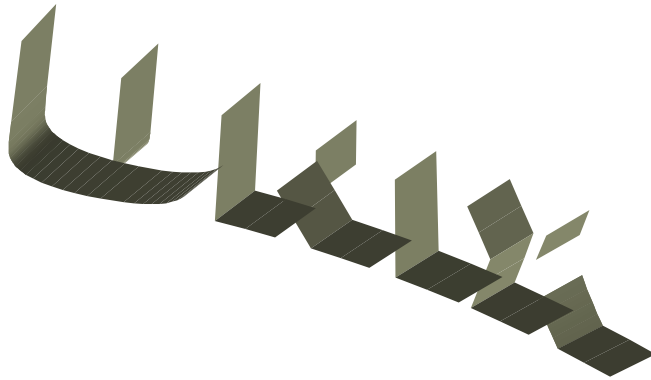


Introduction to the UNIX Operating System

**Joe VanDyke
Kevin Shinpaugh**



**Virginia Tech
Computing Center
Systems Programming**

Table of Contents

INTRODUCTION TO THE UNIX OPERATING SYSTEM.....	4
WHY UNIX ?.....	4
VARIETIES OF UNIX:.....	5
WHAT CAN UNIX SYSTEMS BE USED FOR ?	5
LOGGING INTO THE SYSTEM	6
PASSWORDS	6
USERID PERMISSIONS	6
GETTING HELP	7
SHELLS	14
ENVIRONMENT VARIABLES.....	15
TERMINAL CONTROL.....	16
CONTROL-KEYS.....	17
NAVIGATION AND DIRECTORY STRUCTURE.....	18
FILE TYPES AND PERMISSIONS	21
CHANGING FILE PERMISSIONS.....	22
FILE/DIRECTORY MANIPULATION.....	23
FILE COMMAND SYNTAX:.....	24
INFORMATION COMMANDS.....	26
USEFUL COMMANDS.....	27
SYSTEM PROCESSES AND CONTROL	28
PRINTING COMMANDS.....	30
PIPES AND REDIRECTION	31
COMMUNICATIONS COMMANDS	32
MESSAGING COMMANDS	32
INFORMATION COMMANDS.....	33
INTERNET/INTRANET COMMANDS	34
SECURE COMMUNICATIONS.....	34
X-WINDOWS OVERVIEW	35
WINDOWS	35

CDE – COMMON DESKTOP ENVIRONMENT 36
SUMMARY OF INSERT MODE COMMANDS 41
SELECTED COMMAND MODE COMMANDS..... 41
MOVEMENT COMMANDS 42
UNIX COMPARED TO DOS 43
INSERT MODE COMMANDSERROR! BOOKMARK NOT DEFINED.
SHORT LIST OF MAIL COMMANDS 44
ELM COMMANDS 44
REFERENCES 45

Introduction to the Unix Operating System

The UNIX system was started in the late 1960's. The authors of UNIX developed the operating system after AT&T- Bell Labs left the Multics project. The early seventies saw several changes, including being mostly written in C by 1973. AT&T was prohibited from selling the product, but they began to license it to universities and other corporations. The total number of sites running UNIX was about 500 by 1977. University of California, Berkley, rewrote the original system and released BSD UNIX in the late seventies. The UNIX system continues to grow, and by the mid-eighties, over 100,000 sites were running some derivative. AT&T sold UNIX in 1993 to Novell. Novell in turn gave the name UNIX to the X/OPEN consortium. Santa Cruz Operation bought the remaining product, UNIXWARE from Novell. Today, over 3,000,000 sites are running one form or another of UNIX. Virginia Tech's involvement with UNIX can be traced to the mid-1980's when the Computer Science department required incoming students to own a machine running UNIX. The first production UNIX machines at the Virginia Tech Computing Center appeared around 1990.

Why UNIX ?

The UNIX operating system offers a number of mainframe-like features, such as multi-users, multitasking, and batch job control. UNIX runs on a variety of hardware configurations – from PC to mainframe size. This flexibility in hardware is one of the many strengths of UNIX. The ability to get good, or great, performance out of relatively inexpensive hardware is one of the biggest selling points with Universities and businesses looking to cut the expense of mainframes. UNIX systems run many of the same programs as mainframes, and can be tailored to look almost exactly like the systems they are replacing. UNIX systems are also very reliable in comparison to solutions like Windows NT, which make them prime candidates for mission critical databases and number crunching.

The advantages of UNIX can be summed up as follows:

- Well designed operating system.
- Stability
- Speed
- Multi-user support
- Relatively easy to use.
- Easily customized.

Varieties of UNIX:

SCO/UNIXWARE – the current name of the original AT&T UNIX. (System V)

Solaris 2 – Sun's System V version.

SunOS – BSD based Sun system. (Obsolete)

AIX – IBM's version (SysV and BSD derived)

HP/UX – Hewlett-Packard

IRIX – Silicon Graphics

Digital UNIX – Digital Equipment Corporation

Ultrix – Digital's previous UNIX (obsolete)

LINUX – a freeware (mostly) UNIX downloadable from the net. Several varieties exist.

FreeBSD – a freeware implementation of BSD Unix. (Also **OpenBSD**, **NetBSD**)

HURD – a new os, much like LINUX in that it's free, from the GNU consortium – all commands are written and maintained by Internet users.

How can UNIX systems be used?

- Networking – telnet, ftp and several other services are standard in UNIX.
- Servers – application servers, mail servers, database servers, and web servers all run under UNIX.
- Graphics – most special effects in movies are done with SGI workstations.
- Number crunching.
- Programming
- Office productivity (Word Perfect, spreadsheets)
- Web development.
- Business applications (centralized ordering, payroll, etc.)

Logging into the System

The UNIX system requires a user to enter a user name and password to use the machine. While there are several possible ways to accomplish this, the most general ways are via a telnet session, or on the console of the machine. After checking the username/ password against it's master file, the system will grant you access. The process of logging in is also written to several system log files. These log files enable system managers to track usage. They also serve as evidence when the systems are broken into.

Passwords

It is very important to select a good password for your UNIX login id. Good passwords are ones that are:

- Not easily guessed (nothing personal, no dictionary words, no birthdays or social security numbers)
- Contain mixed case and numbers
- 6 to 8 characters

Passwords are changed using the **passwd** command. Generally, it is a good practice to change your password every 60-90 days, and to never reuse the same password twice.

passwd	Change user's password	<pre>passwd mrfixit Changing password for "mrfixit" mrfixit's Old password: mrfixit's New password: Enter the new password again:</pre>
---------------	------------------------	---

Userid Permissions

The UNIX system assigns privileges to users based on userid and group memberships. Userid privileges are ones granted to your own personal files, or can be used to determine if you have any privileges to a file or directory at all. Group privileges can be used to allow several people to have the same permissions on files or directories. Groups can be setup for any criteria, but are usually set to mirror administrative groups or special projects. Group permissions can let everyone that belongs to a specific group share data and programs, while keeping other users from seeing or using them. You can belong to more than one group at any one time. The **groups** command will give you a list of which groups you belong to.

group	List user's groups	<pre>groups spd system</pre>
--------------	--------------------	------------------------------

Getting Help

There are several ways to get help under the UNIX system. The most basic, and common, is the **man** system. This is basically an online manual system for commands. In addition, some systems also have the GNU **info** system, which is very similar. Many UNIX vendors supplement the man pages with their own system. IBM has a proprietary version of **info**, which is available either locally or as a web based help system. Sun Microsystems provides **AnswerBook**, which is available either as a standalone app, or via the web. Compaq's Tru64 **Bookreader** help system is being converted to be web based. Silicon Graphics provides help documentation via **insight**.

The **man** command can be used in two ways:

If you know the command you are looking for:

man command

If you have a general subject area:

man -k subject

If your **man** page is in a non-standard directory:

man -M *pathname*

The output of the man command looks like this:

```
man csh
Reformatting page.  Wait... done

User Commands                                csh(1)

NAME
    csh - shell command interpreter with a C-like syntax

SYNOPSIS
    csh [ -bcefinstvVxX ] [ argument... ]

DESCRIPTION
    csh, the C shell, is a command interpreter with syntax
    reminiscent of the C language.  It provides a number of con-
    venient features for interactive use that are not available
    with the Bourne shell, including filename completion, com-
    mand aliasing, history substitution, job control, and a
    number of built-in commands.  As with the Bourne shell, the
    C shell provides variable, command and filename substitu-
    tion.

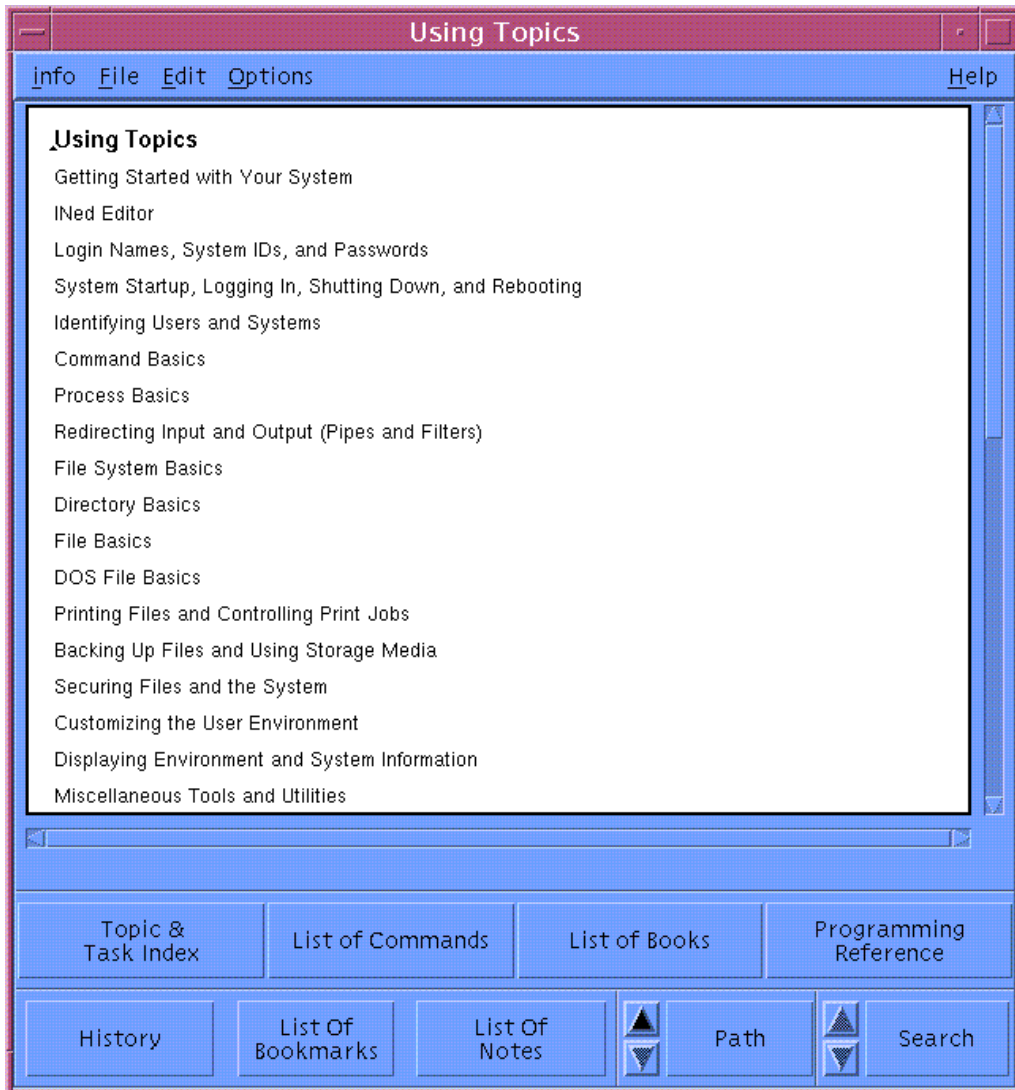
Initialization and Termination
    When first started, the C shell normally performs commands
    from the .cshrc file in your home directory, provided that
--More--(1%)
```

The man system uses the more command to display the text one page at a time.

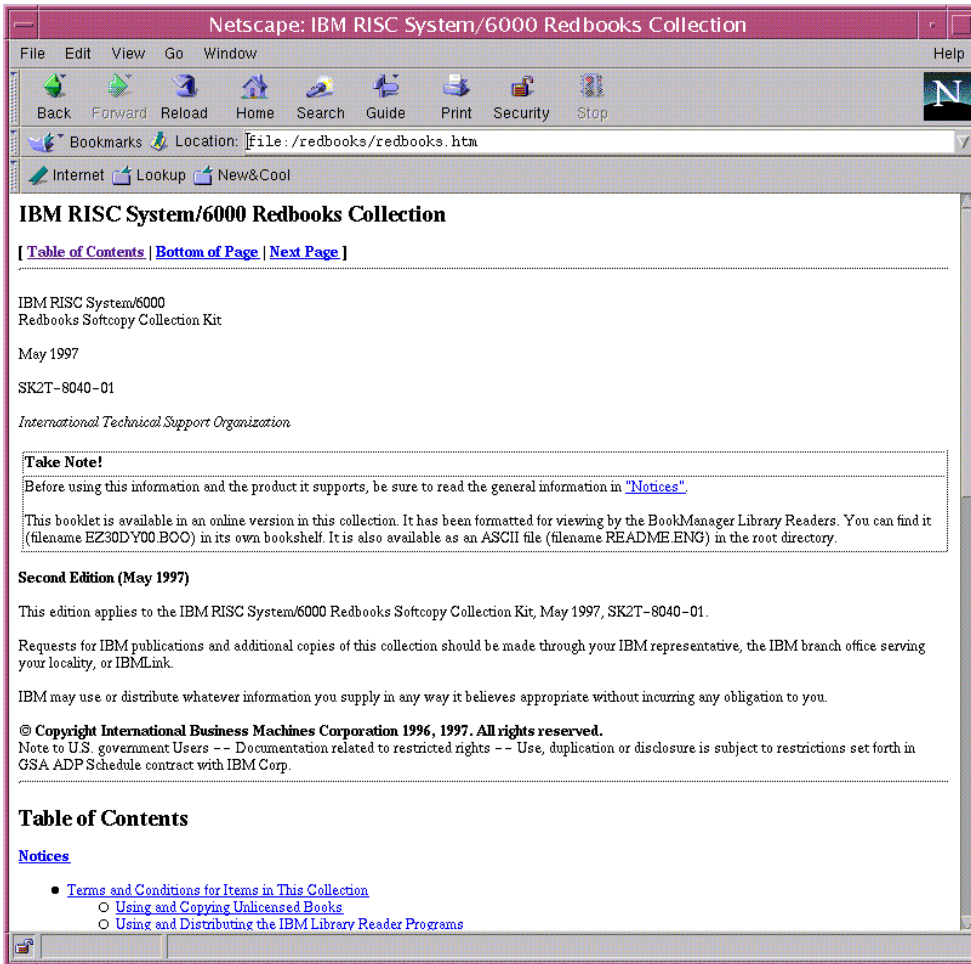
Vendor Help Systems

Most of the vendor help systems are available through the X-windows system. These systems are as follows:

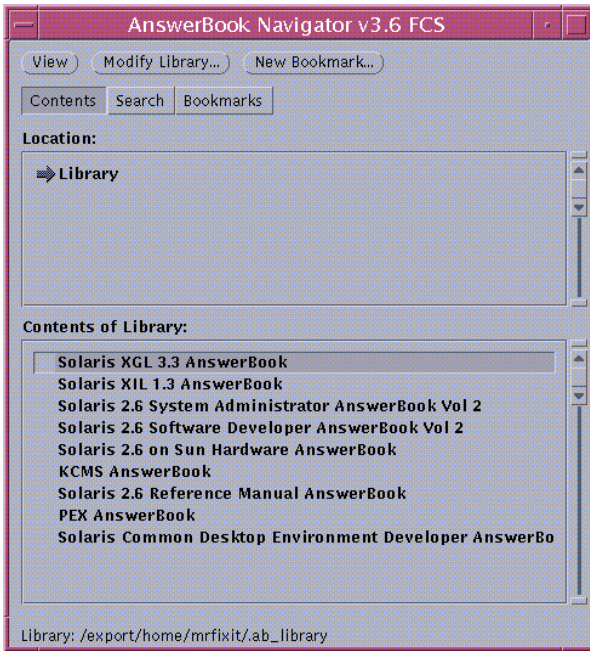
AIX	InfoExplorer, info, redbooks (html based)	info file://redbooks/redbooks.html
Sun Solaris	AnswerBook	ab2 http://docs.sun.com
Digital/Compaq	bookreader	http://dock.cc.vt.edu/DigitalUnix
SGI	insight	insight



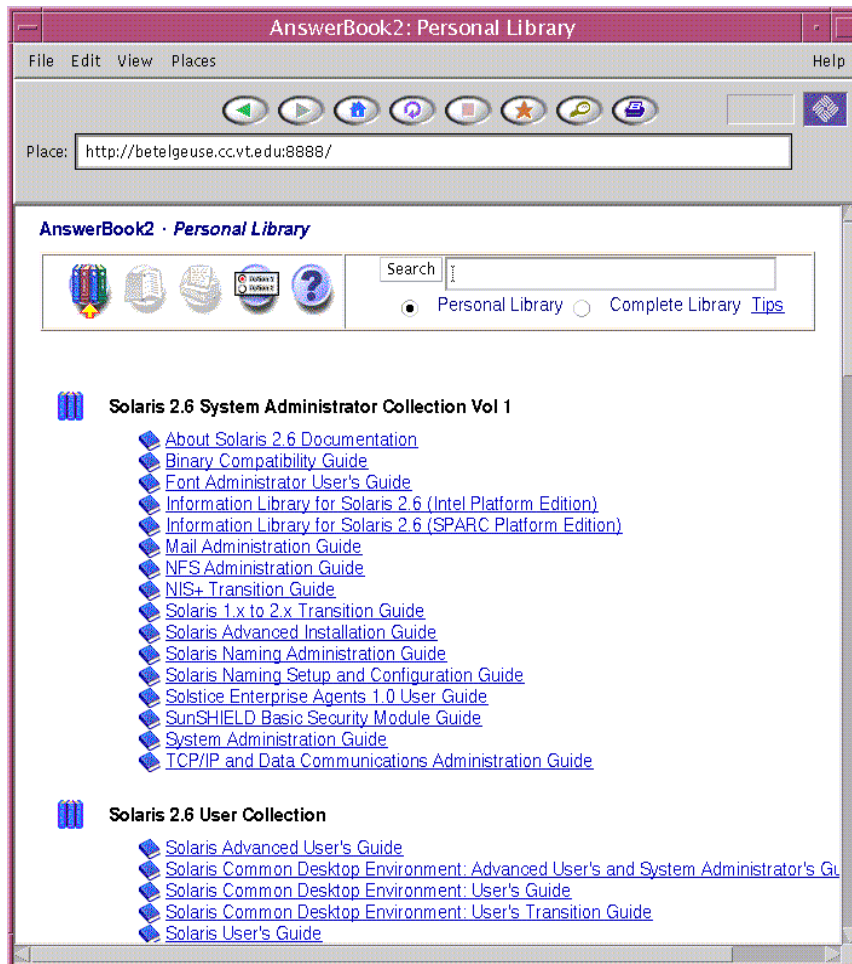
IBM AIX Info System Screen



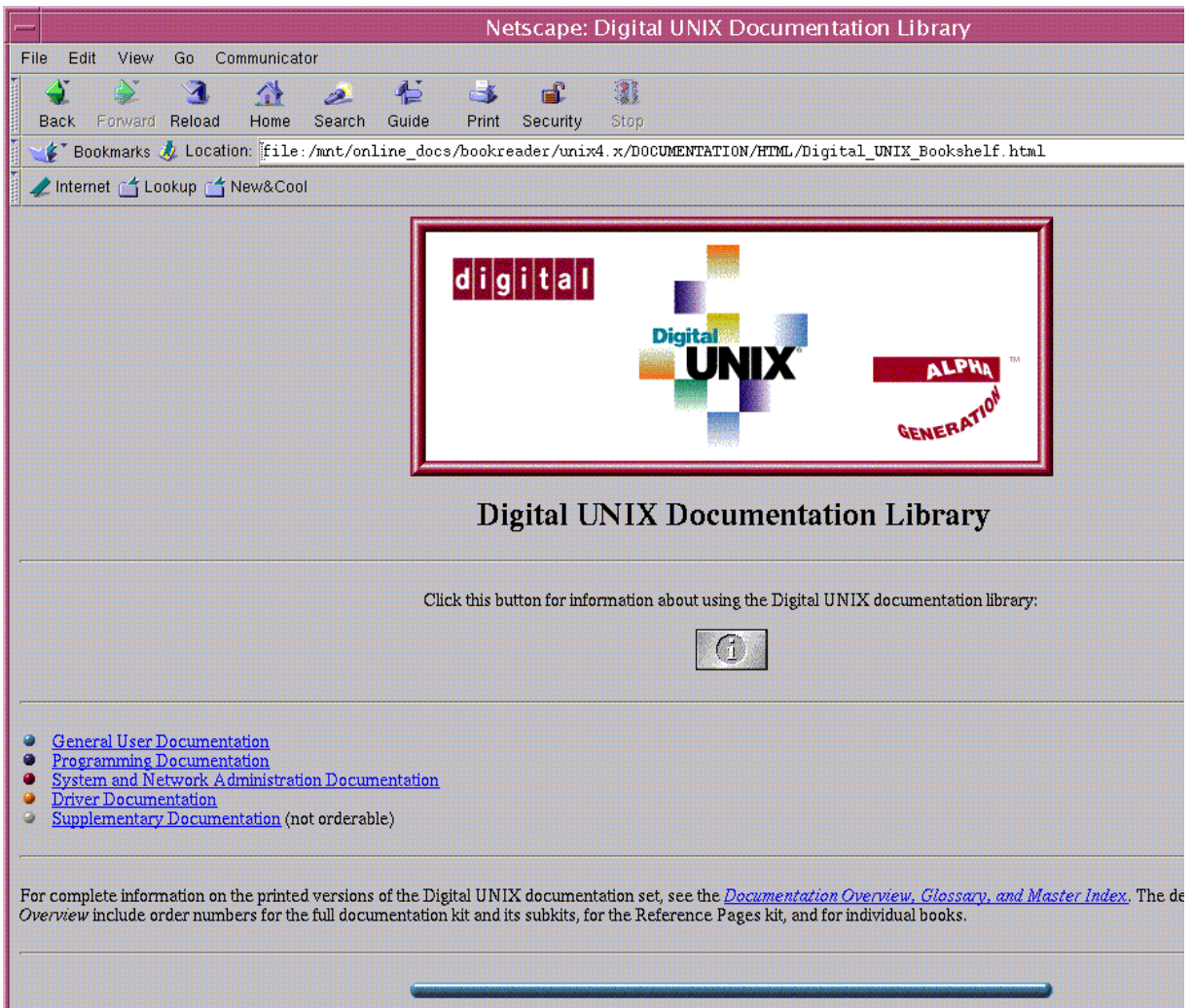
IBM Redbooks Web Page



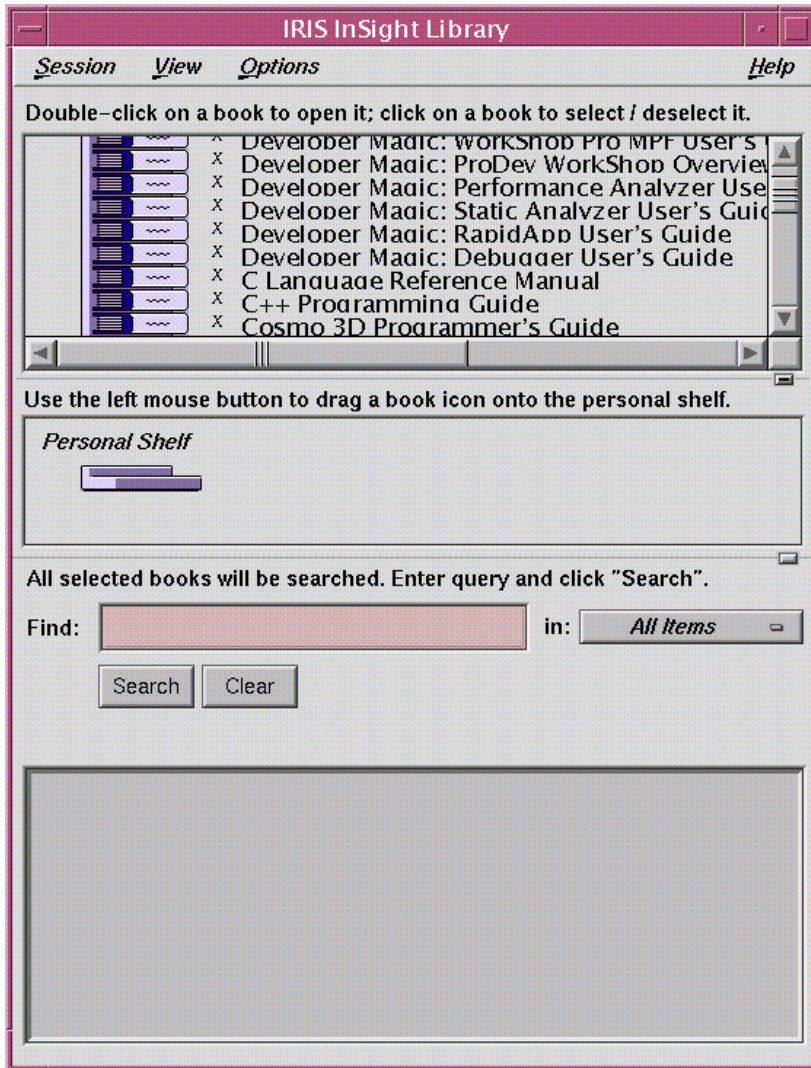
Sun Answerbook – Pre Solaris 2.6



Sun Answerbook2 – a HTML based help system



Digital Unix Bookreader information system – HTML based



Silicon Graphics IRIX Insight Documentation System

Shells

The first program that you will execute after logging into a UNIX machine is a shell. Many different shells are found on UNIX systems. The most popular are **sh**, **bash**, **csh**, **tcsh**, and **ksh**. Most users are assigned **csh** or **ksh** as their default shell. **bash** and **tcsh** are relatively new shells, and are not available on all systems yet.

Shells can be used as the login shells, or as execution programs for scripts. Scripting is one of the most powerful features of shells, as they can be used like a programming language to do many repetitive tasks or serve as front-end programs for a number of different programs. System administrators often use shell scripts to create menu systems for operators and users. Shells used as login programs can be configured in many different ways. Login shells read configuration files that exist in the user's home directory. These configuration files are special files that begin with a .

sh, bash, ksh	.profile
csh	.login, .cshrc
tcsh	.login, .cshrc, .tcshrc

Shells have different capabilities, while sharing some similarities. While **csh** and **tcsh** are the most complex in terms of programmability, they are generally not used when administration scripts are used because they have poor flow control and redirection. **csh** and its descendant **tcsh** share many common features, such as job control, command history and rich shell programming features. However, **tcsh** also adds command completion to the mix and command editing. **sh**, **bash**, and **ksh** all have the same base. **sh** is the original shell, the Bourne shell, named for one of the creators of UNIX. **ksh** is an extension to **sh** that adds better job control and command completion. **bash** is another rewrite of **sh**, that adds many of the features of **ksh**, and is known as the "Bourne Again Shell". **bash** is the standard shell for Linux based systems, and is gaining popularity as a total replacement for **sh**.

Environment Variables

Shells use environment variables to control and enhance the workspace. These variables, which can be set either in the startup script or on the fly, are customizable by the user at any time.

A few environment variables that you are commonly used are:

DISPLAY	Graphical display to use
EDITOR	Your default editor
GROUP	login group
HOME	Path to your home directory
HOST	Hostname of your system
LOGNAME	Your login user name
PATH	Paths to be searched for commands
SHELL	login shell
TERM	Your terminal type
USER	Your username
NOCLOBBER	Prevents you from accidentally overwriting files

Terminal control

The **stty** command can be used to set a variety of terminal functions. Its most common use is to set the backspace key, which varies from keyboard to keyboard. You can also use the **stty** command to reset a terminal that is locked up.

Command	Result	Example
stty	Reports on current terminal settings	<pre>(mrfixit@betelgeuse) ~> stty speed 38400 baud; -parity rows = 24; columns = 80; ypixels = 0; xpixels = 0; erase = ^h; swtch = <undef>; brkint -inpck -istrip icrnl -ixany imaxbel onlcr tab3 echo echoe echok echoctl echoke iexten</pre>
stty -a	Reports on all options	<pre>(mrfixit@betelgeuse) ~> stty -a speed 38400 baud; rows = 24; columns = 80; ypixels = 0; xpixels = 0; eucw 1:0:0:0, scrw 1:0:0:0 intr = ^c; quit = ^\; erase = ^h; 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 -crtsets -crtsexoff -pare xt -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 tab3</pre>
stty kill <i>character</i>	Set the line kill character	<pre>(mrfixit@betelgeuse) ~> stty kill ^x</pre>
stty erase <i>character</i>	Set the erase/backspace character	<pre>(mrfixit@betelgeuse) ~> stty erase ^h</pre>
stty intr <i>character</i>	Set the interrupt character	<pre>(mrfixit@betelgeuse) ~> stty intr ^c</pre>
stty sane	Reset the terminal	<pre>(mrfixit@betelgeuse) ~> stty sane</pre>

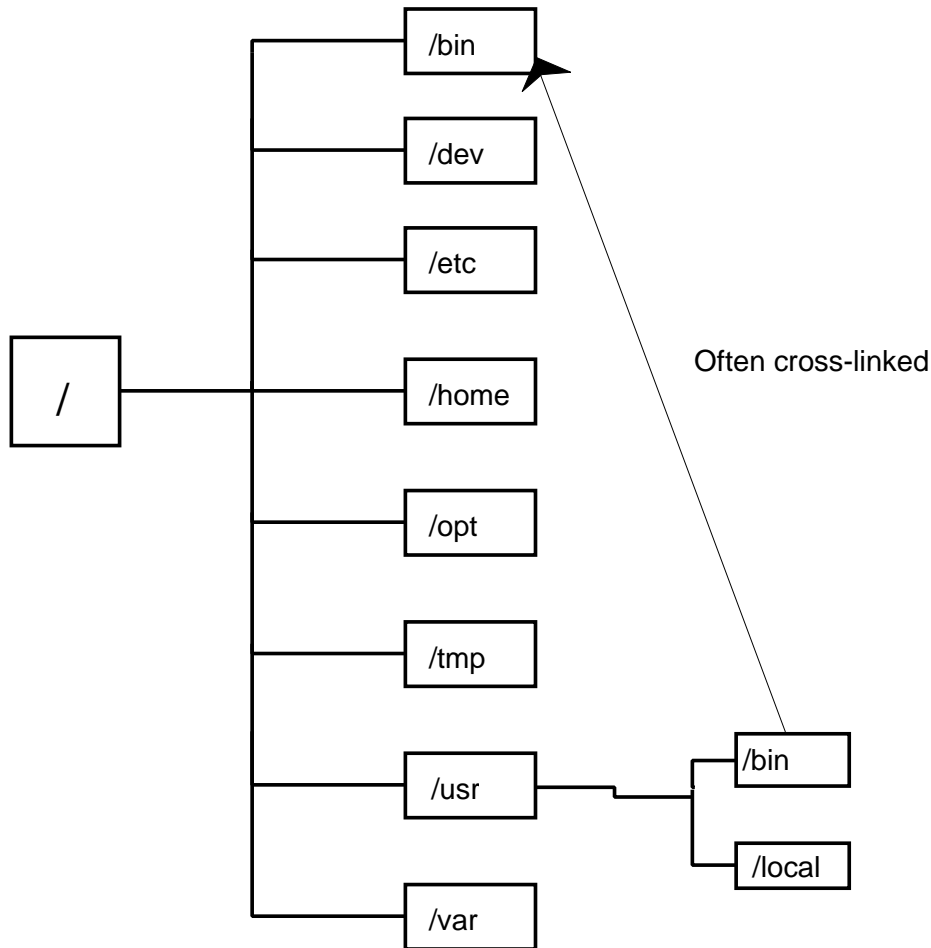
Control-keys

Several key combinations can affect the operation of a UNIX system. These control keys can stop execution of a command, suspend execution of a command, log-off the system, pause out put, or cause the terminal to beep. The most useful control sequences are those that affect jobs. These sequences require you to hold down the control key while pressing the appropriate meta-key.

Control Sequence	Result
control-c	Terminate current running job
control-d	Exit system. Exit some commands.
control-z	Suspend operation
control-s	Stop scrolling (halt output)
control-q	Restart scrolling (restart output)

Navigation and Directory Structure

The UNIX system uses a tree structure for the file system. You start at the bottom, the root (/), and work your way out the branches.



Important Directory	Location	Contains
bin	/bin,/usr/bin	Many UNIX commands (ls, csh, sh, login)
etc	/etc	password database, network configuration files, startup files
local	/usr/local	Locally added products (elm, netscape)
ucb	/usr/ucb	BSD legacy commands
opt	/opt	Many systems store optional packages here (compilers, netscape)
var	/var	System logs, mail and printer spools

The process of traversing the limbs and branches is accomplished using the change directory, or `cd` command. The system also has shortcuts to get from place to place. Some of these shortcuts are:

Shortcut	Symbol
.	Current directory
..	One level up
~	Home directory
*	Can be used for completion of a directory name.

Example: Change to the home directory of joeuser.

```
cd ~joeuser
```

Other important navigation commands:

pwd – present working directory

ls – list directory

The **ls** command, like many UNIX commands accepts many arguments to get more than the standard amount of information. These arguments can be combined to give more functionality to the command.

ls -a	Lists hidden files	ls -acshrc .login
ls -l	Gives more information	ls -l total 25206 -rw-r--r-- 1 zeus staff 1097547 Jul 29 13:50 1617.pdf -rw-r--r-- 1 zeus staff 23787 Jul 29 13:50 card.gif -rw-r--r-- 1 zeus staff 2432 Jul 29 13:50 dot.tcsh -rw-r--r-- 1 zeus staff 237037 Jul 29 13:50 final report 2.hqx -r-xr-xr-x 1 zeus staff 11516772 Jul 29 13:50 netscape -rw-r--r-- 1 zeus staff 0 Jul 29 13:50 test.html
ls -al	Lists hidden files and more information	ls -al total 25214 drwx--x--x 2 zeus staff 512 Jul 29 13:50 . drwxr-xr-x 11 root root 512 Jul 29 13:47 .. -rw----- 1 zeus staff 124 Jul 29 13:47 .cshrc -rw----- 1 zeus staff 575 Jul 29 13:47 .login -rw-r--r-- 1 zeus staff 1097547 Jul 29 13:50 1617.pdf -rw-r--r-- 1 zeus staff 23787 Jul 29 13:50 card.gif -rw-r--r-- 1 zeus staff 2432 Jul 29 13:50 dot.tcsh -rw-r--r-- 1 zeus staff 237037 Jul 29 13:50 final report 2.hqx -r-xr-xr-x 1 zeus staff 11516772 Jul 29 13:50 netscape -rw-r--r-- 1 zeus staff 0 Jul 29 13:50 test.html
ls -l	List in a column	ls -l 1617.pdf card.gif dot.tcsh final report 2.hqx netscape test.html

File Types and Permissions

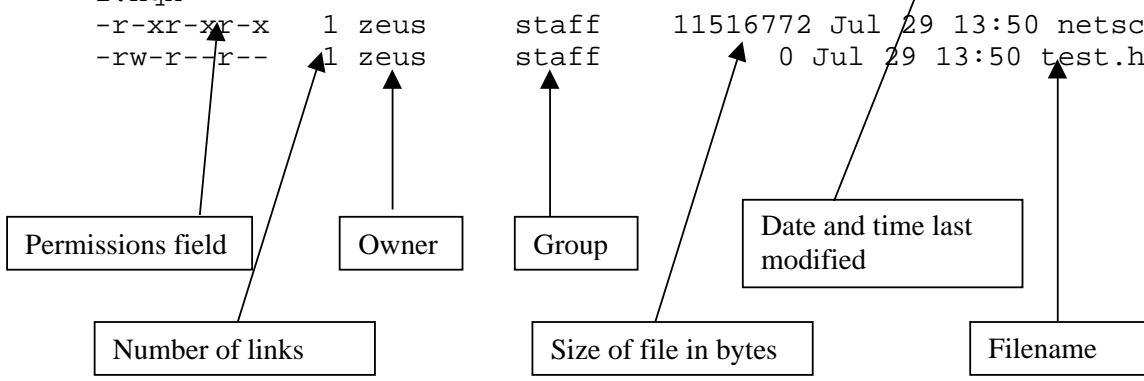
The UNIX system has a very good security system on files. You can see what permissions exist on a command by using the `ls -l` command.

First Character in Field		Next 3 sets of 3 Characters	
Character	Entry is a:	Character	Permissions
d	Directory	r	Read permission
-	Plain file	w	Write permission
b	Block device	x	Execute permission
c	character-type special file	s	Setuid execution
l	symbolic link	S	Setuid no-execute
s	socket	-	No permission
		t	Sticky bit

Setuid refers to the ability of a program to run as the super-user. This is set on programs that need extra permissions, such as daemons.

The `ls-l` command also gives information about the userid, group, and size of the files or directories being examined.

```
ls -l
total 25206
-rw-r--r--  1 zeus   staff   1097547 Jul 29 13:50 1617.pdf
-rw-r--r--  1 zeus   staff    23787 Jul 29 13:50 card.gif
-rw-r--r--  1 zeus   staff    2432 Jul 29 13:50 dot.tcsh
-rw-r--r--  1 zeus   staff   237037 Jul 29 13:50 final report
2.hqx
-r-xr-xr-x  1 zeus   staff   11516772 Jul 29 13:50 netscape
-rw-r--r--  1 zeus   staff    0 Jul 29 13:50 test.html
```



Changing file permissions

File permissions can be modified using the **chmod**, **chown**, and **chgrp** commands.

The **chmod** command changes the permissions on the file. **chown** and **chgrp** change ownership permissions.

The **chmod** command can be used in two ways: with an octal number or with a symbol. The symbolic approach is much easier to remember.

The symbolic method requires you to know two groups of symbols in order to change permissions. The first is the class of user. The second is the permission you are seeking to give the class. It also requires you to use an operator to grant or deny the privilege.

Group	Meaning
u	User/owner of the file
g	Group of the owner
o	All other system users
a	Can be used to represent ALL groups.

Letter	Permission granted
r	Read Permission
w	write permission
x	execute permission
s	Execute the program as the owner
t	Sticky bit (allows for quicker execution of program) CAN ONLY BE SET BY ROOT and only applies to the User/Owner grouping.

Example: To give everyone read permission on a file, *testfile*, the following **chmod** should be used:

```
chmod a+r testfile
```

Changing ownership:

Command	Syntax
chown	chown <i>userid filename</i> Example: chown oper test.sh
chgrp	chgrp <i>userid filename</i> Example: chgrp staff test.sh

File/Directory Manipulation

Moving around in the system is only part of getting around in the system. To be productive, you probably also will want to manipulate files and directories.

Directory commands:

mkdir - Create a directory.

Syntax - *mkdir* directoryname

rmdir - Remove an **empty** directory.

Syntax - *rmdir* directoryname

To delete an entire directory and its contents (including subdirectories) use the **rm -r** command.

File commands:

touch - create a file (empty).

tail - look at the end of a file

head - look at the beginning of the file

cat - display file contents on the screen

more - display file contents one page at a time. Some systems replace this command with **less**.

cp - copy a file

mv - move a file from one location (or filename) to another.

rm - delete a file or directory

pg - displays file contents one page at a time. Adds the ability to page backward through the file.

File Command Syntax:

Command	Format	Example command
touch	touch filename	touch test
tail	tail -# filename , where # is the number of lines you want to see.	tail +4 dot.tcsh ###----->path<----- ###----->time<----- ###----->watch<-----
head	head -# filename , where # is the number of lines you want to see	head -4 dot.tcsh ### This file has been auto generated ### by the tcsh module for the Dotfile Generator ### on Mon Jun 1 16:19:10 EDT 1998
cat	cat filename	cat test.out PID TTY TIME CMD 16596 pts/4 0:00 tcsh 8501 ? 1210:36 Xsun 8565 ? 0:00 dsdm 8587 ? 0:06 dtfile 16593 ? 0:00 dtexec 8579 pts/2 0:47 ttsessio 8567 pts/2 0:00 tcsh 8618 ? 0:00 cat
more	more filename	more test.out PID TTY TIME CMD 16596 pts/4 0:00 tcsh 8501 ? 1210:36 Xsun 8565 ? 0:00 dsdm 8587 ? 0:06 dtfile 16593 ? 0:00 dtexec 8579 pts/2 0:47 ttsessio 8567 pts/2 0:00 tcsh 8618 ? 0:00 cat 11894 ? 0:00 dtfile 8591 ? 0:00 sdtvolch 16598 pts/4 0:00 man 8529 ? 0:00 fbconsol 8589 ? 0:01 perfmete 8564 pts/2 0:00 sdt_shel 16604 pts/4 0:00 sh 9938 ? 0:01 httpd 8586 ? 0:21 dtwm 8533 ? 0:00 speckey 8580 pts/2 0:15 dtsessio 8520 ? 0:00 Xsession 16605 pts/4 0:00 more --More-- (87%)
cp	cp filename1 filename2	cp test.out test.out2
mv	mv filename1 filename2	mv test.out2 test.out1
rm	rm filename to delete rm -r directory to remove a directory and its contents	rm test.out rm -r test

pg	pg filename	pg test.out PID TTY TIME CMD 16596 pts/4 0:00 tcsh 8501 ? 1210:36 Xsun 8565 ? 0:00 dsdm 8587 ? 0:06 dtfile 16593 ? 0:00 dtexec 8579 pts/2 0:47 ttsessio 8567 pts/2 0:00 tcsh 8618 ? 0:00 cat 11894 ? 0:00 dtfile 8591 ? 0:00 sdtvolch 16598 pts/4 0:00 man 8529 ? 0:00 fbconsol 8589 ? 0:01 perfmete 8564 pts/2 0:00 sdt_shel 16604 pts/4 0:00 sh 9938 ? 0:01 httpd 8586 ? 0:21 dtwm 8533 ? 0:00 speckey 8580 pts/2 0:15 dtsessio 8520 ? 0:00 Xsession 16605 pts/4 0:00 more 16594 ?? 0:00 dtterm :
rmdir	rmdir directory	rmdir test

Information Commands

The **df** and **du** commands give information on the amount of space used on the UNIX system, and the amount of space used by a directory, respectively. They can be useful to see how much space is left on a system (a prime cause of problems), or to see how much space your home directory is using.

Command	Syntax	Example
df	df	<pre>df / (/dev/dsk/c0t0d0s0): 193116 blocks 58779 files /usr (/dev/dsk/c0t0d0s6): 676344 blocks 319020 files /proc (/proc): 0 blocks 906 files /dev/fd (fd): 0 blocks 0 files /var (/dev/dsk/c0t0d0s4): 93322 blocks 58894 files /export/home (/dev/dsk/c0t0d0s7): 195066 blocks 290637 files /opt (/dev/dsk/c0t4d0s7): 800844 blocks 243651 files /usr/openwin (/dev/dsk/c0t0d0s3): 114422 blocks 123359 files /tmp (swap): 232608 blocks 9542 files</pre>
	df -k	<pre>df -k Filesystem kbytes used avail capacity Mounted on /dev/dsk/c0t0d0s0 123455 26897 84213 25% / /dev/dsk/c0t0d0s6 720199 382027 280557 58% /usr /proc 0 0 0 0% /proc /dev/fd 0 0 0 0% /dev/fd /dev/dsk/c0t0d0s4 123455 76794 34316 70% /var /dev/dsk/c0t0d0s7 600559 503026 37478 94% /export/home /dev/dsk/c0t4d0s7 1019183 618761 339272 65% /opt /dev/dsk/c0t0d0s3 259879 202668 31224 87% /usr/openwin swap 115664 400 115264 1% /tmp</pre>

du	du directory	<pre>du zeus 25214 zeus</pre>
	du -k directory	<pre>du -k zeus 12607 zeus</pre>

Useful Commands

A few commands that can be used to improve your proficiency with UNIX are **script**, **whereis**, **which**, and **date**.

Command	Syntax	Output
date	date	date Thu Jul 30 11:04:21 EDT 1998
script	script filename	Will create a file that contains a record of your session.
whereis	whereis filename	Searches your path for the command: whereis date date: /usr/bin/date /usr/man/man1/date.1
which	which filename	Will tell you which copy of a command that you are executing. which csh /bin/csh

whereis is a BSD based command and may not be available on all systems.

The **find** command can generate the same information. The **find** command has many arguments, but it can be used very simply.

find	find path -name searchstring -print	find /usr/bin -name csh -print /usr/bin/csh
-------------	--	--

System Processes and Control

Every command that is executed under the UNIX system is given a process ID number. This allows the user and the system to keep track of, modify, or end any command. A user can get information about processes via the **ps** command. This command without any arguments will give the user information about commands they have running. With arguments, the user can see all commands running on the system, or even those commands run by another user.

The following table shows three common invocations of the **ps** command: no arguments, show all processes in extended form, and show all processes for a specific user.

ps	<pre>ps PID TTY TIME CMD 14464 pts/4 0:00 csh</pre>
ps -eaf	<pre>ps -eaf UID PID PPID C STIME TTY TIME CMD root 0 0 0 Jun 11 ? 0:00 sched root 1 0 0 Jun 11 ? 0:35 /etc/init - root 2 0 0 Jun 11 ? 0:00 pageout root 3 0 0 Jun 11 ? 45:04 fsflush root 437 1 0 Jun 11 console 0:00 /usr/lib/saf/ttymon -g -h -p betelgeuse.cc.vt.edu console login: -T sun -d /de root 243 1 0 Jun 11 ? 0:30 /usr/sbin/syslogd -n - z 12 root 436 1 0 Jun 11 ? 0:00 /usr/lib/saf/sac -t 300 root 217 1 0 Jun 11 ? 0:06 /usr/sbin/inetd -s root 120 1 0 Jun 11 ? 0:03 /usr/lib/security/cryptorand /dev/random</pre>
ps -u username	<pre>ps -u mrfixit PID TTY TIME CMD 8501 ? 912:43 Xsun 8565 ? 0:00 dsdm 8587 ? 0:03 dtfile 8579 pts/2 0:36 ttsessio 8567 pts/2 0:00 tcsh 8618 ? 0:00 cat 11894 ? 0:00 dtfile 8591 ? 0:00 sdtvolch 8529 ? 0:00 fbconsol 8589 ? 0:01 perfmete 8564 pts/2 0:00 sdt_shel 9938 ? 0:01 httpd 8586 ? 0:17 dtwm 8533 ? 0:00 speckey</pre>

The **ps** command will give you the process id number, or **PID** of the command. You can use this **PID** to put jobs in the background, bring jobs out of the background, or stop the command.

kill PID	end a command.
kill -9 PID	Terminate job immediately
kill -HUP PID	Restart a job

Another important process control feature is immediate backgrounding when starting an application. Adding a **&** to the end of a command will place it in the background. This is useful when you know a program will run for a long time, or when you run multiple commands at the same time

The **ksh** adds a specific command to look at running jobs, the **jobs** command. Although the **jobs** command can be used in **cs**h to list running jobs, the **ksh** environment allows you to modify running jobs as well.

Printing Commands

Printing on a UNIX system can be a bit tricky, depending on whether the system used BSD or SYSV based printing. Some systems support both **lp** and **lpr**.

Command	Function	Example
lpr (lp) filename	Print a file	lpr -P bm3130 rep
lpq (lpstat)	Show the status of printed jobs	See below
lprm (cancel) filename	Remove a file from the print queue	lprm 2
pr filename	Print a file on the screen	See below

```

lpq
Queue   Dev   Status   Job Files   User   PP %   Blks   Cp Rnk
-----
ibm3130 @prin READY
ibm3130: logical      Prt/Job   JOB      Global      Job
Job      Total   Int
ibm3130: logical      Prt/Job   JOB      Global      Job
Job      Total   Int
ibm3130: logical      Prt/Job   JOB      Global      Job
Job

pr rep

Tue Jan 13 13:59:21 EST 1998 rep Page 1

Filesystem   1024-blocks   Free %Used   Iused %Iused   Mounted on
/dev/hd4      8192          1204 86%        1364 34% /
/dev/hd2      450560        4 100%       14318 13% /usr
/dev/hd9var   73728         10164 87%        422 3% /var
/dev/hd3      32768         31272 5%         86 2% /tmp
/dev/hd1      8192          6972 15%        971 48% /home
/dev/lv01     638976        286900 56%        20298 13% /usr/src
/dev/lv02     704512        15416 98%        19029 11% /usr/local
/dev/lv03     1781760       262352 86%        29546 7% /us2
/dev/fslv01   5373952       3816212 29%        51910 4% /us3
/dev/fslv03   98304         16548 84%        5796 24% /usr/local/matlab5.0
/dev/fslv04   425984        16568 97%        4955 5% /usr/local/sas612
/dev/fslv05   180224        38460 79%        1988 5% /usr/local/fn90mp
/dev/fslv06   16384         13764 16%        218 6% /usr/local/nsr
/dev/fslv08   409600        225144 46%        1795 2% /rs6000.inst.images
/dev/lv04     262144        30784 89%        29 1% /usr/src/matlab5.1
/dev/lv05     163840        9684 95%        3616 9% /usr/local/sas
/dev/lv06     147456        15388 90%        1550 5% /usr/local/vni
/dev/lv07     65536         16236 76%        3887 24% /usr/local/matlab42c
/dev/lv10     81920         24972 70%        32 1% /usr/src/matlab42c
/dev/lv00     1048576       184948 83%        30810 12% /us1
/dev/lv08     262144        71144 73%        3607 6% /usr/local/sas609_ts42b
Int
Total

```

Pipes and Redirection

A very powerful feature of UNIX is the ability to redirect, or pipe, command output from one command to another or into a file instead of the standard output (usually the terminal). This feature has many uses, such as sending a file as an email attachment or getting the output of a command inserted into a file. Pipes can also be used to sort a file, search a file for a specific pattern or even copy files from one location to another.

>	Redirect output from one file/program to another one. Create new file if it doesn't exist. Example: <code>date > todaysdate</code>
<	Redirect output from a file into another file/program. Example: <code>mail user@site < todaysdate</code>
>>	Same as the > redirect, except will append to an existing file. Example: <code>ls -l >> todaysdate</code>
	Pipe output from one program to another. Examples: <code>ps -e grep csh</code> - this command will execute a <code>ps</code> command and only output the processes using <code>csh</code> . <code>ls -al more</code> - this will list the directory, and send the output to the <code>more</code> program so you can page through the output one page at a time.

Redirections and pipes are a very powerful, and somewhat confusing, part of UNIX. However, even beginning users can use pipes to enhance their UNIX experience.

Other important redirections are the use of UNIX's `stdin`, `stdout`, and `stderr`. These special variables are normally set to the keyboard, screen, and terminal respectively. These variables can be changed, but generally, redirections are used instead of changing them.

Communications Commands

One of the most useful aspects of the UNIX operating system is the large number of communications tools that are offered in it. UNIX is not only the backbone of the Internet, but also the birthplace of many utilities and protocols that are in use today. UNIX communications commands make everyday administration of multiple systems easier with tools like NIS/NIS+, which allow multiple machines to share resources such as userids, user home directories, and many other resources. NFS, which is a way of sharing files across the net, can be used in conjunction with NIS or by itself. It is a simple way to share directories and files between UNIX machines and other types of machines, like PC's. Other commands, like **talk**, **write** and **wall** are used to communicate with users both on the same machine or with other machines connected to the network. UNIX supports all of the common utilities that you may have used before including telnet, ftp, WWW browsers (including Netscape) and email. A variety of programs have appeared over the years to process e-mail. Some of these are text-based and can be easily used on terminals. Many newer e-mail programs are X-windows based, and are more suitable for use on the console or a xterminal. In addition, UNIX systems can serve as Internet services providers, giving access to the web, email and even dial-in connections to the Internet. One recent survey placed UNIX as the operating system on over 60% of the servers on the Internet.

Messaging Commands

talk	talk user@machine example: talk oper@vtajx.cc.vt.edu
write	write user
wall	wall message

All messaging commands are terminated with control-c or control-d. **talk** has the best interface, where you can see what both people are typing. **write** and **wall** are used to send onetime messages, either to a single user or all users. These commands will not display messages to users who have disabled receiving messages via the `set messages off` command. The **wall** command can override these protections if used by the super-user.

Information Commands

finger	finger, finger user , finger ser@machine	<pre> finger Login Name TTY Idle When Where mrfixit Joe VanDyke console 24 Thu 14:06 :0 zeus ??? pts/4 Wed 13:49 finger oper Login name: oper In real life: Systems Operations Directory: /home/opr/oper Shell: /bin/csh No Plan . finger oper@vtsabe.cc.vt.edu [vtsabe.cc.vt.edu] Login name: oper In real life: Systems Operations Directory: /home/opr/oper Shell: /bin/csh No Plan. </pre>
who	who	<pre> who mrfixit console Jul 23 14:06 (:0) mrfixit pts/3 Jul 29 13:35 (:0.0) zeus pts/4 Jul 29 13:49 (maelstrom.cc.vt.edu) </pre>
w	w	<pre> w 02:16PM up 60 days, 4:11, 5 users, load average: 0.15, 0.05, 0.04 User tty login@ idle JCPU PCPU what shewchuk pts/2 01:34PM 0 0 0 rlogin rhodes pts/5 08:39AM 3:09 0 0 -ksh shewchuk pts/6 01:34PM 7 0 0 rlogin jaganjk pts/7 10:08AM 1:24 1 0 vi mrfixit pts/8 02:16PM 0 0 0 w </pre>
hostname	hostname	<pre> hostname betelgeuse.cc.vt.edu </pre>
uname	uname	<pre> uname -a SunOS betelgeuse.cc.vt.edu 5.6 Generic_105181-05 sun4u sparc SUNW,Ultra-1 </pre>

Information commands allow you to find out what user ids are logged on to a system. They can also tell you where the person logged in from if they came in remotely, and what program or programs they are running. **finger** with no arguments will give you information about who is logged on, if you **finger** a particular user, it will give you detailed information about the userid. **who** will give you information about who is logged in and where they logged in from. The **w** command will tell you what a particular user is doing on the system. The **hostname** command will tell you the name of the system you are on. This is useful when you have multiple connections to other machines active on a machine. The **uname** command can give you information on the hostname and operating system that you are using.

Internet/Intranet commands

Internet/Intranet commands are used to communicate with machines and users across the network. While they can also be used to communicate with users on the current machine using mail, telnet and ftp are generally used to go to other machines. They will work to the current machine, however. **telnet** is used to open an interactive session with a remote machine. **ftp** is a mechanism that is used to transfer files to or from a remote machine. **mail** is the basic mail command that can be used to read or send electronic mail from one user to another. E-mail systems vary from machine to machine, but most have both **mail** and **mailx** as default mail engines. Other products, such as **elm**, **pine**, **dtmail**, **dxmail**, and **exmh** have been created to give more user friendly interfaces to the mail systems.

telnet	telnet machine
ftp	ftp machine
mail	mail user@machine , mailx user@machine elm user@machine, other mailers are similar.

Secure Communications

It is becoming more and more apparent at this time that secure communications from one machine or person to another is necessary. A very good package, **ssh**, has been created to meet this need. **ssh** has two components in the UNIX world, a client and a daemon. The daemon accepts secure communications, the client originates and receives the communications. Clients are also available for Macs and PC's so those systems can communicate securely with UNIX systems. **ssh** is able, in its most basic use, to function like telnet. It also replaces the functionality of **rsh** and **rexec**, which are powerful commands that allow a user to do many commands remotely. Those commands are inherently insecure, and as such, are turned off on most machines. **ssh** gives you the functionality of running commands remotely, without logging in, over a secure link. We will be removing many services, such as **telnet**, in the future and replacing them with **ssh**. **scp** is a secure way to copy files from one system to another.

ssh	ssh machinename	<pre>ssh betelgeuse.cc.vt.edu Host key not found from the list of known hosts. Are you sure you want to continue connecting (yes/no)? yes Host 'betelgeuse.cc.vt.edu' added to the list of known hosts. mrfixit's password: Last login: Fri Aug 28 13:52:16 1998 from :0 Sun Microsystems Inc. SunOS 5.6 Generic August 1997 No mail. Sun Microsystems Inc. SunOS 5.6 Generic August 1997 (mrfixit@betelgeuse) ~></pre>
scp	scp user@host1:filename user@host2:filename	<pre>scp card.gif mrfixit@vtaix.cc.vt.edu: mrfixit@vtaix.cc.vt.edu's password: card.gif 23 KB 23.2 kB/s ETA: 00:00:00 100%</pre>

ssh can also forward X11 commands through a secure path.

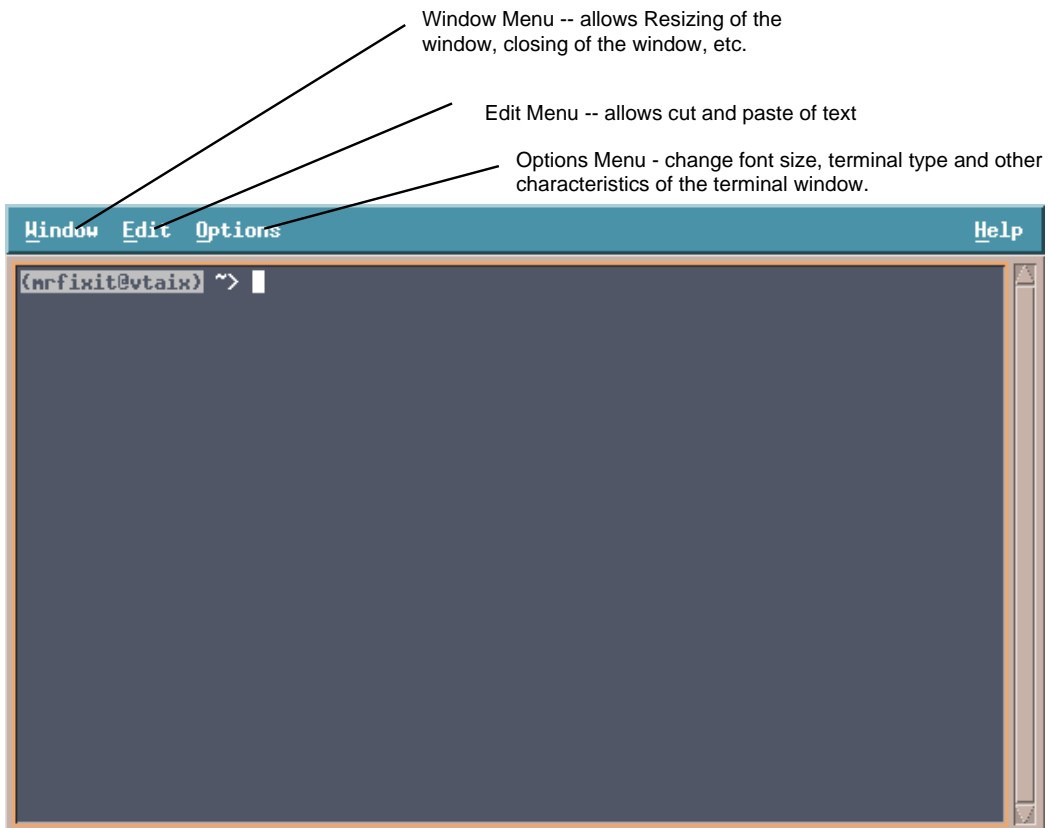
X-windows overview

The X11 system is the graphical interface used most often by UNIX system. This system provides a windowing system much like the Macintosh or MS-Windows. The environment is highly customizable. Most of the machines used in the machine room use the Open Groups' Common Desktop Environment. This provides a common interface across several different hardware platforms. Xwindows is a client server environment, but with a twist. The server is actually the machine displaying the windows, while the client is the executed program. For example, if you bring up `xclock`, it is the client, and your screen is the server. Xwindows is a bi-directionally network intensive environment. It is not recommended for use across modems.

Windows

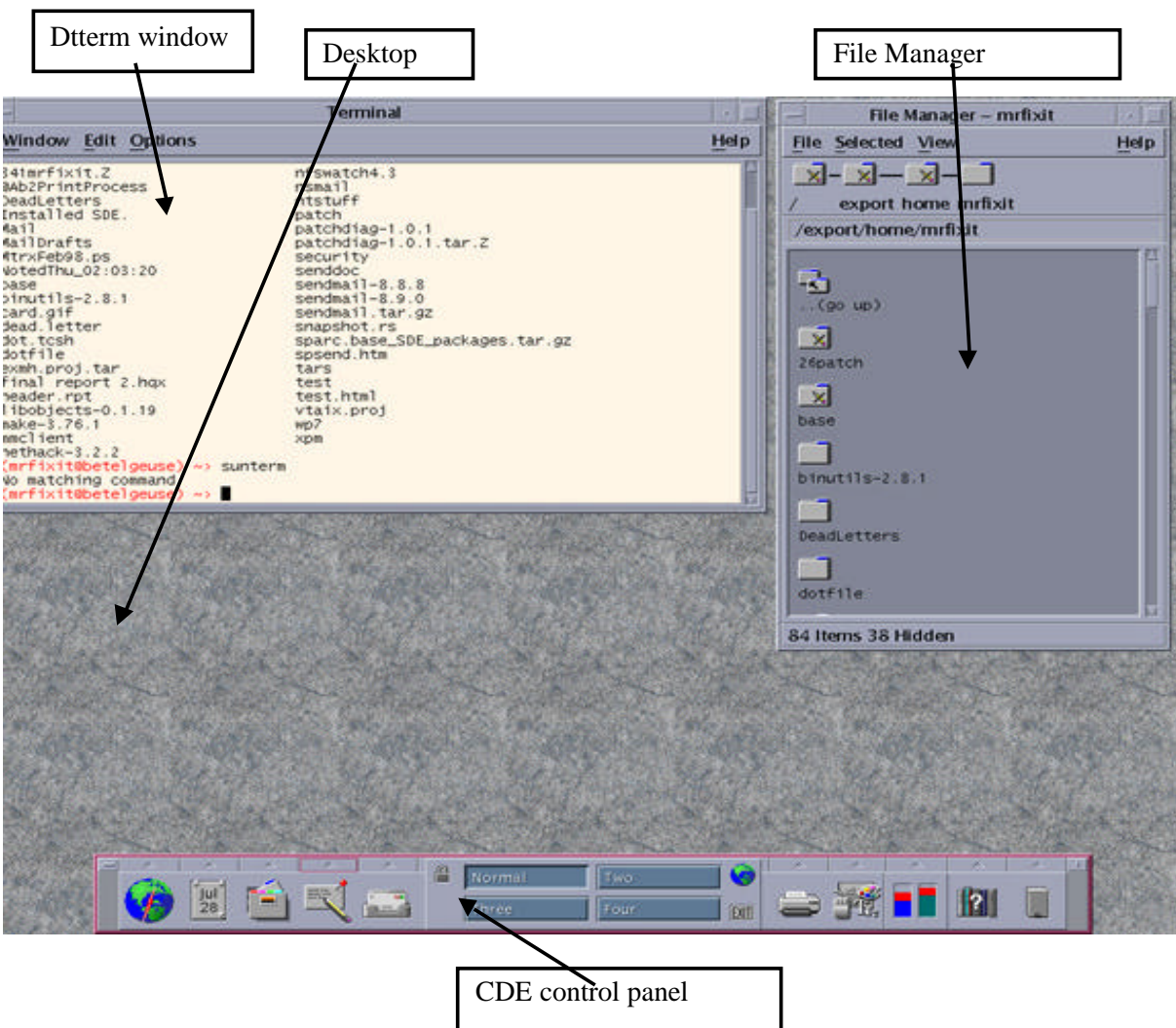
The basic component of X11 is the window. These appear, in most cases, whenever you run a command under X. These windows function very similarly to other windowing systems you have used.

This X-terminal window shows many of the characteristics of X-windows:

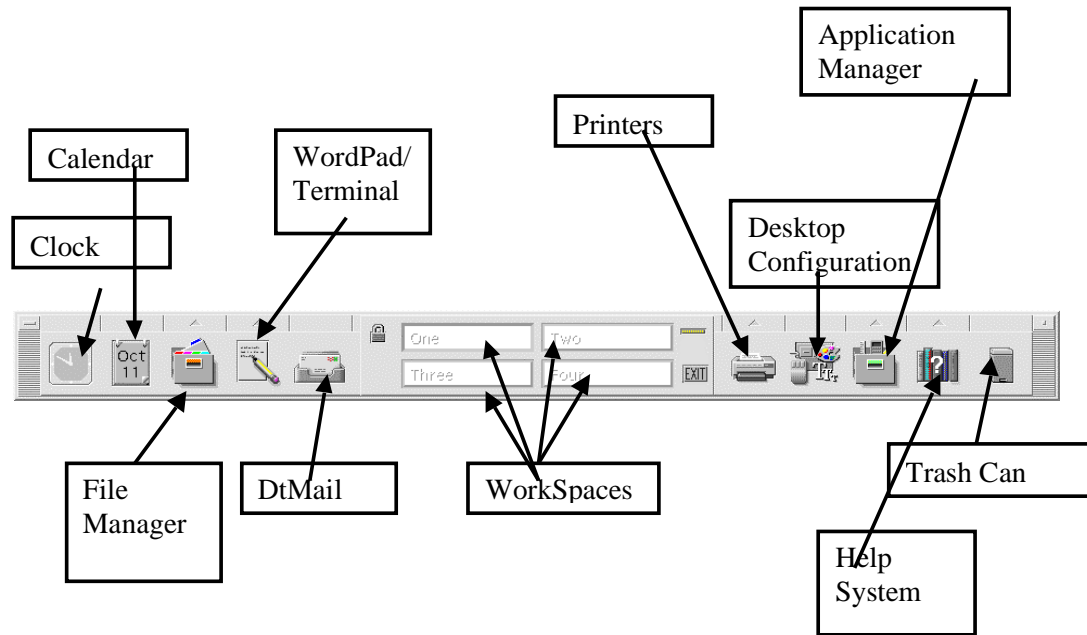


CDE - Common Desktop Environment

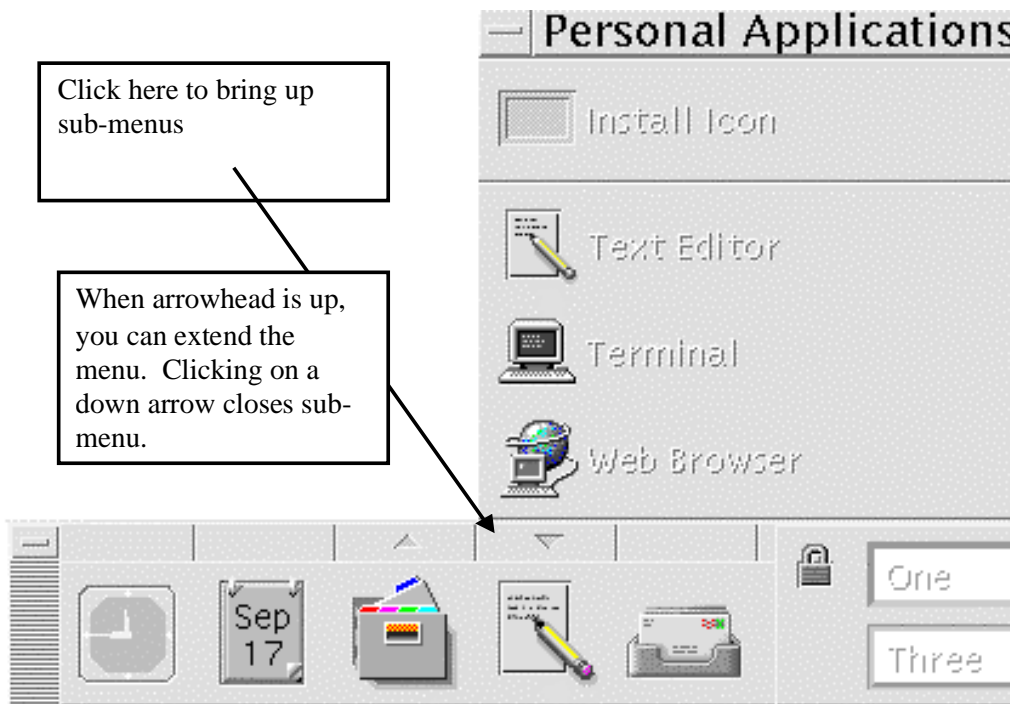
Logging into CDE will present you with a screen like this:



When you enter CDE, you will be presented with several windows, and a menu bar across the bottom of the screen. This control panel allows you to do many things with mouse clicks you would otherwise need to type commands for. The control panel looks like this



In addition, some of the panel buttons have sub-panels under them – the example below is of the Applications panel.



Many things can also be accomplished by clicking on the desktop area with the right button on the mouse. This will bring up a menu of actions that you can perform. Bringing up a shell is one of the most important, because it allows you to enter UNIX commands and get responses to them.

X-Windows comes with several productivity tools, including a calculator, a text editor, an email interface, and a calendar. All of these can be accessed either from the CDE desktop, the Application manager, or by the command line.

If you execute an X-program from the command line, it is important to always run it as a background process. You can lose control of the machine if you do not run the program in the background.

Examples of commands that you can run from the command line: `xterm`, `netscape`, `xemacs`, and `xclock`.

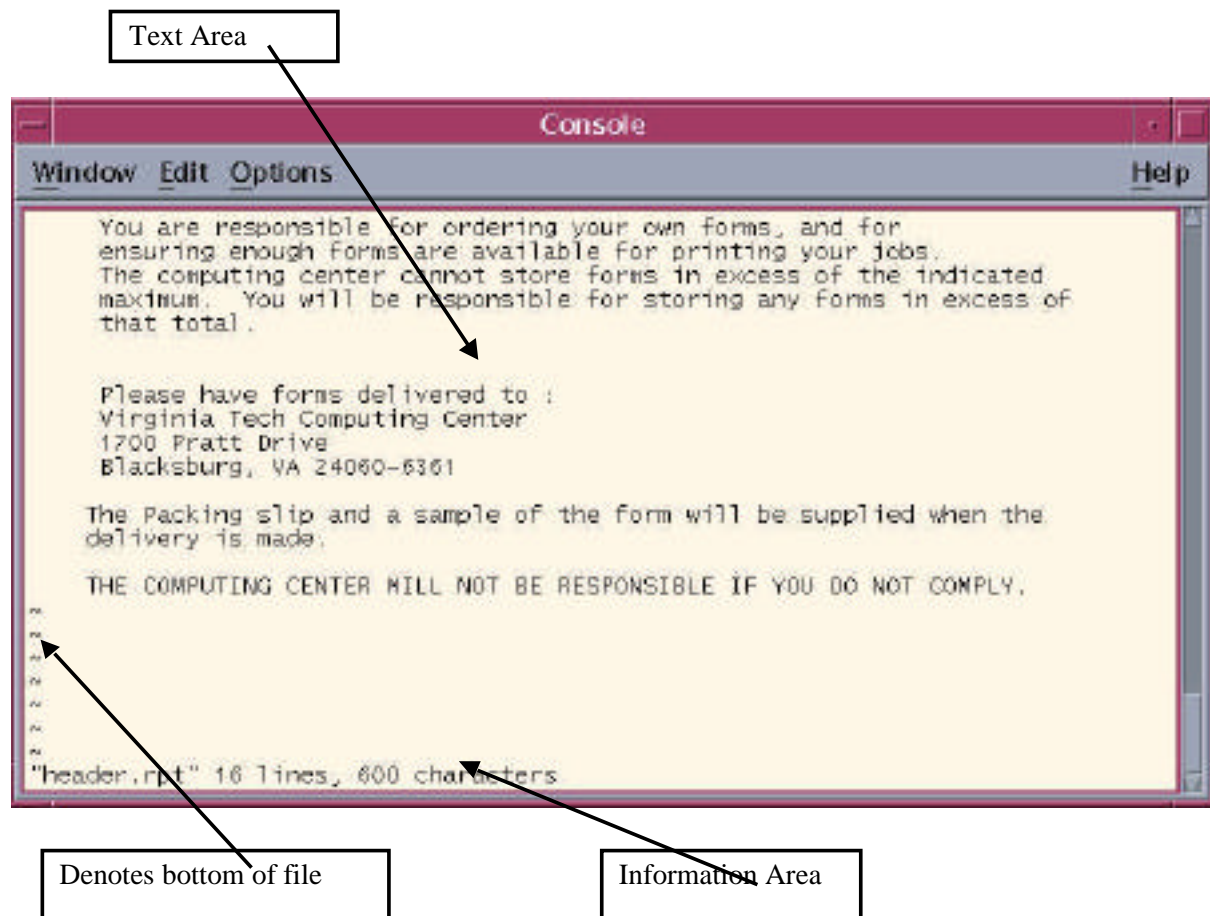
Another important item is the `DISPLAY` shell variable. Setting this variable can allow you to run X-based programs on another machine and get the display on your own workstation. For `csh`, the command is `setenv DISPLAY hostname:0.0`. For `sh/ksh` you use `SET DISPLAY=hostname:0.0; export DISPLAY`. (This is not necessary with `ssh`, since it does this automatically for you.)

Xwindows and the Common Desktop Environment function very similarly to PC based window systems. They provide a convenient way to navigate the system. However, there is very little that you can not do in UNIX via the command line. The X-Windows system is becoming standard, but it is not a required part of the operating system.

The vi Editor

The standard UNIX editor in common use is the visual editor or **vi**. This editor is an extension of the original editor, **ed**. It gives a full-screen line editor to the operating system. Many people exposed to the **vi** editor are intimidated by the complexity of the editor. You do not have to know all of the options to **vi** to be very productive with it. The main thing to remember is that **vi** is normally in **COMMAND** mode, where it will accept a number of different commands. **INSERT** mode is the mode in which you put text into the file. **vi** also has a third mode, **LASTLINE** mode – which functions like command mode. You can cause **vi** to display the mode you are in by doing a **:set showmode** command. This will display the insert mode that you are in, displaying it in the lower right hand corner of the screen. This display can be a bit confusing, however, because it breaks the **INSERT** mode into **INSERT**, **OPEN**, and **APPEND**. These modes correspond with the **INSERT** mode command you use.

The vi editor looks like this:



Summary of INSERT Mode commands

Insert text in front of cursor	i
Insert text after cursor	a
Insert text before first non-blank character	I
Insert at end of line	A
Insert line above current line	O
Insert line below current line	o
Replace character	r
Overwrite character	R

Selected COMMAND Mode commands

These are only a few of the commands that you can use in **vi** :

Go to Bottom of Document	G
Go to Top of Document	1G
Save Document	ZZ
Save Document (LASTLINE MODE)	:wq
Force Save (LASTLINE MODE)	:wq!
Save as another file	:w filename
Quit without saving document	:q!
Backward Search	?searchstring
Forward Search	/ searchstring
Move lines	nny - move n lines of text
Put moved lines above cursor	P
Put moved lines below cursor	p
Delete line	dd
Erase to end of line	D
Erase word	dw
Delete character	x
Delete previous character	X

Most commands can also be prefaced by a number. For example, to delete 4 words, type **4dw**.

Movement commands

Move cursor up	k
Move cursor down	j
Move cursor left	h
Move cursor right	l
Move forward one screen	^F (control-f)
Move backward one screen	^B (control-b)

In addition, most terminals also support moving with the arrow keys. These terminals also move into command mode when you use the arrow keys.

Unix compared to DOS

Some common DOS commands and their Unix equivalents:

DOS Command	UNIX Command
dir	ls
md <i>directory</i>	mkdir <i>directory</i>
cd <i>directory</i>	cd <i>directory</i>
date	date
copy <i>filename.ext filename2.ext</i>	cp <i>filename filename2</i>
del <i>filename</i>	rm <i>filename</i>
erase <i>filename</i>	rm <i>filename</i>
help <i>cmd</i>	man <i>cmd</i>
pkzip	compress/uncompress, pack/unpack, gzip
print <i>fn</i>	lpr -P <i>printer name fn</i>

Many other common DOS commands work the same way – an example of this would be the Java™ compilers and runtimes. Both Unix and Dos use the **java** and **javac** commands.

Additionally, many NT commands are exactly the same due to U.S. Government POSIX compliance standards.

Short List of Mail commands

Read next message	<Return>
Display list of mail commands	?
list the subjects of incoming mail	h
Select message number	#
Send a reply to sender and all recipients	r
send a reply only to sender of message	R
Save mail message to a file	s
Write message to a specific file	w
Edit message with vi	v
Delete a message or list of messages	d
exit (Quit from mail)	q
exit (without changing)	x

ELM Commands

Request help	?
Move cursor down	j or down arrow
Move cursor up	k or up arrow
Create/Send new message	m
Reply to current message	r
Delete message	d
Undelete message	u
Add aliases	a

References

Mark G. Sobell Unix System V: A Practical Guide. 3rd Edition. 1995 Addison-Wesley Menlo Park CA.

Mark G. Sobell A Practical Guide to Solaris Addison-Wesley Menlo Park CA.

Daniel Gilly, et.al. UNIX in a Nutshell. 1994 O'Reilly and Associates. Sebastopol, CA.

Frank G. Fiamingo, Linda DeBula, Linda Condron. Introduction to UNIX 1996. The Ohio State University.

WWW based resources:

<http://www.geek-girl.com/unix.html>

<http://www.cs.vt.edu>

There is also a very good Frequently Asked Questions guide for UNIX. It is available on the geek-girl site.