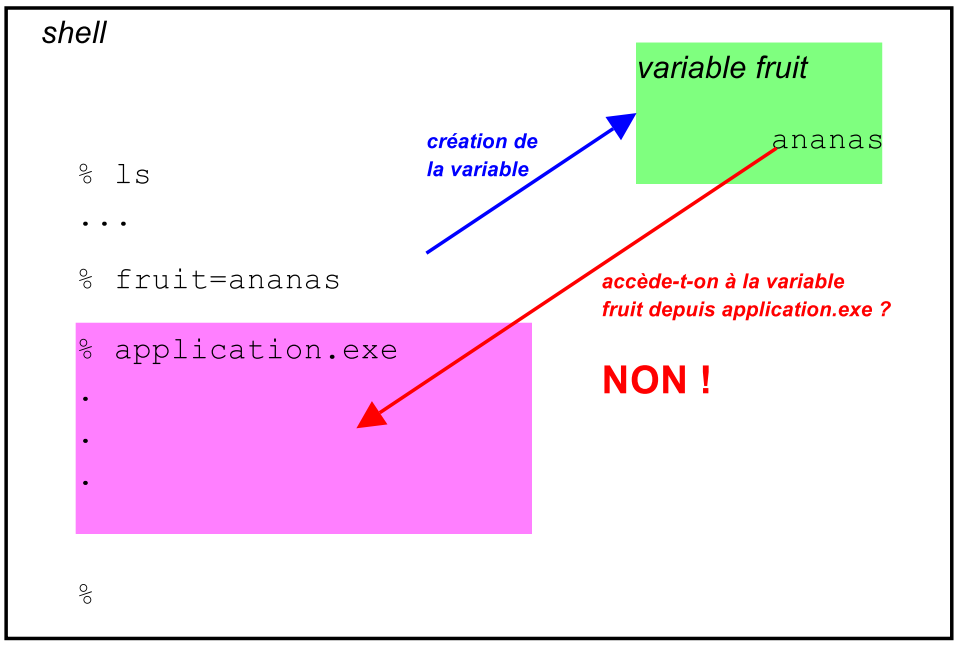
```
% expr 2 * 3
expr: syntax error
```


◇ Visibilités des variables

A un shell sont associées des variables uniquement définies dans ce shell et uniquement accessibles dans ce shell.

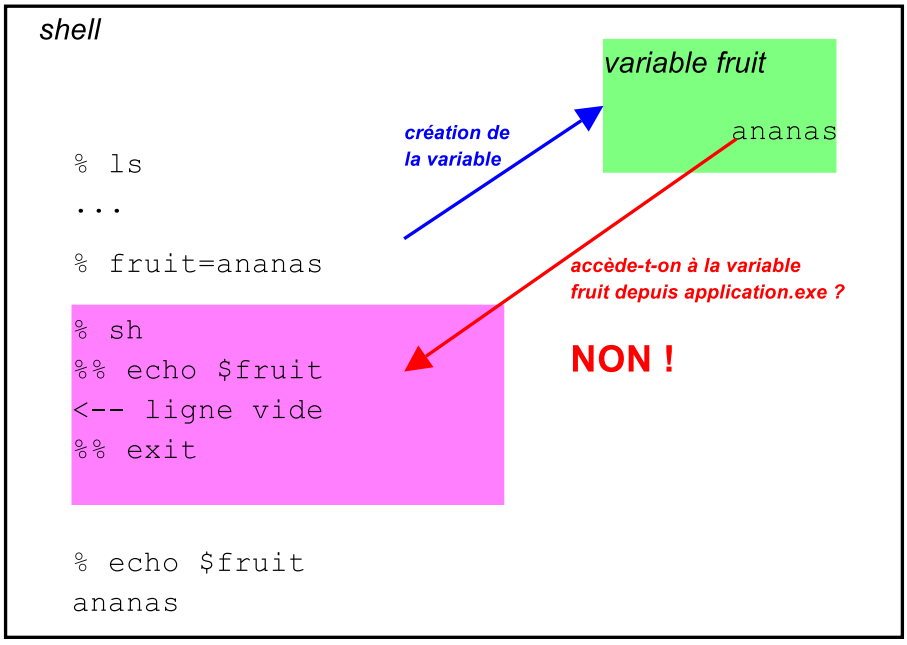
Les variables disparaissent lorsque le shell meurt.

Comment les applications accèdent-elles aux variables du shell ?



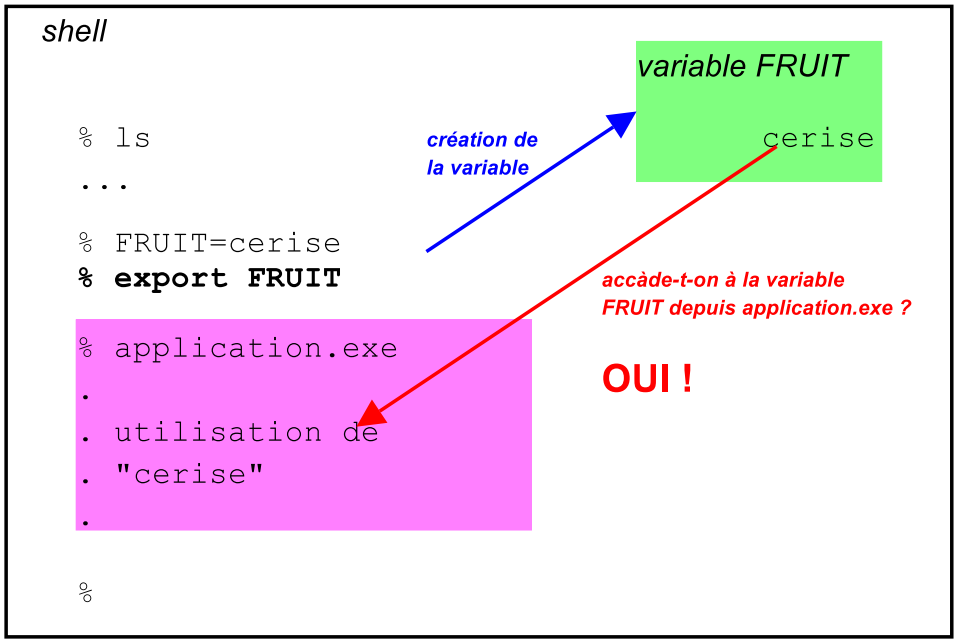
⇒ Les variables ne sont pas héritées par défaut par les commandes lancées par le shell.

Vérification en lançant une application spéciale, un shell (intérêt du shell : le shell permet de faire des vérifications en mode interactif).



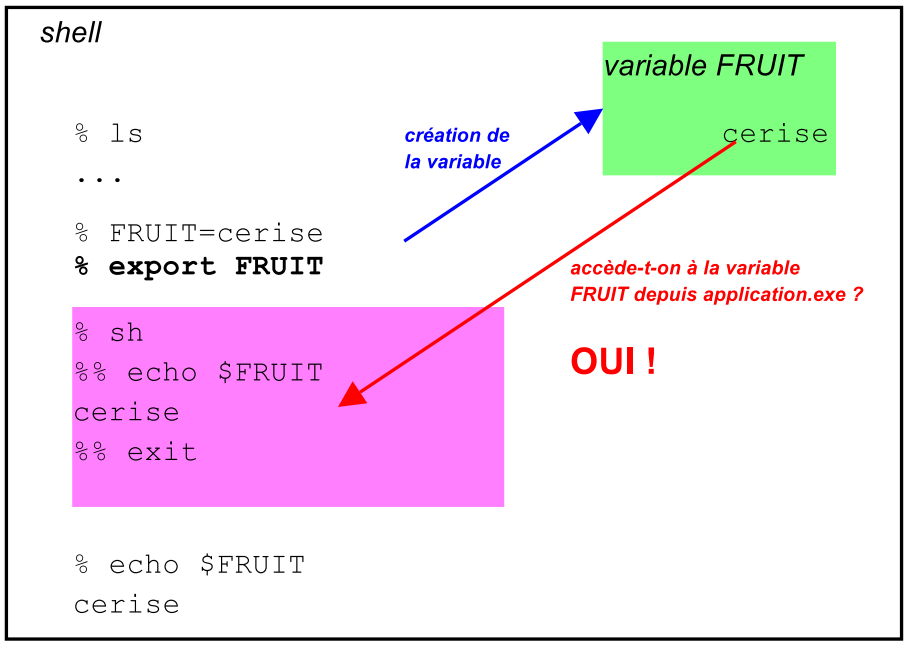
⇒ Les variables ne sont pas héritées par défaut par les commandes lancées par le shell.

La solution consiste à utiliser « **export** » :

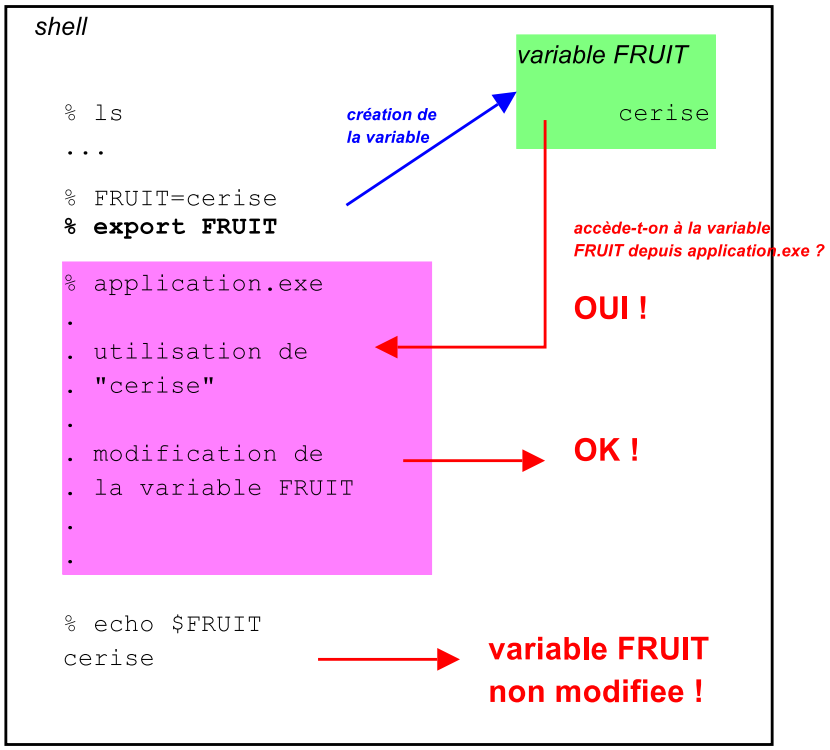


NB : traditionnellement, les variables d'environnement sont écrites en lettres majuscules

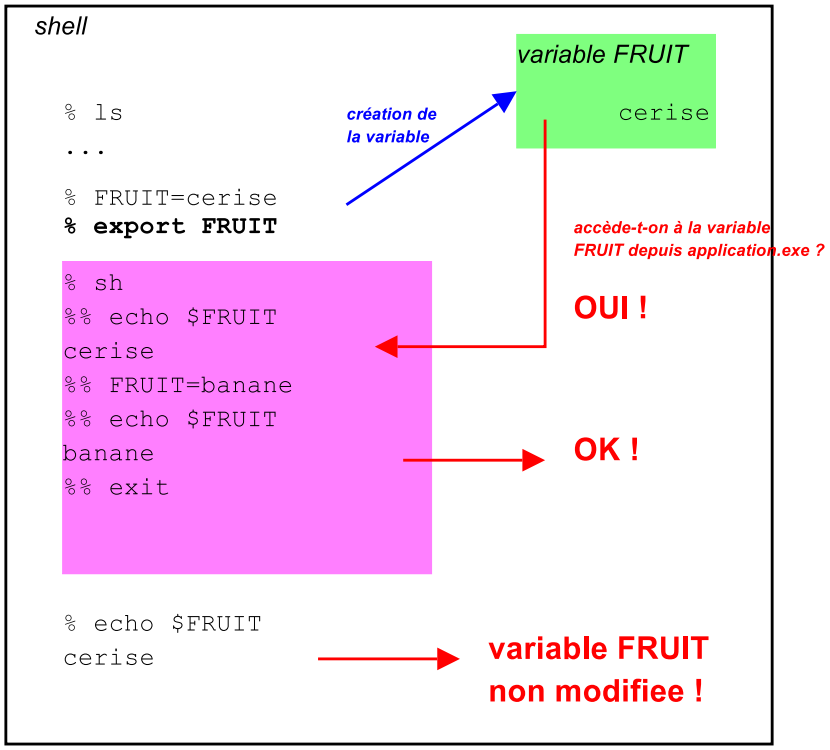
Vérification avec un shell :



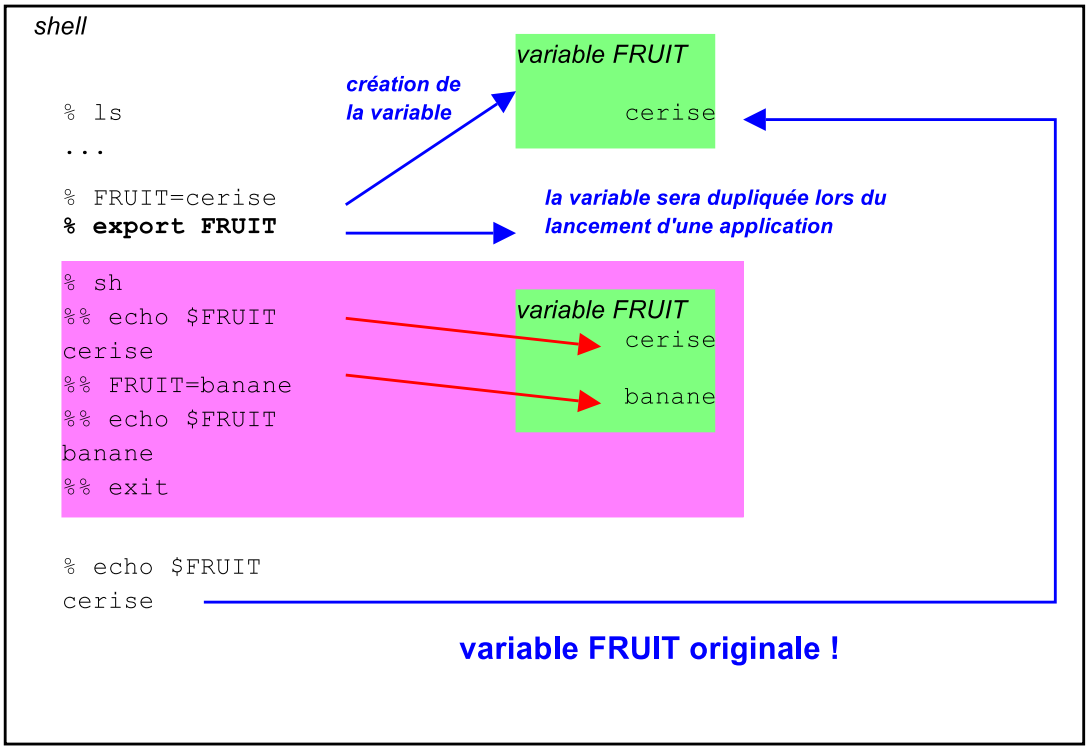
ATTENTION ! ! ! : L'environnement est hérité en lecture uniquement.



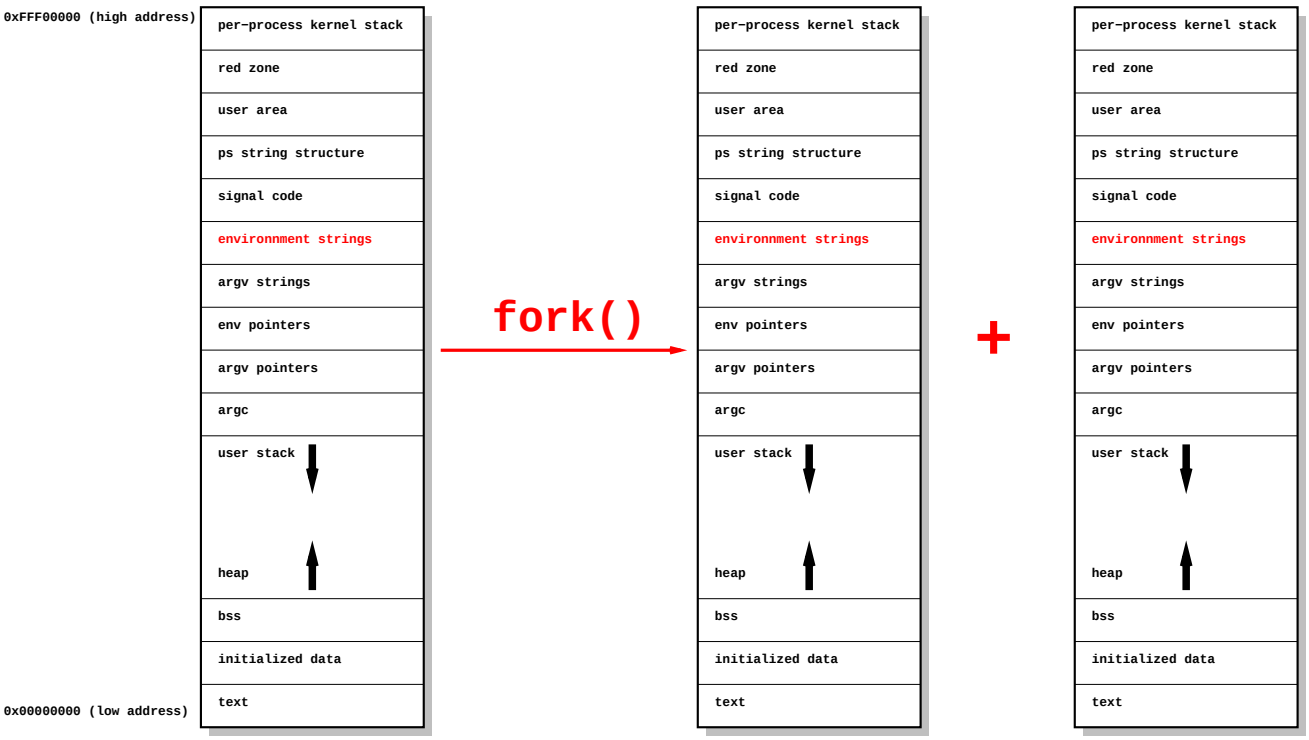
Vérification avec un shell :



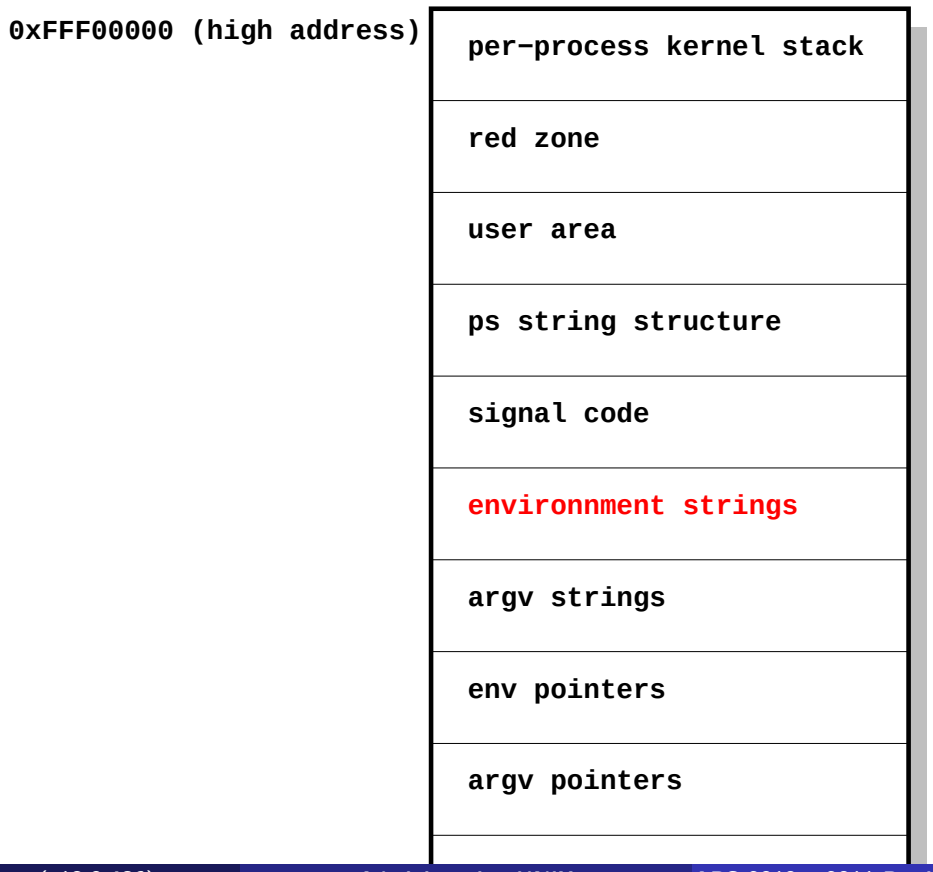
Le fonctionnement réel est le suivant :



Retour sur le mécanisme du fork() utilisé lorsque le shell lance une commande (voir page 513) :



Dans la structure d'un processus UNIX, il y a une zone pour stocker des variables d'environnementi (en rouge sur le dessin) :



Pour afficher la liste des variables d'environnement :

- « `env` »
- « `printenv` » (peut-être non standard ?)

Attention : les variables d'environnement ne seront pas classées (comme page 587).

Conseil : faire « `env | sort` »

Voici la liste des variables d'environnement standard :

Variable	Description
`HOME`	répertoire d'accueil personnel où l'utilisateur stocke ses fichiers (en anglais *home directory*)
`USER`	nom d'utilisateur (en anglais *username*)
`SHELL`	chemin du shell en cours d'utilisation
`PATH`	liste des dossiers système dans lesquels chercher les commandes
`TERM`	type du terminal texte utilisé

Les autres variables d'environnement ne sont pas standard (a priori).

En Bourne Shell, la variable « `PATH` » stocke le chemin de commandes.

Chemin de commandes ≡ liste de répertoires séparés par des « : »

Parcours de tous les répertoires jusqu'à trouver la commande en question

- Hypothèse : le « `PATH` » vaut « `/bin:/usr/bin:/usr/local/bin` »
- Hypothèse 2 : l'utilisateur tape « `ls` »

1. Le shell cherche si l'exécutable « `/bin/ls` » existe. Si non étape suivante.
2. Le shell cherche si l'exécutable « `/usr/bin/ls` » existe. Si non étape suivante.
3. Le shell cherche si l'exécutable « `/usr/local/bin/ls` » existe. Si non étape suivante.
4. Si plus de répertoires de « `PATH` » à analyser, afficher l'erreur « `command not found` »

En mode interactif, pour ajouter le répertoire « /chemin/vers/application/bin » à son « PATH » :

- en début de « PATH », faire :

```
% PATH=/chemin/vers/application/bin:$PATH
% export PATH
```

- en fin de « PATH », faire :

```
% PATH=$PATH:/chemin/vers/application/bin
% export PATH
```

Pour retirer un répertoire de son « PATH » :

- pas de méthode à part retaper tout :

```
% PATH=/repertoire1:/repertoire2:/repertoire3:...:/repertoirel
% export PATH
```


On ne met jamais « . » dans son PATH ! DANGER ! ! !

« . » ≡ le répertoire courant

Dangers :

- Que contient le répertoire courant ?
- L'environnement peut être hostile.
- On peut faire des fautes de frappe, des coquilles.

Si besoin d'une commande dans le répertoire courant, l'appeler explicitement par « ./ » :

```
% ./commande-dans-répertoire-courant options paramêtres...
```

Lien avec le cours de programmation C :

```
% gcc prog.c
% ./a.out
blabla
```


Hypothèse : on est sous Bourne Shell ou sous BASH.
Hypothèse 2 : on se rappelle le passage « Variable d'environnement PATH » vu dans le chapitre « Pratique du Bourne Shell ».

Objectif : on veut modifier son « PATH » de façon permanente.

Solution :
On utilise le fichier « $HOME/.profile » avec un lien symbolique « $HOME/.bashrc » dessus.

On règle ainsi (sh ou bash) :

- cas du shell interactif de login
- cas du shell interactif non de login
- cas du shell non interactif

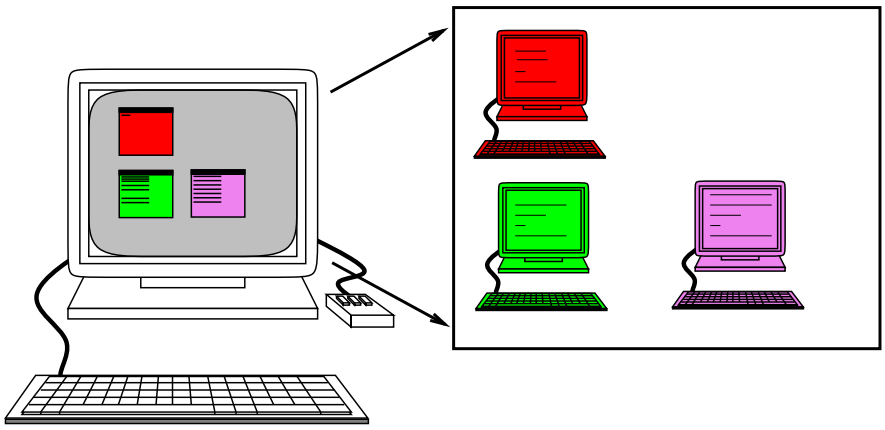
On ajoute dans le fichier « `$HOME/.profile` »

```
PATH=/repertoire1:/repertoire2:/repertoire3:...:/repertoireN
export PATH
```

ou

```
PATH=$PATH:/chemin/vers/application/bin
export PATH
```


Rappel :



Tous les programmes texte utilisent des « **escape sequences** » pour déplacer le curseur à l'écran.

⇒ nécessité de savoir comment faire

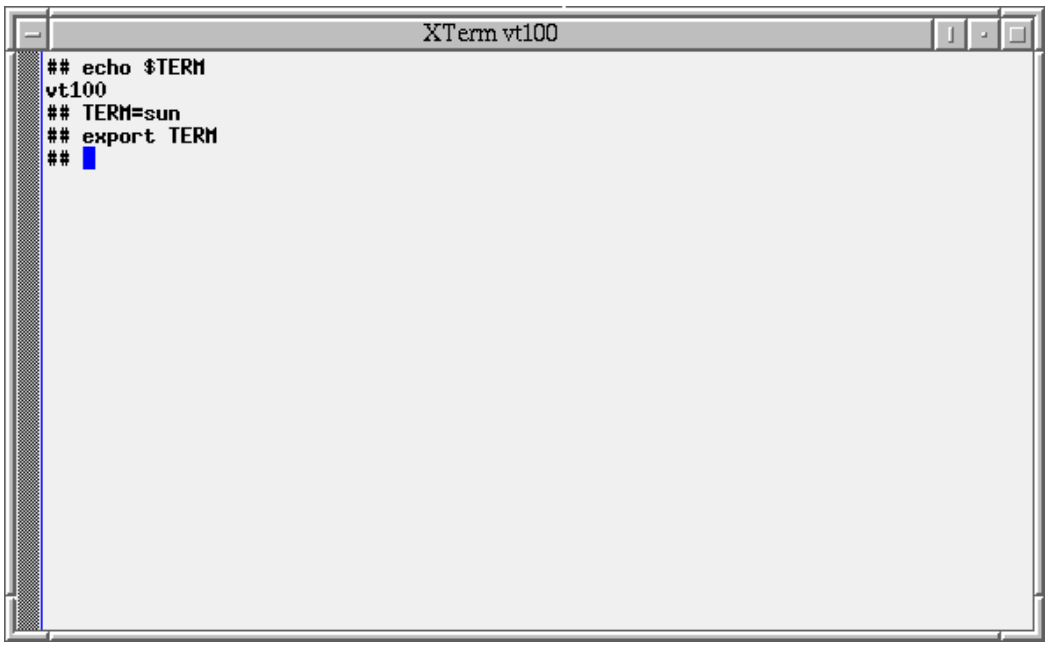
⇒ base de données des escape sequences ; l'entrée dans la base de données est donnée par la variable « `TERM` »

La variable d'environnement « TERM » définit le modèle de terminal texte utilisé.
Dans le passé, il y avait pléthore de console de terminaux texte.
⇒ base de données des modèles de terminaux :

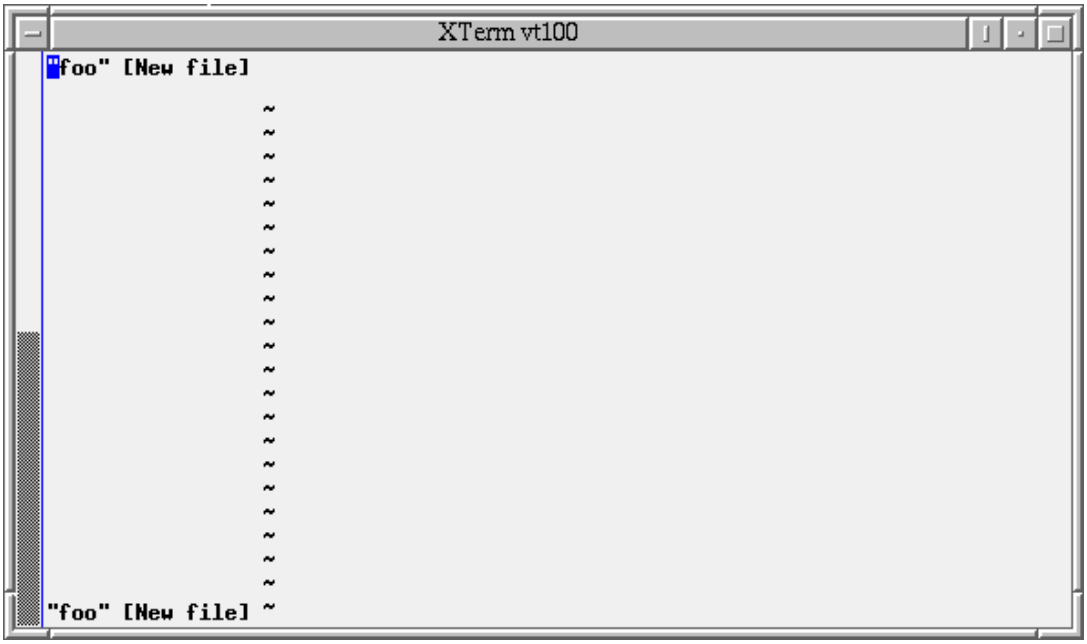
- Implémentation système BSD : base de données au format TERMCAP stockée dans le fichier « /etc/termcap »
- Implémentation système System-V : base de données au format TERMINFO stockée dans l'arborescence **/usr/share/lib/terminfo/**
  Par exemple pour le minitel :
  « /usr/share/lib/terminfo/m/minitel »
  Base à jour disponible à http://www.tuxedo.org/~esr/terminfo
  Utilitaires « tic » (terminfo compiler), « captoinfo » pour convertir les fichiers de description de terminaux de termcap à terminfo

Ne jamais changer la valeur de « TERM » mise automatiquement par le système sinon problèmes dans éditeurs, etc.

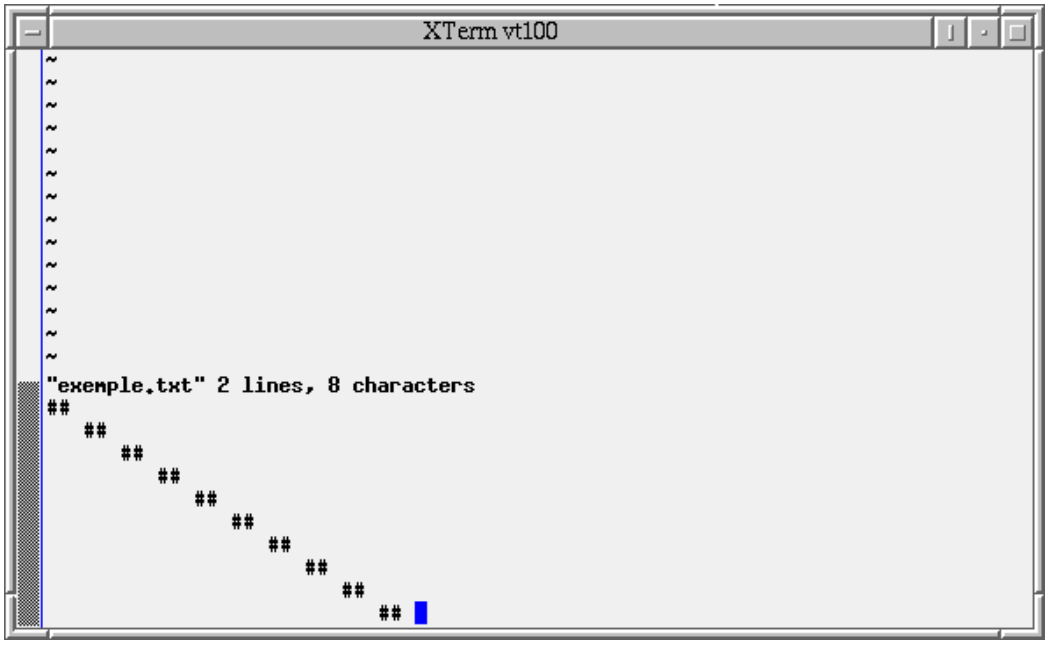
Jouons avec le feu en changeant la valeur mise par le système pour votre fenêtre :



Sous l'éditeur « `vi` », les ennuis sont là :



Sous le shell, les ennuis sont là :



Solution : commande « `reset` » ; retour aux réglages par défaut du terminal si des escape séquences ont fichu la pagaille

La commande « `stty` » donne les caractéristiques bas niveau du terminal.

Dans les résultats renvoyés on trouve les configurations de certaines séquences de touches utilisées interactivement.
Par exemple l'effacement du caractère précédent de la ligne via la touche Del ou Backspace.

Il y a 2 versions de la commande selon la famille d'UNIX. L'affichage des données est différent mais on manipule la même chose au final.

Attention à ne pas confondre ce que permet la gestion du terminal texte et ce que permet le shell (par exemple « `bash` » offre des possibilités de déplacement du curseur au sein de la ligne de commande).

Principales noms de séquences utiles :

- séquence « erase » : effacement du caractère précédent
- séquence « werase » : effacement du mot précédent
- séquence « kill » : effacement de la ligne compléte
- séquence « intr » : envoi du signal SIGINT
- séquence « quit » : envoi du signal SIGABRT
- séquence « susp » : envoi du signal SIGTSTP
- séquence « eof » : End Of File
- séquence « start » : relance le flux de l'affichage texte
- séquence « stop » : arrête le flux de l'affichage texte
- séquence « lnext » : permet la saisie de la séquence suivante sans l'interpréter

◇ Exemple de « stty » sur Solaris

```
% /usr/bin/stty -a
speed 9600 baud;
rows = 55; columns = 80; ypixels = 719; xpixels = 579;
csdata ?
eucw 1:0:0:0, scrw 1:0:0:0
intr = ^c; quit = ^\; erase = ^?; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -crtscts -crtsxoff
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff imaxbel
isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop echoctl -echoprt echoke -defecho -flusho -pendin iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

◇ Modification de séquence via « stty »

Reprenons les séquences précédentes :

```
intr = ^c; quit = ^\; erase = ^?; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
```

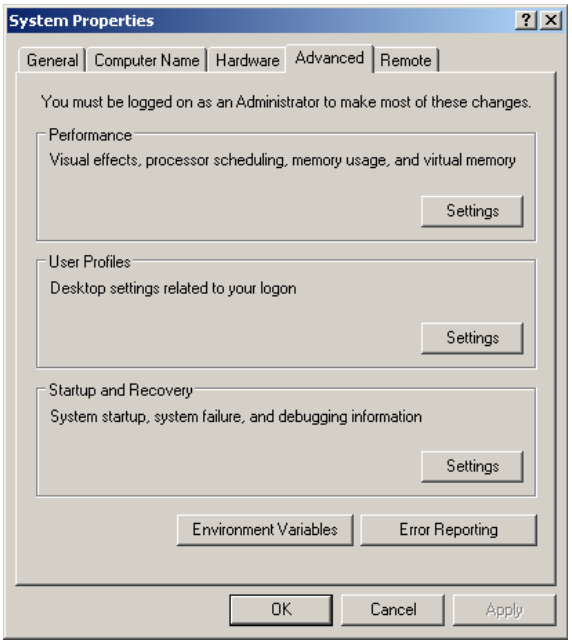
Le shell standard sous Linux est « bash » qui permet de revenir en début de ligne tapée par « ^e ».
Pour changer « intr » de « ^c » en « ^e », on tapera donc en pratique :

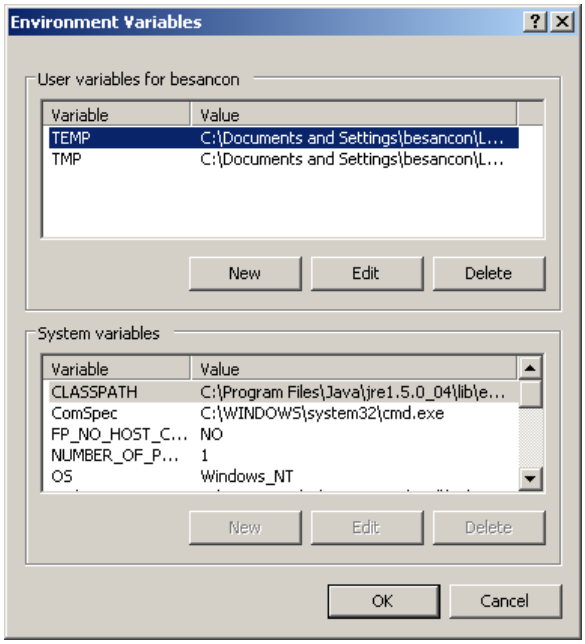
```
% stty intr ^v^e
```

Le « ^v » neutralise l'effet de « ^e » (sous bash retour en début de ligne) lorsqu'on le tape.
Le « ^v » est indiqué par la séquence « lnext » de « stty ».

Chapitre 13 • Pratique du Bourne shell
§13.26 • (Windows : : Variables d'environnement)

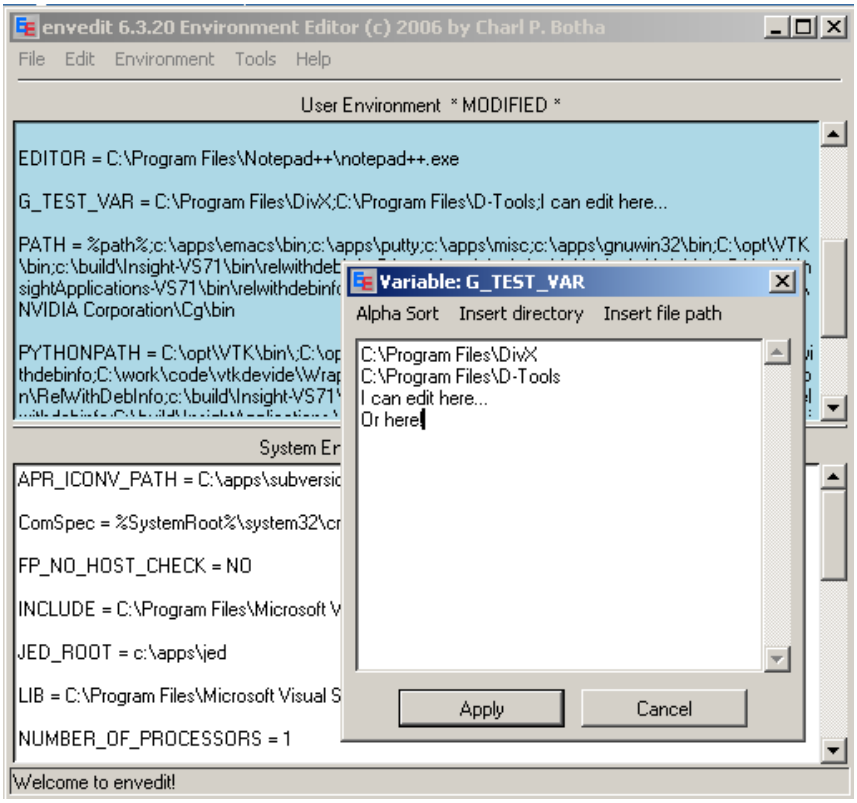
Interface de base fournie par Windows :
Start > Control Panel > System > Advanced > Environment Variables





Interface peu pratique

Programme plus ergonomique : **envedit** sur `http://cpbotha.net/Software/envedit`



Chapitre 13 • Pratique du Bourne shell

§13.27 • Ordre d'évaluation de la ligne de commande

1. Redirection des entrées/sorties
2. Substitution des variables
3. Substitution des noms de fichiers

◇ Exemple 1 :

```
% pipe=\|
% echo $pipe
|
```

Explications :

1. pas de caractères de redirection des entrées/sorties ;
   la commande est « `echo $pipe` »
2. remplacement de la variable par son contenu ;
   la commande est « `echo |` »
3. pas de caractères de substitution de noms de fichiers ;
   la commande est « `echo |` »

◇ Exemple 2 :

```
% star=\*
% echo $star
ananas banane cerise
```

Explications :

1. pas de caractères de redirection des entrées/sorties ;
   la commande est « `echo $star` »
2. remplacement de la variable par son contenu ;
   la commande est « `echo *` »
3. remplacement du caractère « `*` » par la liste des fichiers ;
   la commande est « `echo ananas banane cerise` »

Pour se déconnecter : taper « exit »

Parfois, sous le shell BASH de LINUX on peut se déconnecter via « Ctrl-D » :

```
% ^D
<-- on est déconnecté
```

« Ctrl-D » est appelé **EOF** (*End Of File*)

Possibilité de désactiver la séquence « Ctrl-D » sous BASH via une variable spéciale interne de BASH :

```
% set -o ignoreeof
% ^DUse "exit" to leave the shell.
```


La commande « login » lance un **shell de login**.
Le nom du shell est alors précédé d'un signe moins :

```
% ps -edf | grep bash
besancon  1773     1  0   Aug 24 console  0:00 -bash
besancon  1959  1949  0   Aug 24 pts/3    0:09 bash
besancon  8300  8299  0   Aug 26 pts/5    0:00 bash
```

Le shell sait alors qu'il est un shell de login (via « argv[0] »)
⇒ le shell exécute les fichiers de configuration de login
Un shell non de login n'exécutera pas les fichiers de configuration de login et sera initialisé plus simplement.
On peut forcer le mode shell de login :

- avec bash : « bash --login »
- avec tcsh : « tcsh -l »
- avec xterm : « xterm -ls » ou
  « xterm -xrm '*loginShell: true' »

Un shell est interactif si les commandes sont saisies sur un terminal.
Un shell est non interactif si les commandes sont lues dans un fichier.
Si le shell est non interactif, on n'exécutera pas les fichiers de commandes interactives et le shell sera initialisé plus simplement.
La commande « `tty -s` » permet de tester si l'on est en mode interactif ou pas :

- code de retour 0 : mode interactif
- code de retour 1 : mode non interactif

On peut forcer le mode shell interactif :

- avec bash : « `bash -i` »
- avec sh : « `sh -i` »
- avec tcsh : « `tcsh -i` »
- avec csh : « `csh -i` »

◇ Mode shell de login interactif

(sauf option contraire « `--noprofile` ») :

1. on exécute « `/etc/profile` »
2. on exécute le premier et seulement le premier fichier existant parmi « `$HOME/.bash_profile` », « `$HOME/.bash_login` », « `$HOME/.profile` »

◇ Mode shell non interactif mais avec l'option « `-login` »

(sauf option contraire « `--noprofile` ») :

1. on exécute « `/etc/profile` »
2. on exécute le premier et seulement le premier fichier existant parmi « `$HOME/.bash_profile` », « `$HOME/.bash_login` », « `$HOME/.profile` »

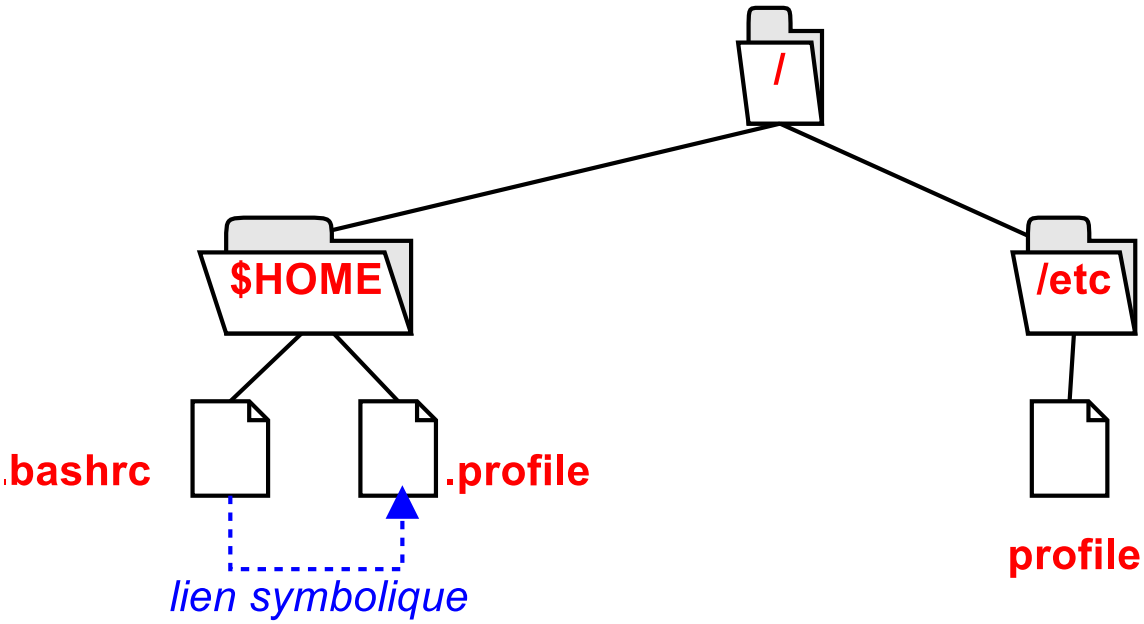
◇ Mode shell non de login interactif :

1. on exécute « `$HOME/.bashrc` » (sauf option contraire « `--norc` »)

◇ Mode shell non de login non interactif :

1. on exécute le fichier mentionné par « $BASH_ENV »

Conseil pour traiter simplement les trois premiers cas :



◇ Quand un shell de login se termine :

1. on exécute « `$HOME/.bash_logout` »

◇ Travailler sous bash avec des caractères accentués (éèà...) :

ajouter au fichier « `$HOME/.inputrc` » les lignes suivantes :

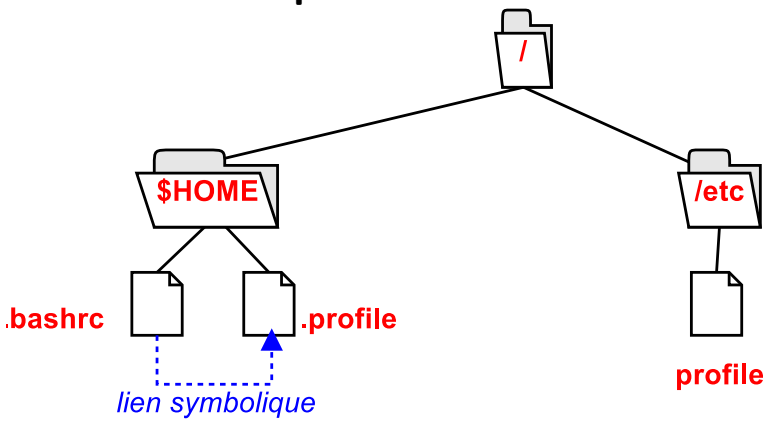
```
set meta-flag on
set convert-meta off
set output-meta on
```

En mode shell de login :

1. on exécute « `/etc/profile` »
2. on exécute « `$HOME/.profile` »

En mode autre : on n'exécute aucun fichier !

Compatibilité avec le conseil pour bash :



En mode shell de login :

1. on exécute « `/etc/csh.cshrc` »
2. on exécute « `/etc/csh.login` »
3. on exécute « `$HOME/.tcshrc` » (ou « `$HOME/.cshrc` » à la place si non existant)
4. on exécute « `$HOME/.history` »
5. on exécute « `$HOME/.login` »
6. on exécute « `$HOME/.cshdirs` »

En mode shell non de login ou en mode non interactif :

1. on exécute « `/etc/csh.cshrc` »
2. on exécute « `$HOME/.tcshrc` » (ou « `$HOME/.cshrc` » à la place si non existant)

Quand un shell de login se termine :

1. on exécute « /etc/csh.logout »
2. on exécute « $HOME/.logout »

Chapitre 13 • Pratique du Bourne shell
§13.34 • Fichiers d'initialisation pour csh

En mode shell de login :

1. on exécute « /etc/.login »
2. on exécute « $HOME/.cshrc »
3. on exécute « $HOME/.login »
4. on exécute « $HOME/.cshdirs »

Quand un shell de login se termine :

1. on exécute « $HOME/.logout »

Sur LINUX, le shell s'appelle BASH = version améliorée du Bourne Shell

Une des fonctionnalités interactives les plus intéressantes : l'expansion des noms de fichier via la touche TAB : on parle de **complétion** :

```
% ls
abricot.txt     asperge.txt     choux.txt       poire.txt
ananas.txt      banane.txt      fraise.txt      poireau.txt
artichaut.txt   cerise.txt      patate.txt      pomme.txt
% ls pTAB
patate.txt   poire.txt    poireau.txt  pomme.txt
% ls poTAB
poire.txt    poireau.txt  pomme.txt
% ls pomTAB
% ls pomme.txt
```


La complétion est aussi disponible sous WINDOWS dans un « `cmd.exe` ».

Pour Windows NT et Windows 2000 : réglage dans la base de registres : affecter la valeur 9 à la clef
`HKEY_CURRENT_USER/Software/Microsoft/Command Processor/CompletionChar`

Pour Windows XP : fonctionnalité installée de base

Chapitre 14

Programmation en Bourne shell

Le shell propose un langage de programmation interprété.

Son utilité :

- automatisation d'actions
- utilisation de structures plus avancées :
  - boucles
  - tests
  - ...
- scripts d'installation de logiciels à adapter

On appelle « shell script » un programme écrit dans la syntaxe d'un shell et s'appuyant sur les commandes UNIX.

Caractéristiques d'un shell script :

- C'est un programme écrit en langage shell.
- Il est écrit pour un shell particulier, à la syntaxe bien particulière. Un shell script ne peut pas être exécuté par un autre shell en général.
- Il est exécutable.
  ⇒ faire « `chmod a+x exemple.sh` »

Structure d'un shell script :

- **Désignation du shell utilisé**
  La première ligne du shell script commence par « `#!` » suivi du path du shell utilisé et de ses arguments éventuels.
- **Commentaires**
  Un commentaire est introduit par le caractère « `#` » et se poursuit jusqu'à la fin de la ligne.
  Un commentaire peut être placé n'importe où.
  La première ligne du script est un commentaire très particulier.
- **Code**
  Traditionnelles lignes de code respectant la syntaxe du shell utilisé.

Exemple :

```
#! /bin/sh
#
# Script d'exemple. Il affiche simplement la date.
#
date
```

ATTENTION ! ! !

Ceci est faux (pas uniquement à cause de l'orthographe) :

```
#############################################################################
#                                                                           #
# ARS 1999/2000
# AUTEUR : BERGOUGNOUX YVES                                                 #
# DATE DE CREATION : 07/01/2000                                             #
# DATE DE MODIFICATION : 07/01/2000                                         #
# THEME : enrgitre l'expiration d'un compte de stagiaire ayant les memes    #
# droit que sont parain pour géré la fin du compte dans le fichier cpt_sta#
# NOM DE LA COMMANDE : hda5/home/yves/projet/set-deadline (disque UNIX)     #
#                                                                           #
#############################################################################

#!/bin/sh

...
```

Pourquoi ?

A cause de la position de la ligne « `#!/bin/sh` »

En l'absence d'indication de l'interpréteur de commandes en première ligne du script, le script est exécuté par le shell courant de la session de l'utilisateur.

Attention aux systèmes comme Linux où le shell par défaut est compatible avec le Bourne shell, masquant ainsi l'erreur ! ! !

Preuve :

Soit le script « erreur.sh » suivant :

```
# Je suis un commentaire qui n'a rien a faire ici
#!/bin/sh
for i in *
do
  echo $i
done
```

Si l'on exécute le script précédent, on obtient selon le shell de la session :

```
% echo $SHELL
/bin/csh

% ./erreur.sh
for: Command not found.
do: Command not found.
i: Undefined variable.
```

```
% echo $SHELL
/bin/bash

% ./erreur.sh
a
erreur.sh
repertoire1
```

Moralité :

LA PREMIERE LIGNE DU SCRIPT DOIT ETRE CELLE EN « `#!/bin/sh` »

Méthode 1 pour debugger un script :

> *The most effective debugging tool is still careful thought, coupled with judiciously placed print statements. (Brian Kernighan [1978])*

Par exemple :

```
#!/bin/sh

n="quelque chose"
echo "n étape 1 = $n"
n="quelque chose d'autre"
echo "n étape 2 = $n"
```

Méthode 2 pour debugger un script :
placer la ligne « set -x » lorsque l'on veut que le debug commencer :

```
#!/bin/sh
set -x
echo $USER
```

Si vous souhaitez debugger tout le script, faites le commencer par « #!/bin/sh -x » :

```
#!/bin/sh -x
echo $USER
```


L'option « -x » du shell convertit les metacharacters du shell avant d'afficher la ligne ainsi convertie **avant son exécution**.

On obtient ici :

```
% ./exemple.sh
+ echo besancon
besancon
```

(les explications sur les metacharacters seront vues dans les pages à venir)

La commande « **exit** » renvoie une valeur de retour pour le shell script et provoque l'arrêt de l'exécution du script.

La valeur de retour est un entier compris entre 0 et 255.

Le code de retour d'un shell script suit la même convention que pour les commandes UNIX :

- code de retour nul
  le script s'est exécuté correctement
- code de retour non nul
  le script a rencontré une condition logique d'erreur

Exemple de script :

```
#! /bin/sh
exit 0
```


Comme tout programme, on peut passer des paramètres à un shell script.

Variable	Description
$0	Nom du shell script
$1 à $9	Les 9 premiers paramètres
$#	Le nombre de paramètres
$*	Tous les paramètres passés au shell script sous la forme de mots individuels séparés
$@	Tous les paramètres passés au shell script

Exemple : soit le script `exemple.sh` suivant :

```
#!/bin/sh
echo "Parametre 1 : $1"
echo "Parametre 2 : $2"
```

Son exécution donne :

```
% ./exemple.sh AAAAA BBBBB
Parametre 1 : AAAAA
Parametre 2 : BBBBB
```


Comment accéder à tous les paramètres ?

Soit le script `erreur.sh` suivant :

```
#!/bin/sh
echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12
```

Son exécution donne :

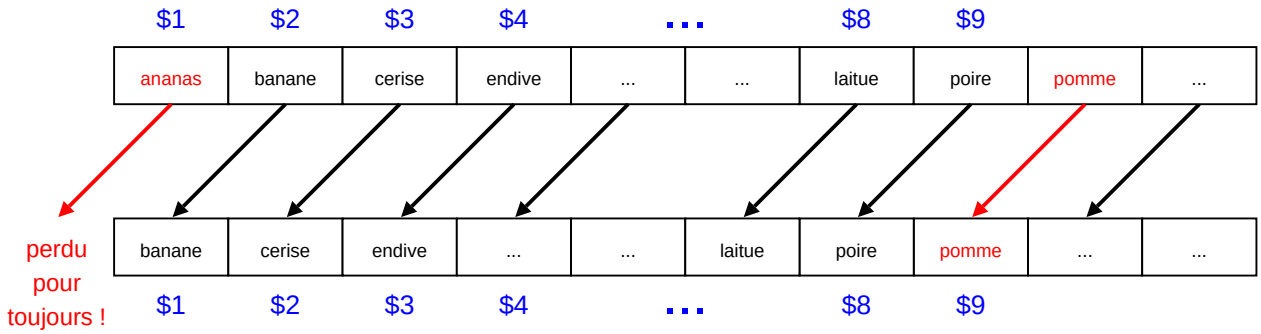
```
% ./erreur.sh a b c d e f g h i j k l
./erreur.sh a b c d e f g h i a0 a1 a2
```

Pourquoi ces résultats ?

Conclusion ⇒ comment faire ?

La solution consiste à utiliser la commande « **shift** ».

(en anglais *shift*, décalage)



Soit le script `exemple.sh` suivant :

```
#!/bin/sh
echo $1 $2 $3
shift
echo $1 $2 $3
```

Son exécution donne :

```
% ./exemple.sh a b c d
a b c
b c d
```


Attention !

A chaque emploi de `shift`, le paramètre `$1` précédent est perdu. Du même coup, ce paramètre est supprimé de `$*` et `$@`, `$#` est décrémenté de 1.

```
#!/bin/sh
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
```

```
% ./exemple.sh a b c d
4 ; a b c ; a b c d
3 ; b c d ; b c d
2 ; c d  ; c d
1 ; d   ; d
0 ;    ;
```

Attention !
L'emploi de `shift` nécessite que le shell script ait au moins un paramètre :

```
#!/bin/sh
echo $1 $2 $3
shift
echo $1 $2 $3
```

```
% ./exemple.sh <-- pas de paramètres
<-- ligne vide du premier echo
shift: can't shift that many
```


Attention : avec BASH on peut accèder à plus que les 9 premiers paramètres !

Pour cela utiliser l'écriture entre accolades :

```
#!/bin/bash

echo $1 $11 $21
echo $1 ${11} ${21}
```

Cela donne :

```
% ./exemple.sh a b c d e f g h i j k l m n o p q r s t u v w
a a1 b1
a k u
```

Deux syntaxes pour la liste des paramètres :

- « $* » : Tous les paramètres passés au shell script sous la forme de mots individuels séparés
- « $@ » : Tous les paramètres passés au shell script

Soit un shell script que l'on appelle ainsi :

```
% ./exemple.sh   "ananas"   "deux mots"   "cerise"
```

Alors :

« $* » est une liste de **4** éléments :

1. "ananas"
2. "deux"
3. "mots"
4. "cerise"

« $@ » est une liste de **3** éléments :

1. "ananas"
2. "deux mots"
3. "cerise"

◇ Guillemets et « $* »

L'écriture « "$*" » se traduit par :

```
"mot1 mot2 mot3 ..."
```

◇ Guillemets et « $@ »

L'écriture « "$@" » se traduit par :

```
"chaine1" "chaine2" "chaine3" ...
```

La variable « $? » contient le code de retour de la dernière commande exécutée.

On ne peut que consulter cette variable.

Soit le script « exemple.sh » suivant avec ses commentaires explicatifs :

```
#!/bin/sh

# Hypothèse : le fichier fichier1 existe
ls -l fichier1
echo $?

# Hypothèse : le fichier fichier2 n'existe pas
ls -l fichier2
echo $?
```

Son exécution donne :

```
% ./exemple.sh
-rw-r--r--   1 besancon ars         0 Jul  9 00:27 fichier1
0
fichier2: No such file or directory
2
```

◇ Lien avec le langage C (1)

Soit le programme C :

```
#include<stdio.h>

int main()
  {
    exit(23);
  }
```

Alors :

```
% gcc exemple.c -o exemple.exe
% ./exemple.exe
% echo $?
23
```


◇ Lien avec le langage C (2)

Rappel : la fonction « `main()` » renvoie un entier.

Soit le programme C :

```
#include<stdio.h>

int main()
  {
    return(23);
  }
```

Alors :

```
% gcc exemple.c -o exemple.exe
% ./exemple.exe
% echo $?
23
```

La variable « $$ » contient le PID du shell script qui est en train de s'exécuter.

On ne peut que consulter cette variable.

Soit le script « exemple.sh » suivant :

```
#!/bin/sh
sleep 30
echo Je suis le processus $$
```

Son exécution donne :

```
% ./exemple.sh
Je suis le processus 1406
```

On peut en parallèle (grâce aux 30 secondes du sleep) chercher par la commande « ps » le numéro de processus :

```
% ps -edf | grep exemple.sh
besancon  1406 28602  0 00:31:02 pts/1    0:00 /bin/sh ./exe
```

Utilisation classique : création de fichiers temporaires uniques associés au script

```
#!/bin/sh
temporaire=/tmp/exemple.$$
touch $temporaire
ls -l $temporaire
```

Son exécution donne

```
% ./exemple.sh
-rw-r--r--  1 besancon ars  0 Jul 9 00:34 /tmp/exemple.1416
```

Une autre exécution donne

```
% ./exemple.sh
-rw-r--r--  1 besancon ars  0 Jul 9 00:35 /tmp/exemple.1419
```


(en anglais *built in*)

Le shell dispose de commandes internes (*builtins*) : « `cd` », « `set` », etc.

Si une commande est builtin, elle est programmée dans le code C du shell.

Si une commande n'est pas builtin, elle correspond à un exécutable dans l'arborescence du système.

Attention, passage difficile :
La commande « `cd` » est un builtin dans tous les shells.

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
cd /
```

On se trouve initialement dans « `/tmp` ».
On lance le shell script.
Où se retrouve-t-on ?

Une méthode pour identifier les builtins est d'utiliser la commande du Bourne Shell « **type** » :

```
% type cd
cd is a shell builtin

% type echo
echo is a shell builtin

% type ls
ls is /bin/ls
```

La commande d'affichage de caractères est « **echo** ».

Attention !
La commande « echo » peut ne pas être un builtin du shell.
En Bourne Shell, c'est toujours un builtin.

Il existe une commande UNIX /bin/echo :

```
% ls -l /bin/echo
-r-xr-xr-x  1 bin  bin  32768 May 20 12:30 /bin/echo
```

Le comportement du builtin peut être différent de celui de la commande UNIX.

◇ Première syntaxe possible pour echo

La commande echo comprend des séquences semblables à celles de « printf() » du langage C (commande echo d'inspiration System-V) :

Séquence	Description
\b	Backspace
\c	Pas de newline envoyé
\n	Newline
\r	Carriage return
\t	Tabulation horizontale
\v	Tabulation verticale
\\	Backslash
\nnn	Caractère dont le code octal ASCII est donné

C'est le cas du « echo » du shell « sh » sur SUN SOLARIS.

Les séquences spéciales de « echo » fonctionnent entre apostrophes ou guillemets.

Cf page 543 pour le sens des apostrophes et des guillements.

Cf page 549 pour le sens de antislash entre guillemets.

C'est cohérent...

Exemple de cette syntaxe :
Soit le script « exemple.sh » suivant :

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
echo "a*b="
expr $a \* $b
# Affichage correct
echo "a*b=\c"
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
a*b=6
```

◇ Seconde syntaxe possible pour `echo`

La commande « `echo` » comprend des options (commande echo d'inspiration BSD) :

- option « `-n` » : Pas de newline envoyé

C'est le cas de « `echo` » du shell « `bash` » sur LINUX.

Exemple de cette syntaxe :

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
echo "a*b="
expr $a \* $b
# Affichage correct
echo -n "a*b="
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
a*b=6
```

Exemple : qu'arrive-t-il si l'on se trompe de syntaxe ?
(ici syntaxe BSD avec un echo de syntaxe System-V)

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
/bin/echo "a*b="
expr $a \* $b
# Affichage correct
/bin/echo -n "a*b="
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
-n a*b= <--- Vous noterez le -n !
6
```


(en anglais *read*)
La commande « **read** » permet de lire au clavier et de placer les mots lus dans une liste de variables.
C'est l'équivalent de « `scanf()` ».

Syntaxe :

```
read variable-list
```

Le premier mot va dans la première variable, le deuxième mot va dans la deuxième variable. . .Tous les mots en trop sont stockés dans la dernière variable mentionnée.

◇ Exemple 1

Soit le script « exemple.sh suivant :

```
#!/bin/sh
echo "Entrez quelque chose : \c"
read reponse
echo "Vous avez entré : $reponse"
```

Son exécution donne :

```
% ./exemple.sh
Entrez quelque chose : il etait une fois
Vous avez entré : il etait une fois
```

On notera au passage l'utilisation de « \c » dans la première ligne « echo » pour coller la réponse entrée au texte de la question.

◇ Exemple 2

Soit le script « exemple.sh » suivant :

```
#!/bin/sh
read variable1 variable2
echo "Premiere variable : $variable1"
echo "Seconde  variable : $variable2"
```

Son exécution donne :

```
% ./exemple.sh
Unix MS-DOS Windows 95 Windows NT MacOS
Premiere variable : Unix
Seconde  variable : MS-DOS Windows 95 Windows NT MacOS
```

Syntaxes :

1
```
if condition-est-vraie
then
  bloc-de-commandes-unix
fi
```

2
```
if condition-est-vraie
then
  bloc-1-de-commandes-unix
else
  bloc-2-de-commandes-unix
fi
```

La condition (booléenne) est en général le code de retour d'une commande UNIX. Le code de retour de la commande détermine le test « `if` » :

- Code de retour valant zéro :
  Le test « `if` » est vrai.
- Code de retour non nul :
  Le test « `if` » est faux.

Conseils :

- Ne pas utiliser une autre forme possible :

```
if condition-1-est-vraie
then
  bloc-1-de-commandes-unix
elif condition-2-est-vraie
  bloc-2-de-commandes-unix
fi
```

  car rapidement illisible à mon goût
- Indenter les blocs pour être lisible

Exemple :

```
#!/bin/sh
if ls > /dev/null
then
  echo Il y a des fichiers
fi
```


Forme générique d'un « if » dans un shell script :

```
#!/bin/sh
if commande [options] parametres > resultats.txt 2> erreurs.txt
then
  # code de retour (exit) valant 0
  bloc-1-de-commandes-unix
else
  # code de retour (exit) différent de 0
  bloc-2-de-commandes-unix
fi
```

La commande « case » permet de tester une chaîne de caractères par rapport à un certain nombre d'autres chaînes prédéfinies :

```
case chaine-a-tester in
  possibilite1) bloc-1-de-commandes-unix
                ;;
  possibilite2) bloc-2-de-commandes-unix
                ;;
...
  possibiliteN) bloc-N-de-commandes-unix
                ;;
esac
```


◇ Exemple 1 : forme simple

Les possibilités sont de simples chaines de caractères statiques :

```
#!/bin/sh
echo -n "Donnez un chiffre entre 1 et 3 -->"
read reponse
case "$reponse" in
  "1") echo "Vous avez entré le chiffre 1"
       ;;
  "2") echo "Vous avez entré le chiffre 2"
       ;;
  "3") echo "Vous avez entré le chiffre 3"
       ;;
   * ) echo "Erreur"
       exit 1
       ;;
esac
exit 0
```

◇ Exemple 2 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Donnez un chiffre entre 1 et 5 -->"
read reponse
case "$reponse" in
  [1-5]) echo "Le chiffre est bien entre 1 et 5. Merci."
         ;;
      *) echo "Erreur"
         exit 1
         ;;
esac
exit 0
```


◇ Exemple 3 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Entrer le mot unix ou windows ou macintosh --> "
read reponse
case "$reponse" in
  unix | windows) echo "Gagné !"
                  ;;
       macintosh) echo "Perdu !"
                  ;;
              *) echo "Réponse non autorisée !"
                 exit 1
                 ;;
esac
exit 0
```

◇ Exemple 4 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Entrer un mot français -->"
read reponse
case "$reponse" in
  [aeiouy]*) echo "Le mot commence par une voyelle."
             ;;
     [0-9]*) echo "Le mot commence par un chiffre"
             ;;
          *) echo "Le mot commence par autre chose."
             ;;
esac
exit 0
```


Dans de nombreuses structures shell, on teste une condition.
La commande « **test** » permet de réaliser divers tests.

Liste partielle (lisez la page de manuel de `test` pour compléter) :

Format	**Description**
« `-d objet` »	Vrai si l'objet existe et est un répertoire (en anglais *directory*)
« `-f objet` »	Vrai si l'objet existe et est un fichier (en anglais *file*)
« `-s objet` »	Vrai si l'objet existe, est un fichier et a une taille supérieure à zéro (en anglais *size*)
« `-w objet` »	Vrai si l'objet existe et que l'on peut écrire dans l'objet (en anglais *writable*)
« `-x objet` »	Vrai si l'objet existe et que l'on peut l'exécuter (en anglais *executable*)

Format	Description
« `-n string` »	Vrai si la chaine est non vide.
« `s1 = s2` »	Vrai si les chaines s1 et s2 sont identiques.
« `s1 != s2` »	Vrai si les chaines s1 et s2 ne sont pas identiques.
« `n1 -eq n2` »	Vrai si les chaines n1 et n2 sont mathématiquement égales (en anglais *equal*).
« `n1 -ne n2` »	Vrais si les chaines n1 et n2 ne sont pas mathématiquement égales (en anglais *not equal*).
« `n1 -gt n2` »	Vrai si la chaine n1 est mathématiquement strictement supérieure à n2 (en anglais *greater than*).
« `n1 -ge n2` »	Vrai si la chaine n1 est mathématiquement supérieure ou égale à n2 (en anglais *greater or equal*).
« `n1 -lt n2` »	Vrai si la chaine n1 est mathématiquement strictement inférieure à n2 (en anglais *less than*).
« `n1 -le n2` »	Vrai si la chaine n1 est mathématiquement inférieure ou égale à n2 (en anglais *less or equal*).

Format	Description
« `! expression` »	Vrai si l'expression est fausse.
« `expression1 -a expression2` »	Vrai si expression1 et expression2 sont vraies (en anglais *and*)
« `expression1 -o expression2` »	Vrai si expression1 ou expression2 est vraie (en anglais *or*)

Attention à ne pas baptiser du nom « `test` » un de vos scripts à vous. La commande « `test` » du système pourrait avoir priorité sur votre script selon le PATH si bien que votre script ne tournerait jamais.

⇒ Baptisez vos scripts de test (et aussi exécutables) de noms comme « `essai` », etc.

Double forme de la commande `test` :

1. forme normale :

```
#!/bin/sh
if test "$1" = hello
then
  echo hello world
fi
```

2. forme crochet :

```
#!/bin/sh
if [ "$1" = hello ]
then
  echo hello world
fi
```

Vérification :

```
% ls -li /bin/[ /bin/test
15422 -r-xr-xr-x  2 bin  bin  45056 May 20 12:31 /bin/[
15422 -r-xr-xr-x  2 bin  bin  45056 May 20 12:31 /bin/test
```

Par contre pas de commande UNIX « `]` », bien sûr.

Exemple 1 :

```
#!/bin/sh
if test "$1" = hello
then
  echo hello world
fi
```

Exemple 2 :

```
if [ $1 -gt $2 -o $1 -eq $2 ]
then
  echo $1 is greater than or equal to $2
fi
```

Exemple 3 :

```
#!/bin/sh
[ $# -eq 0 ] && echo You entered no parameters
```


ATTENTION : si vous avez bien lu les pages précédentes, vous aurez remarqué l'emploi de guillemets :

```
#!/bin/sh
if test "$1" = hello
then
  echo hello world
fi
```

```
#!/bin/sh
echo "Donnez un chiffre entre 1 e
read reponse
case "$reponse" in
  [1-5]) echo "Le chiffre est bie
         ;;
      *) echo "Erreur"
         exit 1
         ;;
esac
exit 0
```

Pourquoi ?

◇ Exemple 1 : peut-on ne rien mettre ?

Soit le script :

```
#!/bin/sh
if test $1 = "cerise"
then
  echo OK
fi
```

Il est incorrect comme on peut le voir ainsi :

```
% ./exemple.sh "ananas banane"
./exemple.sh: test: unknown operator banane
```


Explications :
Utiliser l'option « -x » de SH pour démontrer l'erreur (voir page 651) :

```
#!/bin/sh -x
if test $1 = "cerise"
then
  echo OK
fi

% ./exemple.sh "ananas banane"
+ test ananas banane = cerise
./exemple.sh: test: unknown operator banane
```

La ligne de test est en fait devenue :

```
if test ananas banane = cerise
```

Syntaxe fausse !

◇ Exemple 2 : peut-on mettre des apostrophes ?

Soit le script :

```
#!/bin/sh
if test '$1' = "cerise"
then
  echo OK
fi
```

Il est incorrect comme on peut le voir ainsi :

```
% ./exemple.sh "cerise"
<-- pas de résultat
```


Explications :
Utiliser l'option « -x » de SH pour démontrer l'erreur (voir page 651) :

```
#!/bin/sh -x
if test '$1' = "cerise"
then
  echo OK
fi

% ./exemple.sh "cerise"
+ test $1 = cerise
```

La ligne est donc devenue :

```
if test '$1' = cerise
```

Pas d'erreur de syntaxe mais de logique !

◇ Exemple 3 : la solution avec les guillemets

Soit le script :

```
#!/bin/sh
if test "$1" = "cerise"
then
  echo OK
fi
```

Il est correct comme on peut le voir ainsi :

```
% ./exemple.sh "cerise"
OK <-- résultat correct
% ./exemple.sh "ananas banane"
<-- pas de résultat mais c'est correct
```


Explications :
Utiliser l'option « -x » de SH pour démontrer le bon fonctionnement (voir page 651) :

```
#!/bin/sh -x
if test "$1" = "cerise"
then
  echo OK
fi
```

```
% ./exemple.sh "cerise"
+ test cerise = cerise
+ echo OK
OK
```

```
% ./exemple.sh "ananas banane"
+ test ananas banane = cerise
```

Syntaxes :

```
while condition
do
  bloc-cmdes-unix
done
```

```
for variable in list
do
  bloc-cmdes-unix
done
```

```
until condition
do
  bloc-cmdes-unix
done
```

Ces 3 formes sont équivalentes.

En pratique, on choisira la forme pour laquelle la condition s'exprime le plus facilement ou le plus naturellement.

◇ Exemple 1a

```
#!/bin/sh
while [ "$1" ]
do
  echo $1
  shift
done
```

```
% ./exemple.sh a b "deux mots" d e
a
b
deux mots
d
e
```

◇ Exemple 1b

```
#!/bin/sh
compteur=5
while [ $compteur -ge 0 ]
do
  echo $compteur
  compteur=`expr $compteur - 1`
done
```

```
% ./exemple.sh
5
4
3
2
1
0
```


◇ Exemple 2a

```
#!/bin/sh
echo $#
for i in "$@"
do
  echo $i
done
```

```
% ./exemple.sh a b "deux mots" d e
5
a
b
deux mots
d
e
```

◇ Exemple 2b

```
#!/bin/sh
compteur=0
for i in "$@"
do
  compteur=`expr $compteur + 1`
  echo "argv[$compteur]=$i"
done
```

```
% ./exemple.sh a b "deux mots" d e
argv[1]=a
argv[2]=b
argv[3]=deux mots
argv[4]=d
argv[5]=e
```


◇ Exemple 3a

```
#!/bin/sh
until [ "$1" = "" ]
do
  echo $1
  shift
done
```

```
% ./exemple.sh a b "deux mots" d e
a
b
deux mots
d
e
```

◇ Exemple 3b

```
#!/bin/sh
compteur=5
until [ $compteur -lt 0 ]
do
  echo $compteur
  compteur=`expr $compteur - 1`
done
```

```
% ./exemple.sh
5
4
3
2
1
0
```


Chapitre 14 • Programmation en Bourne shell
§14.19 • Techniques classiques de boucle for

◇ Exemple 1

Changer l'extension « .txt » en l'extension « .doc » de tous les fichiers du répertoire courant :

```
#!/bin/sh
for i in *.txt
do
  mv $i `basename $i .txt`.doc
done
```

◇ Exemple 2

Changer l'extension « `.txt` » en l'extension « `.doc` » de tous les fichiers indiqués dans le fichier `liste.txt` :

```
#!/bin/sh
for i in `cat liste.txt`
do
  mv $i `basename $i .txt`.doc
done
```


◇ Exemple 3

Changer l'extension « `.txt` » en l'extension « `.doc` » de tous les fichiers indiqués en paramètres sur la ligne de commande :

```
#!/bin/sh
for i in "$@"
do
  mv $i `basename $i .txt`.doc
done
```

(voir page 663 pour les explications sur la syntaxe « `"$@"` »)

Deux commandes du shell permettent de contrôler le flux d'exécution du code :

- « `break` »
  fait sortir prématurément de la boucle courante
- « `break n` »
  fait sortir prématurément de « n » niveaux de boucles imbriquées
- « `continue` »
  fait finir prématurément la boucle courante
- « `continue n` »
  fait finir prématurément « n » niveaux de boucles imbriquées


```
#!/bin/sh

for i in *
do
  if [ ! -d "$i" ]
  then
    # Inutile de continuer si on n'a pas un répertoire.
    continue
  fi

  cd $i
  blabla...
  cd ..
done
```

La commande « `script` » sert à enregistrer le texte d'une session shell.
Syntaxe : `script [fichier]`

A noter :

- S'il n'y a pas de paramètre, alors le texte est enregistré dans le fichier appelé « `typescript` ».
- On trouve le caractère « `\r` » (c'est-à-dire Ctrl-M) en fin de chaque ligne de la session.
- En première ligne se trouve la date du début d'enregistrement du texte.
- En dernière ligne se trouve la date de la fin d'enregistrement du texte.

◇ Exemple 1 :

```
% script
Script started, file is typescript
sh-2.05$ ls
exemple.txt  typescript
sh-2.05$ Script done, file is typescript

% cat typescript
Script started on Sun Oct 03 22:42:59 2004
sh-2.05$ ls
exemple.txt  typescript
sh-2.05$
script done on Sun Oct 03 22:43:03 2004
```

◇ Exemple 2 :

```
% script session.txt
Script started, file is session.txt
sh-2.05$ date
Sun Oct  3 22:50:57 MEST 2004
sh-2.05$ Script done, file is session.txt

% od -c session.txt
0000000   S   c   r   i   p   t       s   t   a   r   t   e   d
0000020   n       S   u   n       O   c   t       0   3       2
0000040   5   0   :   5   3       2   0   0   4  \n   s   h   -
0000060   0   5   $       d   a   t   e  \r  \n   S   u   n
0000100   t           3       2   2   :   5   0   :   5   7
0000120   S   T       2   0   0   4  \r  \n   s   h   -   2   .
0000140   $      \n   s   c   r   i   p   t       d   o   n   e
0000160   n       S   u   n       O   c   t       0   3       2
0000200   5   1   :   0   0       2   0   0   4  \n
0000213
```

Chapitre 15

Programmation en langage AWK

Nom « awk » déduit des noms des auteurs (*Aho, Weinberger, Kernighan*)

C'est un utilitaire recherchant des motifs dans un fichier et réalisant des opérations sur les lignes répondant aux critères.

C'est plus généralement un mini langage de programmation à la syntaxe proche du langage C.

Il est très souvent utilisé pour réaliser des filtres sur des fichiers.

Se reporter à un manuel disponible en français :
`ftp://ftp.imag.fr/pub/DOC.UNIX/AWK/awk.pdf`
`ftp://ftp.imag.fr/pub/DOC.UNIX/AWK/awk.ps.gz`

Syntaxe : **awk [options] fichiers**

Options intéressantes :

- option « -FC » où C est le caractère séparateur de champs
- option « -f exemple.awk » pour indiquer le fichier « exemple.awk » contenant le programme awk à exécuter

Un programme AWK peut être composé de :

- définition de fonctions
- instructions

◇ Définition d'une fonction

```
function nom(paramêtres) { instructions }
```

◇ Format des instructions :

```
[masque] { instructions }
```

◇ Variables utilisables :

- champs : « $0 », « $1 », « $2 », ...
- variables prédéfinies
- variables utilisateur

« FS »	caractère séparateur de champs
« NF »	nombre de champs sur la ligne courante
« NR »	numéro d'ordre de la ligne courante
« RS »	caractère séparateur de lignes
« OFS »	caractère séparateur de champs, en sortie
« ORS »	caractère séparateur de lignes, en sortie
« FILENAME »	nom du fichier en cours de traitement
« ENVIRON[] »	variables d'environnement

Les masques conditionnent les instructions à exécuter sur chaque ligne.

Les masques sont évalués lors de la lecture de chaque ligne.

Seuls les instructions associées aux masques actifs sont exécutées pour chaque ligne.

Principaux masques :

- « BEGIN »
- « END »
- « /expression rationnelle/ »

++	incrémentation
--	décrémentation
!	négation logique
* / % + -	multiplication, division, reste, addition, soustraction
< > <= >=	infériorité, supériorité
== =|	égalité, différence
&&	ET logique
||	OU logique

◇ Test

```
if (condition) instruction [ else instruction]
```

◇ Boucles

```
while (condition) instruction
do instruction
while(condition)
for (expr1; expr2; expr3) instruction
```

◇ Contrôle de boucles

```
break
continue
```

◇ Terminaison

```
exit [code_de_retour]
```

◇ Passage à la ligne suivante

```
next
```

◇ Affichage de données

```
print [expressions] [>fichier]
printf format, expressions [>fichier]
```


◇ Longueur d'une chaine

```
length (chaine)
```

◇ Valeur entière d'une expression

```
int(expression)
```

◇ Recherche d'une chaine dans une autre

```
index(chaine1, chaine2)
```

◇ Extraction d'une sous-chaine

```
substr(chaine, position, longueur)
```

Exemples :

- Affichage de tous les noms d'utilisateurs
  ```
  awk -F: '{ print $1 }' < /etc/passwd
  ```
- Inversion des champs 3 et 4
  ```
  awk '{ print $1, $2, $4, $3 }'
  ```
- Affichage du contenu d'un fichier avec numérotation des lignes
  ```
  awk '{ print NR, $0 }'
  ```
- Cumul des sommes présentes en troisième colonne
  ```
  awk '{ s += $3 } END { print s }'
  ```
- Suppression de toutes les lignes dont le premier champ est égal à celui de la ligne précédente
  ```
  awk '{if ( $1 != prev ) { print; prev = $1; } }'
  ```


- Vérification que toutes les lignes ont le même nombre de champs que la première

```
BEGIN { nberr = 0}
{ if (NR == 1)
    nb = NF;
  else
    if ( nb != NF )
      nberr++;
}
END {
if ( nberr != 0 )
  print nberr , " enregistrements incorrects ";
}
```

- Tuer tous les processus UNIX appartenant à l'utilisateur « thb » :

```
% ps aux | awk '$1=="thb" {print "kill " $2}' | sh
```

Ci joint dans la version imprimée de ce cours, une news USENET expliquant en quoi CSH est un mauvais shell.

Path: senator-bedfellow.mit.edu!bloom-beacon.mit.edu!news.mathworks.com!uunet!in1.uu.net!csnews!mox.perl.com!tchrist
From: Tom Christiansen <tchrist@mox.perl.com>
Newsgroups:
comp.unix.shell,comp.unix.questions,comp.unix.programmer,comp.answers,news.answers,comp.infosystems.www.authoring.cgi
Subject: Csh Programming Considered Harmful
Followup-To: comp.unix.shell
Date: 6 Oct 1996 14:03:18 GMT
Organization: Perl Training and Consulting
Lines: 558
Approved: news-answers-request@MIT.Edu
Expires: Sun, 1 Dec 1996 12:00:00 GMT
Message-ID: <538e76$8uq$1@csnews.cs.colorado.edu>
NNTP-Posting-Host: perl.com
Originator: tchrist@mox.perl.com
Xref: senator-bedfellow.mit.edu comp.unix.shell:42768 comp.unix.questions:118257 comp.unix.programmer:50081 comp.answers:
21606 news.answers:83685 comp.infosystems.www.authoring.cgi:44411

Archive-name: unix-faq/shell/csh-whynot
Version: $Id: csh-faq,v 1.7 95/09/28 12:52:17 tchrist Exp Locker: tchrist $

The following periodic article answers in excruciating detail
the frequently asked question "Why shouldn't I program in csh?".
It is available for anon FTP from perl.com in /pub/perl/versus/csh.whynot.gz

*** CSH PROGRAMMING CONSIDERED HARMFUL ***

Resolved: The csh is a tool utterly inadequate for programming,
and its use for such purposes should be strictly banned!

I am continually shocked and dismayed to see people write test cases,
install scripts, and other random hackery using the csh. Lack of
proficiency in the Bourne shell has been known to cause errors in /etc/rc
and .cronrc files, which is a problem, because you *must* write these files
in that language.

The csh is seductive because the conditionals are more C-like, so the path
of least resistance is chosen and a csh script is written. Sadly, this is
a lost cause, and the programmer seldom even realizes it, even when they
find that many simple things they wish to do range from cumbersome to
impossible in the csh.

1. FILE DESCRIPTORS

The most common problem encountered in csh programming is that
you can't do file-descriptor manipulation. All you are able to
do is redirect stdin, or stdout, or dup stderr into stdout.
Bourne-compatible shells offer you an abundance of more exotic
possibilities.

1a. Writing Files

In the Bourne shell, you can open or dup arbitrary file descriptors.
For example,

```
exec 2>errs.out
```

means that from then on, stderr goes into errs file.

Or what if you just want to throw away stderr and leave stdout
alone? Pretty simple operation, eh?

```
cmd 2>/dev/null
```

Works in the Bourne shell. In the csh, you can only make a pitiful
attempt like this:

```
(cmd > /dev/tty) >& /dev/null
```

But who said that stdout was my tty? So it's wrong. This simple
operation *CANNOT BE DONE* in the csh.

Along these same lines, you can't direct error messages in csh scripts
out stderr as is considered proper. In the Bourne shell, you might say:

```
echo "$0: cannot find $file" 1>&2
```

but in the csh, you can't redirect stdout out stderr, so you end
up doing something silly like this:

```
sh -c 'echo "$0: cannot find $file" 1>&2'
```

1b. Reading Files

In the csh, all you've got is $<, which reads a line from your tty. What
if you've redirected stdin? Tough noogies, you still get your tty, which
you really can't redirect. Now, the read statement
in the Bourne shell allows you to read from stdin, which catches
redirection. It also means that you can do things like this:

```
exec 3<file1
exec 4<file2
```

Now you can read from fd 3 and get lines from file1, or from file2 through
fd 4. In modern, Bourne-like shells, this suffices:

```
read some_var 0<&3
read another_var 0<&4
```

Although in older ones where read only goes from 0, you trick it:

```
exec 5<&0  # save old stdin
exec 0<&3; read some_var
exec 0<&4; read another_var
exec 0<&5  # restore it
```

1c. Closing FDs

In the Bourne shell, you can close file descriptors you don't want open, like 2>&−, which isn't the same as redirecting it to /dev/null.

1d. More Elaborate Combinations

Maybe you want to pipe stderr to a command and leave stdout alone. Not too hard an idea, right? You can't do this in the csh as I mentioned in 1a. In a Bourne shell, you can do things like this:

```
exec 3>&1; grep yyy xxx 2>&1 1>&3 3>&− | sed s/file/foobar/ 1>&2 3>&−
grep: xxx: No such foobar or directory
```

Normal output would be unaffected. The closes there were in case something really cared about all its FDs. We send stderr to sed, and then put it back out 2.

Consider the pipeline:

```
A | B | C
```

You want to know the status of C, well, that's easy: it's in $?, or $status in csh. But if you want it from A, you're out of luck −− if you're in the csh, that is. In the Bourne shell, you can get it, although doing so is a bit tricky. Here's something I had to do where I ran dd's stderr into a grep −v pipe to get rid of the records in/out noise, but had to return the dd's exit status, not the grep's:

```
device=/dev/rmt8
dd_noise='^[0−9]+\+[0−9]+ records (in|out)$'
exec 3>&1
status='((dd if=$device ibs=64k 2>&1 1>&3 3>&− 4>&−; echo $? >&4) |
            egrep −v "$dd_noise" 1>&2 3>&− 4>&−) 4>&1'
exit $status;
```

The csh has also been known to close all open file descriptors besides the ones it knows about, making it unsuitable for applications that intend to inherit open file descriptors.

2. COMMAND ORTHOGONALITY

2a. Built−ins

The csh is a horrid botch with its built−ins. You can't put them together in many reasonable ways. Even simple little things like this:

```
% time | echo
```

which while nonsensical, shouldn't give me this message:

```
Reset tty pgrp from 9341 to 26678
```

Others are more fun:

```
% sleep 1 | while
while: Too few arguments.
[5] 9402
% jobs
[5]     9402 Done                 sleep |
```

Some can even hang your shell. Try typing ^Z while you're sourcing something, or redirecting a source command. Just make sure you have another window handy. Or try

```
% history | more
```

on some systems.

Aliases are not evaluated everywhere you would like them do be:

```
% alias lu 'ls −u'
% lu
HISTORY  News     bin      fortran  lib      lyrics   misc     tex
Mail     TEX      dehnung  hpview   logs     mbox     netlib
% repeat 3 lu
lu: Command not found.
lu: Command not found.
lu: Command not found.

% time lu
lu: Command not found.
```

2b. Flow control

You can't mix flow−control and commands, like this:

```
who | while read line; do
    echo "gotta $line"
done
```

You can't combine multiline constructs in a csh using semicolons. There's no easy way to do this

```
alias cmd 'if (foo) then bar; else snark; endif'
```

You can't perform redirections with if statements that are evaluated solely for their exit status:

```
if ( { grep vt100 /etc/termcap > /dev/null } ) echo ok
```

And even pipes don't work:

```
    if ( { grep vt100 /etc/termcap | sed 's/$/###' } ) echo ok
```

But these work just fine in the Bourne shell:

```
    if grep vt100 /etc/termcap > /dev/null ; then echo ok; fi

    if grep vt100 /etc/termcap | sed 's/$/###/' ; then echo ok; fi
```

Consider the following reasonable construct:

```
  if ( { command1 | command2 } ) then
      ...
  endif
```

The output of command1 won't go into the input of command2. You will get the output of both commands on standard output. No error is raised. In the Bourne shell or its clones, you would say

```
    if command1 | command2 ; then
        ...
    fi
```

2c. Stupid parsing bugs

Certain reasonable things just don't work, like this:

```
    % kill -1 `cat foo`
    `cat foo`: Ambiguous.
```

But this is ok:

```
    % /bin/kill -1 `cat foo`
```

If you have a stopped job:

```
    [2]     Stopped              rlogin globhost
```

You should be able to kill it with

```
    % kill %?glob
    kill: No match
```

but

```
    % fg %?glob
```

works.

White space can matter:

```
    if(expr)
```

may fail on some versions of csh, while

```
    if (expr)
```

works! Your vendor may have attempted to fix this bug, but odds are good that their csh still won't be able to handle

```
    if(0) then
      if(1) then
          echo A: got here
      else
          echo B: got here
      endif
      echo We should never execute this statement
    endif
```

3. SIGNALS

In the csh, all you can do with signals is trap SIGINT. In the Bourne shell, you can trap any signal, or the end-of-program exit. For example, to blow away a tempfile on any of a variety of signals:

```
    $ trap 'rm -f /usr/adm/tmp/i$$ ;
            echo "ERROR: abnormal exit";
            exit' 1 2 3 15

    $ trap 'rm tmp.$$' 0   # on program exit
```

4. QUOTING

You can't quote things reasonably in the csh:

```
    set foo = "Bill asked, \"How's tricks?\""
```

doesn't work. This makes it really hard to construct strings with mixed quotes in them. In the Bourne shell, this works just fine. In fact, so does this:

```
     cd /mnt; /usr/ucb/finger -m -s `ls \'u\'`
```

Dollar signs cannot be escaped in double quotes in the csh. Ug.

```
    set foo = "this is a \$dollar quoted and this is $HOME not quoted"
    dollar: Undefined variable.
```

You have to use backslashes for newlines, and it's just darn hard to get them into strings sometimes.

```
    set foo = "this \
    and that";
    echo $foo
```

```
    this  and that
    echo "$foo"
    Unmatched ".
```

Say what? You don't have these problems in the Bourne shell, where it's just fine to write things like this:

```
    echo        'This is
             some text that contains
             several newlines.'
```

As distributed, quoting history references is a challenge. Consider:

```
    % mail adec23!alberta!pixel.Convex.COM!tchrist
    alberta!pixel.Convex.COM!tchri: Event not found.
```

5. VARIABLE SYNTAX

There's this big difference between global (environment) and local (shell) variables. In csh, you use a totally different syntax to set one from the other.

In the Bourne shell, this

```
    VAR=foo cmds args
```

is the same as

```
    (export VAR; VAR=foo; cmd args)
```

or csh's

```
    (setenv VAR;  cmd args)
```

You can't use :t, :h, etc on envariables. Watch:

```
        echo Try testing with $SHELL:t
```

It's really nice to be able to say

```
    ${PAGER-more}
```

or

```
    FOO=${BAR:-${BAZ}}
```

to be able to run the user's PAGER if set, and more otherwise. You can't do this in the csh. It takes more verbiage.

You can't get the process number of the last background command from the csh, something you might like to do if you're starting up several jobs in the background. In the Bourne shell, the pid of the last command put in the background is available in $!.

The csh is also flaky about what it does when it imports an environment variable into a local shell variable, as it does with HOME, USER, PATH, and TERM. Consider this:

```
    % setenv TERM ''/bin/ls -l / > /dev/tty''
    % csh -f
```

And watch the fun!

6. EXPRESSION EVALUATION

Consider this statement in the csh:

```
    if ($?MANPAGER) setenv PAGER $MANPAGER
```

Despite your attempts to only set PAGER when you want to, the csh aborts:

```
    MANPAGER: Undefined variable.
```

That's because it parses the whole line anyway AND EVALUATES IT! You have to write this:

```
    if ($?MANPAGER) then
        setenv PAGER $MANPAGER
    endif
```

That's the same problem you have here:

```
    if ($?X && $X == 'foo') echo ok
    X: Undefined variable
```

This forces you to write a couple nested if statements. This is highly undesirable because it renders short-circuit booleans useless in situations like these. If the csh were the really C-like, you would expect to be able to safely employ this kind of logic. Consider the common C construct:

```
    if (p && p->member)
```

Undefined variables are not fatal errors in the Bourne shell, so this issue does not arise there.

While the csh does have built-in expression handling, it's not what you might think. In fact, it's space sensitive. This is an error

```
   @ a = 4/2
```

but this is ok

```
   @ a = 4 / 2
```

The ad hoc parsing csh employs fouls you up in other places as well. Consider:

```
    % alias foo 'echo hi' ; foo
```

```
    foo: Command not found.
    % foo
    hi
```

7. ERROR HANDLING

Wouldn't it be nice to know you had an error in your script before you ran it? That's what the −n flag is for: just check the syntax. This is especially good to make sure seldom taken segments of code code are correct. Alas, the csh implementation of this doesn't work. Consider this statement:

```
    exit (i)
```

Of course, they really meant

```
    exit (1)
```

or just

```
    exit 1
```

Either shell will complain about this. But if you hide this in an if clause, like so:

```
    #!/bin/csh −fn
    if (1) then
        exit (i)
    endif
```

The csh tells you there's nothing wrong with this script. The equivalent construct in the Bourne shell, on the other hand, tells you this:

```
    #!/bin/sh −n
    if (1) then
        exit (i)
    endif

    /tmp/x: syntax error at line 3: `(' unexpected
```

RANDOM BUGS

Here's one:

```
    fg %?string
    ^Z
    kill  %?string
    No match.
```

Huh? Here's another

```
    !%s%x%s
```

Coredump, or garbage.

If you have an alias with backquotes, and use that in backquotes in another one, you get a coredump.

Try this:

```
    % repeat 3 echo "/vmu*"
    /vmu*
    /vmunix
    /vmunix
```

What???

Here's another one:

```
    % mkdir tst
    % cd tst
    % touch '[foo]bar'
    % foreach var ( * )
    > echo "File named $var"
    > end
    foreach: No match.
```

8. SUMMARY

While some vendors have fixed some of the csh's bugs (the tcsh also does much better here), many have added new ones. Most of its problems can never be solved because they're not actually bugs per se, but rather the direct consequences of braindead design decisions. It's inherently flawed.

Do yourself a favor, and if you *have* to write a shell script, do it in the Bourne shell. It's on every UNIX system out there. However, behavior can vary.

There are other possibilities.

The Korn shell is the preferred programming shell by many sh addicts, but it still suffers from inherent problems in the Bourne shell's design, such as parsing and evaluation horrors. The Korn shell or its public−domain clones and supersets (like bash) aren't quite so ubiquitous as sh, so it probably wouldn't be wise to write a sharchive in them that you post to the net. When 1003.2 becomes a real standard that companies are forced to adhere to, then we'll be in much better shape. Until then, we'll be stuck with bug−incompatible versions of the sh lying about.

The Plan 9 shell, rc, is much cleaner in its parsing and evaluation; it is not widely available, so you'd be significantly sacrificing portability. No vendor is shipping it yet.

If you don't have to use a shell, but just want an interpreted language, many other free possibilities present themselves, like Perl, REXX, TCL, Scheme, or Python. Of these, Perl is probably the most widely available on UNIX (and many other) systems and certainly comes with the most extensive UNIX interface. Increasing numbers vendors ship Perl with their standard systems. (See the comp.lang.perl FAQ for a list.)

If you have a problem that would ordinarily use sed or awk or sh, but it exceeds their capabilities or must run a little faster, and you don't want to write the silly thing in C, then Perl may be for you. You can get at networking functions, binary data, and most of the C library. There are also translators to turn your sed and awk scripts into Perl scripts, as well as a symbolic debugger. Tchrist's personal rule of thumb is that if it's the size that fits in a Makefile, it gets written in the Bourne shell, but anything bigger gets written in Perl.

See the comp.lang.{perl,rexx,tcl} newsgroups for details about these languages (including FAQs), or David Muir Sharnoff's comparison of freely available languages and tools in comp.lang.misc and news.answers.

NOTE: Doug Hamilton <hamilton@bix.com> has a program that he sells for profit for little toy non-UNIX systems. He calls it 'csh' or the 'hamilton csh', but it's not a csh as it's neither bug nor feature compatible with the real csh. Actually, he's fixed a great deal, but in doing so, has created a totally different shell.
--
Tom Christiansen   Perl Consultant, Gamer, Hiker   tchrist@mox.perl.com
"Espousing the eponymous /cgi-bin/perl.exe?FMH.pl execution model is like reading a suicide note -- three days too late."
            --Tom Christiansen <tchrist@mox.perl.com>

Chapitre 16

Langage PERL

Ce cours fait l'hypothèse que l'on sait programmer.

Ce cours fait l'hypothèse que l'on connait les bases du langage C.

Ce cours fait l'hypothèse que l'on connait certaines commandes UNIX dont principalement « `grep` » et ses expressions régulières.

Ce cours dure 4 demi journées en 2010-2011 : pour gagner du temps, ce cours va donc à l'essentiel et ne couvre pas l'intégralité des syntaxes autorisées par PERL. On se cantonnera aux écritures les plus simples, les plus propres, les plus proches de celles du langage C et les plus utiles.

Ce cours n'aborde pas les variables de type *references* (genre de pointeurs de structure à la C).

PERL ≡ *Practical Extraction and Report Language*

Programme téléchargeable sur « `http://www.cpan.org` »

Version pour WINDOWS téléchargeable sur « `http://www.activestate.com` » (comporte des extensions adaptées à WINDOWS ; par exemple : manipulation de la base de registres)

Langage de script (donc = format texte) dont la forme sera identique sur UNIX, sur MACOS et sur WINDOWS :

```
#!/usr/bin/perl
# Ceci est un commentaire.

printf("Bonjour, il fait froid !\n") ;
```

Extension « `.pl` » par convention.

Pas de compilation.

Le fichier doit être exécutable (« `chmod a+x exemple.pl` »).

Un commentaire débute par « `#` ».

Les principales ressemblances avec le langage C :

- une instruction se termine toujours par un point-virgule « ; »
- bloc d'instructions délimitées par des accolades « { » et « } »
- nommage des variables
- possibilité d'avoir des fonctions, même notation pour appeler les fonctions
- il reste une notion très puissante de pointeur

Les principales différences par rapport au langage C :

- langage de plus haut niveau
- utilisation plus puissante des tableaux
- pas de gestion par l'utilisateur de la mémoire
- pas de typage des variables

CONCLUSION : on peut programmer en langage PERL quasiment comme en langage C. Pour être propre et lisible, on adoptera une approche langage C.

De nombreuses pages de manuel fournies :

- « `man perl` » : point de renvoi vers d'autres pages de manuel (102 sur ma machine !)
- « `man perlfunc` » : fonctions proposées de base dans PERL
- « `man perldsc` » : documentation des structures compliquées comme les tableaux de tableaux, les tableaux de listes associatifs, les tableaux associatifs de tableaux, etc.
- « `man perlref` » : descriptif de ce que sont les références PERL (pointeurs du langage C en gros)
- etc.

Nombreux livres chez l'éditeur O'Reilly : « `http://www.oreilly.com` »

Et aussi chez d'autres éditeurs...

RICHESSE DE PERL = la multitude de modules d'extension téléchargeables sur « `http ://www.cpan.org` »

Site : CPAN = Comprehensive PERL Archive Network ; voir « `http://www.cpan.org` »

Recherche de modules sur « `http://search.cpan.org` »

Pour utiliser un module, ajouter en début de script :

```
#!/usr/bin/perl

use module1 ;
use module2 ;
...
```

Exemple : appel au module permettant d'écrire des fichiers EXCEL au format natif XLS :

```
#!/usr/bin/perl

use strict ;
use Spreadsheet::WriteExcel ;
...
```


L'écriture « `use` » :

```
#!/usr/bin/perl

use module1 ;
use module2 ;
...
```

s'apparente à l'écriture « `#include` » du langage C :

```
#include <module1.h>
#include <module2.h>
...
```

En PERL, par défaut, on peut faire appel à une variable sans l'avoir définie[23] : **c'est source d'erreurs !** :

```
#!/usr/bin/perl
$aaa = 3 ;
$bbb = 4 ;
$r = $aa + $bbb ;   <-- Erreur ou pas !?
```

Ce cours force donc la définition des variables avant leur utilisation.

[2] Comme en shell.

[3] Par défaut, une variable PERL a la valeur « undef » c'est-à-dire 0 ou chaine vide selon le contexte.

Pour cela : employer le module « strict » qui rend l'écriture du script plus stricte :

```
#!/usr/bin/perl

use strict ;
use module1 ;
use module2 ;
...
```

Les variables doivent alors être déclarées avant d'être utilisées.

La syntaxe pour les déclarer est :

```
my $variable ;   # voir page 750
my @tableau ;    # voir page 758
my %hash ;       # voir page 765
```

Par exemple :

```
#!/usr/bin/perl
use strict ;

my $aaa ;
my $bbb ;
my $r ;

$aaa = 3 ;
$bbb = 4 ;
$r = $aa + $bbb ;
```

Par exemple :

```
#!/usr/bin/perl
use strict ;

my $aaa = 3 ;
my $bbb = 4 ;
my $r ;

$r = $aa + $bbb ;
```

Maintenant indication de l'emploi d'une variable non définie :

```
% ./exemple.pl
Global symbol "$aa" requires explicit package name at
./exemple.pl line 10.
Execution of ./exemple.pl aborted due to compilation
errors.
```



```
#!/usr/bin/perl

use strict ;
use module1 ;
use module2 ;

sub mafonction1
  {
    ...
  }

sub main
  {
    ...
  }

main() ;
```

Simplification : on adoptera la forme suivante analogue au C :

1. définitions des fonctions de l'utilisateur
2. définition de la fonction « `main` »
3. appel à « `main` » pour lancer le programme

Le caractère « $ » spécifie une variable contenant une valeur numérique ou une chaine de caractères.

Mêmes règles qu'en langage C pour nommer la variable :

- Caractères autorisés dans le nom : lettres, chiffres, underscore « _ ».
- Le premier caractère du nom ne peut pas être un chiffre.

Exemples :

```
$a = 10;
$_b = 20;
$s1 = "Bonjour tout le monde !";
```


Pas de distinction entre nombres entiers et nombres flottants.

Tous les calculs internes sont faits en nombres flottants.

```
$a = 10 ;
$a = 10.0 ;
$a = 10.0000 ;
$a = 1e1 ;
$a = 0xa ;
$a = 012 ;
```

- Assignation d'une valeur :

```
$a = 10 ;
```

- Opérations sur les variables :

```
$a = $a + 1 ;
$a += 10 ;
$a /= 20 ;
$c = $a * $b ;
```

- Incrément automatique :

```
$a++ ;
$a = $b++ ;
$b-- ;
```


Une chaine de caractères peut être aussi longue que l'on veut.

Chaines de caractères entre apostrophes « ' » : aucun traitement sur le contenu (à part pour backslash).

```
$s1 = 'jean' ;
$s2 = '' ;
$s3 = 'Aujourd\'hui' ;
$s4 = 'Bonjour\n' ;
$s5 = 'Bonjour.
Il pleut.' ;
$s6 = '\\' ;
```

Une chaine de caractères peut être aussi longue que l'on veut.

Chaines de caractères entre guillemets « " » : remplacement des caractères de formatage ainsi que des variables (terme technique = interpolation).

```
$s1 = "jean" ;
$s2 = "Bonjour $s1" ;
$s3 = "" ;
$s4 = "Bonjour\n" ;
```


Les caractères de formatage (proches du langage C) :

Caractère	Action
\n	Nouvelle ligne
\r	Retour
\t	Tabulation
\b	Backspace
\v	Tabulation verticale
\e	Escape
\\	Backslash
\"	Guillemets
\a	Bruit d'une cloche

- Assignation d'une chaine :

```
$s = "Hello world!" ;
```

- Concaténation de 2 chaines :

```
$s = "${s1}${s2}" ;
$s = $s1 . $s2 ;
```

- Répétition :

```
$s = "abc" x 3 ; # --> "abcabcabc"
```

- Suppression du dernier newline d'une chaine :

```
chomp($s) ;
```


La fonction « `printf()` » fonctionne comme celle du langage C :

```
printf("%s\n", $s) ;
```

La fonction « `sprintf()` » ressemble à celle du langage C :

```
$date = sprintf("%04d-%02d-%02d\n", $annee, $mois, $jour) ;
```

Mêmes séquences de formatage qu'en langage C.

Il existe une fonction « `print()` » mais autant utiliser « `printf()` ».

Le caractère « @ »spécifie une variable liste (ou tableau).

Mêmes règles qu'en langage C pour nommer la variable :

- Caractères autorisés dans le nom : lettres, chiffres, underscore « _ ».
- Le premier caractère du nom ne peut pas être un chiffre.

Exemples :

```
@a = ("un", "deux", "trois", "quatre") ;

$elt = $a[0] ;  # un
$elt = $a[3] ;  # quatre

$elt = $a[-1] ; # quatre
$elt = $a[-2] ; # trois
```


- Assignation d'une liste :

```
@a = ("un", "deux", "trois") ;
@b = (1, 2, 3, 4, 5) ;
```

- Accès en absolu sur un élément de la liste :

```
@a = (1, 2, 3, 4, 5, 6, 7, 8, 9) ;
$elt = $a[4] ;
$a[4] = 0 ;
```

- Taille d'une liste :

```
@a = ("un", "deux", "trois") ;
$taille = @a ;  # taille --> 3
$n = $#a ; # indice du dernier élément --> 2
```

- Ajout d'un ou plusieurs éléments dans une liste :

```
@a = (@a, "6", "7") ;
@a = ("0", @a) ;
```

- Ajouter un ou plusieurs éléments en fin de liste :

```
@a = (1, 2, 3) ;
push(@a, 4, 5) ;
```

- Supprimer le dernier élément de la liste :

```
$elt = pop(@a) ;
```


- Ajouter un ou plusieurs éléments en début de liste :

```
@a = (1, 2, 3) ;
unshift(@a, 4, 5) ;
```

- Supprimer le premier élément de la liste :

```
$elt = shift(@a) ;
```

- Duplication d'une liste :

```
@b = @a ;
```

- Inverser l'ordre des éléments d'une liste :

```
@b = reverse(@a) ;
```

- Trier une liste :

```
@b = sort(@a) ;
```

- Afficher une liste sous forme énumérée :

```
@a = ("un", "deux", "trois") ;
join(", ", @a) --> un, deux, trois
```


En langage C :

```
int main(int argc, char* argv[])
  {
    ...
  }
```

En PERL : liste prédéfinie « `@ARGV` » (« `$ARGV[0]` », « `$ARGV[1]` », etc.)

ATTENTION : décalage d'un cran par rapport au langage C (ou au shell) ! :

- « `$ARGV[0]` » est le premier argument (« `$1` » en shell)
- « `$ARGV[1]` » est le deuxième argument (« `$2` » en shell)
- etc.

Les tableaux/listes utilisent des valeurs numériques pour référencer les données :

- par exemple « `$a[0]` »
- par exemple « `$a[4]` »
- par exemple « `$a[-1]` »
- par exemple « `$a[-2]` »

PERL offre la notion de listes associatives en plus des tableaux classiques.

Les listes associatives utilisent des chaines de caractères pour référencer les données. ULTRA PUISSANT !

Pas d'équivalent dans le langage C, d'où l'intérêt.

Par exemple :

```
$nom{'192.168.216.1'} = "216-pc01" ;

$mails{'besancon'} = 34 ;
$mails{"besancon"} = 34 ;
$mails{besancon} = 34 ;

$n = $mails{besancon} ;
```

Liste associative == hash en anglais

C'est le caractère « % » qui va désigner une liste associative.

Mêmes règles qu'en langage C pour nommer la variable :

- Caractères autorisés dans le nom : lettres, chiffres, underscore « _ ».
- Le premier caractère du nom ne peut pas être un chiffre.

Exemples :

```
%a = {
  "nom" => "Besançon",
  "prenom" => "Thierry",
  "cours" => uc("ars"),
  "annee" => 2009,
} ;
```

- Assignation d'un élément :

```
$a{1.27} = "pomme" ;
$a{ananas} = 1 ;
```

- Suppression d'un élément dans une liste :

```
delete($a{1}) ;
```

- Taille d'une liste

```
$taille = %list ;
```


- Tester l'existence d'un élément :

```
if ( exists($jardin{ananas}) )
  { ... }
```

- Pour récupérer les clefs d'une liste :

```
@liste_clefs = keys(%a) ;
```

- Pour récupérer les valeurs d'une liste :

```
@liste_valeurs = values(%a) ;
```

- Pour parcourir une liste :

```
while ( ($clef, $valeur) = each(%a) )
  {
    printf("clef = %s ; valeur = %s\n", $clef, $valeur);
  }
```

En langage C : fonction « `getenv()` »

En PERL : liste associative « `%ENV` »

```
printf("%s\n", $ENV{HOME}) ;
```


Il n'y a pas de type BOOLEEN en PERL.

Par contre, sont l'équivalent de FAUX (FALSE) les valeurs :

- « `undef` »
- « `''` »
- « `""` »
- « `'0'` »
- « `"0"` »
- « `0` »

Le reste est donc VRAI (TRUE).

La structure « `if` » / « `else` » est identique à celle du langage C :

```
if ( expression-booléenne )
  {
    lignes ;
  }
```

```
if ( expression-booléenne )
  {
    lignes ;
  }
else
  {
    lignes ;
  }
```


Attention : les accolades sont obligatoires même pour une seule ligne !

Langage C :

```
if ( expression-booléenne )
  ligne ;
```

Langage PERL :

```
if ( expression-booléenne )
  {
    ligne ;
  }
```

Autre syntaxe possible :

```
commande if ( expression-booléenne ) ;
```

Par exemple :

```
printf("%s\n", $nom) if ( $nom ne "" ) ;
```

équivalent à :

```
if ( $nom ne "" )
  {
    printf("%s\n", $nom) ;
  }
```


Comme en langage C.

Tant que l'expression est vraie alors on exécute le code :

```
while ( expression-booléenne )
  {
    lignes;
  }
```

Par exemple :

```
$i = 0 ;
while ( $i < 10 )
  {
    $i++ ;
  }
```

Comme en langage C.

Jusqu'à ce que l'expression soit vraie, on exécute le code :

```
until ( expression-booléenne )
  {
    lignes;
  }
```

Par exemple :

```
$i = 0 ;
until ( $i == 10 )
  {
    $i++ ;
  }
```


Très proche du langage C. On initialise une variable, on l'incrémente et on teste la valeur de la variable à chaque itération :

```
for ( initialisation ; test ; increment )
  {
    lignes;
  }
```

Par exemple :

```
for( $i = 0 ; $i < 10 ; $i++ )
  {
    printf("%d\n", $i)  ;
  }
```

Prends une liste de données en entrée et en assigne une dans une variable à chaque itération jusqu'à ce que la liste soit vide :

```
foreach $var (@liste)
   {
     lignes;
   }
```

Par exemple :

```
@liste = ("un", "deux", "trois") ;
foreach $i (@liste)
  {
    printf("%s\n", $i)  ;
  }
```


« `next` » : fait passer au tour suivant de la boucle courante
« `last` » : fait sortir de la boucle courante

Langage C	PERL
« `break` »	« `last` »
« `continue` »	« `next` »

Opérateur	Pour nombres	Pour chaines
Egalité	« == »	« eq »
Pas d'égalité	« != »	« ne »
Plus petit que	« < »	« lt »
plus grand que	« > »	« gt »
Plus petit ou égal à	« <= »	« le »
Plus grand ou égal à	« >= »	« ge »

Opérateur ET logique : « `and` » ou « `&&` »

Opérateur OU logique : « `or` » ou « `||` »

Opérateur de négation : « `not` » ou « `!` »

- « `-r` » : teste si l'objet est readable
- « `-w` » : teste si l'objet est writable
- « `-x` » : teste si l'objet est executable
- « `-z` » : teste si le fichier existe
- « `-s` » : teste si le fichier existe et a une taille > 0
- « `-f` » : teste si l'objet est un fichier
- « `-d` » : teste si l'objet est un répertoire
- « `-l` » : teste si l'objet est un lien symbolique

```
if ( -f $objet )
  { ... }
```


PERL utilise des file descriptors comme en langage C.

Un file descriptor PERL s'écrit sous la forme d'une variable spéciale car elle n'est pas précédée du signe « `$` ».

File descriptors PERL prédéfinis :

- « `STDIN` »
- « `STDOUT` »
- « `STDERR` »

File descriptor nommé « STDIN ».

Pour récupérer une seule ligne dans une variable :

```
$ligne = <STDIN> ;
```

Pour récupérer toutes les lignes à la fois dans un tableau :

```
@lignes = <STDIN> ;
```


ATTENTION : la lecture de « STDIN » ramène aussi le caractère fin de ligne avec la ligne.

```
$ligne = <STDIN> ;
```

⇒ « $ligne » contiendra « ananas\n » (en reprenant l'écriture à la C de la fin de ligne)

Utiliser « chomp() » pour retirer le dernier « \n » :

```
$ligne = <STDIN> ;
chomp($ligne) ;
```

ou

```
chomp(@lignes = <STDIN>) ;
```

Pour traiter une par une les lignes lues sur STDIN :

```
while ( $ligne = <STDIN> )
  {
    # traitement de la ligne
    mafonction($ligne) ;
  }
```

On peut utiliser la variable **implicite** spéciale « $_ » prédéfinie :

```
while ( <STDIN> )
  {
    # traitement de la ligne
    mafonction($_) ;
  }
```


File descriptor nommé « STDOUT ».

Utilisé par défaut par la commande « printf() » comme en langage C.

```
$s = "ananas" ;
printf("%s\n", $s) ;

printf STDOUT ("%s\n", $s) ;
```

Pas de fonction « fprintf() » en PERL : uniquement « printf() » !

File descriptor nommé « STDERR ».

```
printf STDERR ("Erreur : blabla... !\n") ;
```

Pas de fonction « fprintf() » en PERL : uniquement « printf() » !

- Langage C :

  ```
  fprintf(stderr, "%s\n", erreur) ;
  ```

- PERL

  ```
  printf STDERR ("%s\n", $erreur) ;
  ```


Syntaxe ressemblant à celle du shell :

- Pour ouvrir en lecture le fichier « exemple.txt » en utilisant un file descriptor qui s'appellera « DATA » :

  ```
  open( DATA, "< exemple.txt" ) ;
  ```

Pour terminer la lecture du fichier :

```
close( DATA );
```

Syntaxes ressemblant à celles du shell :

- Pour ouvrir en écriture c'est-à-dire créer ou écraser le fichier « exemple.txt » en utilisant un file descriptor qui s'appellera « DATA » :

```
open( DATA, "> exemple.txt" ) ;
```

- Pour ouvrir en écriture et écrire à la fin du fichier « exemple.txt » en utilisant un file descriptor qui s'appellera « DATA » :

```
open( DATA, ">> exemple.txt" ) ;
```

Pour terminer la lecture du fichier :

```
close( DATA ) ;
```


Manière élégante de tester la réussite de l'appel à « open() » via la fonction « die() » :

```
open( DATA, "> exemple.txt" ) || die("erreur XYZ!\n") ;
```

Techniques générales de lecture dans un fichier :

1. Lire une seule ligne :

```
$ligne = <DATA> ;
```

2. Lire toutes les lignes en une passe :

```
@lignes = <DATA> ;
```

3. Traiter toutes les lignes une à une :

```
while ( $ligne = <DATA> )
  {
    # traitement de la ligne
    mafonction($ligne) ;
  }
```

```
while ( <DATA> )
  {
    # traitement de la ligne
    mafonction($_) ;
  }
```


ATTENTION : la lecture de « DATA » ramène aussi le caractère fin de ligne avec la ligne. « $ligne » contiendra « ananas\n » (en reprenant l'écriture à la C de la fin de ligne)

```
$ligne = <DATA> ;

chomp($ligne) ;
```

Pour écrire dans un fichier que l'on a ouvert via le file descriptor « DATA » :

```
open( DATA, "> exemple.txt" ) ;

printf DATA ("%s\n", $variable) ;

close(DATA) ;
```


Voir en langage C la commande « fprintf() ».

```
#include <stdio.h>

main()
  {
    FILE *fd;
    char s[128] = "bonjour" ;

    if ( (fd = fopen("data.txt", "w")) == NULL )
      {
        fprintf(stderr, "fopen(): failed\n");
        exit(1);
      }
    fprintf(fd, "%s\n", s) ;
    fclose(fd);
  }
```

- Récupérer du texte venant d'une commande UNIX :

```
open(DATA, "commande |") ;
```

- Passer du texte à une commande UNIX :

```
open(DATA, "| commande") ;
```


Pour supprimer un fichier :

```
unlink( "fichier.txt" ) ;
```

Dans l'exemple suivant, on renomme le fichier « trace.log » en « trace.log.old » :

```
rename( "trace.log", "trace.log.old" ) ;
```


Dans l'exemple suivant, on met en place le lien symbolique « data » sur le fichier « data.log » :

```
symlink( "data.log", "data" ) ;
```

Nota : Il est possible d'obtenir le lien pointé par un lien symbolique en utilisant la fonction « readlink() ».

```
chmod( 0744, "fichier.txt" );

chown( 456, "exemple.txt" ) ;

chgrp( 65600, "exemple.txt" ) ;
```



```
mkdir( "/tmp/jardin", 0744 ) ;

rmdir( "/tmp/jardin" ) ;
```

PRE = PERL Regular Expressions :

- Très complètes
- Très puissantes
- Très complexes si besoin
- Grande réputation

Documentations :

- « `man perlre` »
- « `man perlrequick` »
- « `man perlretut` »

Il existe des librairies de programmation C fournissant ces expressions régulières de PERL : PCRE (PERL Compatible Regular Expressions).
Cf http ://www.pcre.org

◇ Caractères

- « `a` » : le caractère indiqué « `a` » ; etc.
- « `.` » : un caractère quelconque sauf retour charriot

◇ Classes de caractères

- « `[...]` » : liste des possibilités pour une position
- « `[^...]` » : liste des caractères interdits pour une position

◇ Classes prédéfinies

- « `\d` » : chiffre = « `[0-9]` » (« `\D` » : opposé)
- « `\w` » : caractère alphanumérique = « `[A-Za-z0-9_]` » (« `\W` » : opposé)
- « `\s` » : espace = « `[\f\t\n\r ]` » (« `\S` » : opposé)

◇ Quantificateurs

- « `*` » : 0 ou plus fois le motif précédent
- « `+` » : 1 ou plus fois le motif précédent
- « `?` » : le motif précédent est facultatif
- « `{m,n}` » : répétition du motif précédent entre m fois au minimum et n fois au maximum

◇ Positions

- « `^` » : en début de ligne
- « `$` » : en fin de ligne
- « `\b` » : word boundary : en extrémité de mot
- « `\B` » : non word boundary : en milieu de mot
- « `\A` » : en début de mot
- « `\Z` » : en fin de mot

◇ Groupes / Backreferences

- « `(...)` » : regroupement du motif
- « `\1` » : rappel du motif mémorisé numéro 1
- « `\17` » : rappel du motif mémorisé numéro 17 (n'en abusez pas)

Utilisation de backreference possible au niveau de la regexp.

Par exemple :

```
<img src=(['"]).*\1>
```

désigne ces deux possibilités :

```
<img src='image.jpg'>
<img src="image.jpg">
```

Une fois une regexp avec des groupes utilisés, on peut rappeler les groupes mémorisés via « `$1` », « `$2` », etc.

◇ Précédence

1. regroupement « `(...)` »
2. quantificateurs
3. ancrages « `^` », « `$` », « `\b` », etc. + séquence
4. caractère « `|` »

```
ananas|cerise
```


◇ Opérateurs PERL sur les RE

- « `m/RE/modifiers` » : match opérateur (« `m` » facultatif)
- « `s/RE/PATTERN/modifiers` » : substitute opérateur
- On peut changer les séparateurs :

`m/RE/`	`s/RE/PATTERN/`
`m(RE)`	`s(RE)(PATTERN)`
`m<RE>`	`s<RE><PATTERN>`
`m{RE}`	`s{RE}{PATTERN}`

- modifier « `i` » : lettres minuscules ou majuscules
- modifier « `g` » : substitution globale
- « `=~` » : binding opérateur

```
if ( /ananas/ )
  { ... }
```

```
if ( $fruit =~ /ananas/ )
  { ... }
```

◇ Variables spéciales

- « `$`` » : partie à gauche de ce qui matche
- « `$&` » : ce qui matche
- « `$'` » : partie à droite de ce qui matche

◇ Interpolation

« `/RE/` » : les substitutions y sont possibles comme pour une chaine entre guillemets

```
$a = "toto" ;
if ( /^($a)/ )
  { ...}
```


Pour découper vos chaines de caractères suivant une expression régulière :
Syntaxe : « `@liste = split(/RE/, $chaine) ;` »

Par exemple pour lire « `/etc/passwd` » :

```
open(PWD, "< /etc/passwd") ;
while( <PWD> )
  {
    chomp() ;
    @champs = split(/:/, $_) ;
    printf("LOGIN = %s\n", $champs[0]) ;
  }
close(PWD) ;
```

Pour joindre vos chaines de caractères suivant une expression régulière :
Syntaxe : « `$chaine = join($delimiteur, @liste) ;` »

Par exemple pour générer une ligne pour « `/etc/passwd` » :

```
$chaine = join(":", @liste) ;
```


Contrairement au langage C, on n'indique pas dans la déclaration de la fonction la définition de ses paramètres :

```
sub nom_fonction
  {
    ligne1 ;
    ligne2 ;
    ...
  }
```

Merci d'indenter les lignes comme en langage C.

Pour invoquer une fonction (simplification) :

```
mafonction(12,13) ;

mafonction() ;
```


La fonction récupère ses paramètres via la liste « `@_` » :

```
sub nom_fonction
  {
    my $a ;
    my $b ;

    ($a, $b) = @_ ;

    ...
  }
```

Pour déclarer une variable locale dans une fonction Perl, vous devez utiliser la fonction « my ».

```
sub mafonction
  {
    my $a ;
    ...
  }
```

La durée de vie de votre variable sera ainsi limitée à l'appel de la fonction.

Utiliser la commande « return ».

On peut retourner n'importe quel type de données :

```
sub mafonction
  {
    ...
    return($a) ;
  }
```

```
sub mafonction
  {
    ...
    return(@a) ;
  }
```

```
sub mafonction
  {
    ...
    return(%a) ;
  }
```

Le module « `Spreadsheet::WriteExcel` » permet de créer des fichiers MICROSOFT EXCEL directement au format natif XLS

Les cellules peuvent contenir des données comme des formules.

Les graphiques sont supportés en partie.

Les modules « `Text::CSV` » et « `Text::CSV::Simple` » permettent de manipuler des fichiers de type CSV générés par MICROSOFT EXCEL, OPEN OFFICE ou toute autre application.

Le module « `Config::IniFiles` » permet de manipuler des fichiers au format INI du monde MICROSOFT WINDOWS 95/98 (mais très pratique).

```
[section1]
variable1 = valeur
variable2 = valeur
variable3 = valeur
...

[section2]
variable1 = valeur
...
```


Le module « `DBI` » permet d'interagir avec des bases de données via des fonctions indépendantes du type de la base de données.

Les modules « `DBD::Mysql` », « `DBD ::Pg` », etc. implémentent les détails particuliers pour les bases de données MYSQL, POSTGRESQL, etc.

Le module « `Template` » permet de générer des fichiers remplis par des lots de données à partir de fichiers squelettes.

Le module « `Data::Dumper` » permet d'afficher des structures de données complexes de façon simple :

On fait l'hypothèse que le module n'a pas de dépendance. Sinon il faudra installer les dépendances au préalable.

Ensuite :

1. Récupérer le module sur CPAN
2. Désarchiver le module par « `tar xvzf module.tar.gz` »
3. « `cd module` »
4. « `perl Makefile.PL` »
5. « `make` »
6. « `make install` »