Basic UNIX course 2003

Conducted by Ong Zhiyang

Institute of Microelectronics, Singapore

Table of Contents

1.0 Preface

1.1 Author's Note

The best way to become familiar with the UNIX environment is to spend a lot of time working in a UNIX environment. Alternatively, you can run the Linux operating system at home. You can partition your hard disk/buy an external hard disk and install Microsoft Windows in one section and a distribution of Linux on the other. That way, you get the best of both worlds. You can also run a lot of free applications on Linux, which is available for free off the Internet. Some of these Linux free software are good substitutes for some of the Microsoft programs.

Download Cygwin, which is a free program that provides a UNIX-look-and-feel in Windows, from the Internet. The operating system Mac OS from Apple also allows users to work in a UNIX environment as well.

1.2 Brief History of UNIX

Bell Labs, General Electric and MIT were developing a new operating system, MULTICS, which would provide multi-user, multi-processor, and multi-level (hierarchical) file system, amongst its many forward-looking features in the late 1960s.

When AT&T dropped out, the name of this Operating System that runs on PDP-7 (it ran on PDP-11 two years later) was replaced by UNIX as a pun for the withdrawal of the AT&T programmers.

After Dennis Ritchie developed the C language in 1972, UNIX was re-written in C with the help of Dennis. Sometime later in 1988, AT&T and Sun Microsystems jointly developed System V Release 4, from which the Sun Microsystems developed Solaris 2.8. At IME, the operating environment of the UNIX system in use is predominantly Solaris 2.7.

UNIX, which is a multi-user, multi-processor system, caught on with programmers because it is designed as a programmers' environment with simple user interfaces and simple utilities that can be combined to perform powerful functions. UNIX has a simple interface to Input and Output devices that are consistent with file format. Lastly, its architecture is independent and transparent to the user.

1.3 Course Objectives

The course objectives are to learn:

1. How to log in and out of the system
2. The fundamental commands of the UNIX operating system to access and manage files as well as directories
3. How to change permission on files and directories
4. How to list and control the processes running on the system
5. Other useful commands
6. Commonly FAQs and Tips

2.0 UNIX Structure

2.1 The Operating System

UNIX is a layered operating system that consists of three layers, as shown in Figure 2.1: the **hardware** that provides the services for the UNIX operating system, the **kernel** that interacts directly with the hardware and provides the services to the **user programs**, which do not require any information about the hardware. Therefore, the concept of encapsulation allows programmers to write programs that are independent of the underlying hardware so that these programs can be reused in other UNIX machines.

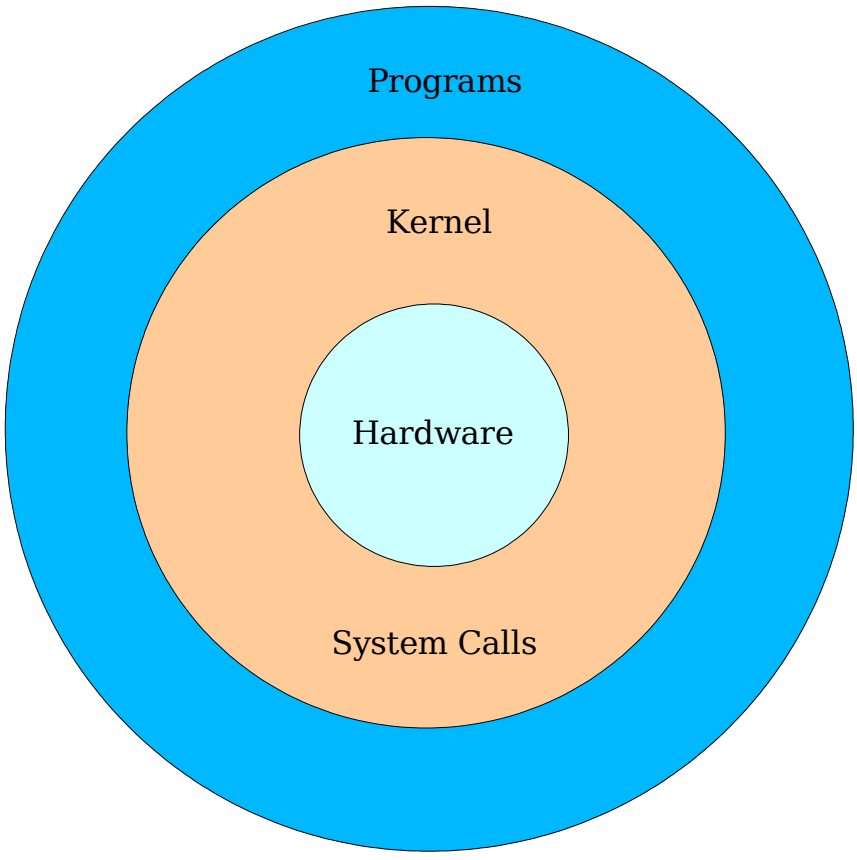


Figure 2.1 The UNIX system structure

By using a set of standard **system calls**, the user programs are allowed to interact with the kernel. These calls request the kernel to provide services such as file access, starting or updating records of a user's account, process control, enabling access to hardware Input and Output devices as well as setting limits on system resources.

Lastly, UNIX allows many users, whom are running many programs each, to be logged into a system simultaneously. The kernel's job is to keep each process and user separate as well as to regulate access to system hardware. In the UNIX operating system, each process has a parent process except for the shell process of the root user. Thus, if a child process “hangs”, its parent process can be used to “kill” it without affecting the parent process or other child processes.

2.2 The File System

The UNIX file system, shown in Figure 2.2, looks like an inverted tree structure. Beginning with the root directory, which is denoted by "/", it resembles the root of a tree with the subdirectories looking like nodes of a tree where there may be a couple of branches branching off whilst the files look like leaves.

Each node is either a file (external nodes) or a directory (internal nodes) of files that can contain other files and subdirectories. A file or directory may be specified by its full (absolute) or relative path name.

A full path name begins with the root, /, and follows the branches of the file system, each separated by "/", until you reach the desired file, "acroread.txt". For example, if the user is at the root directory:

/zhiyang/Acrobat5/Reader/sparcsolaris/bin/acroread.txt

would be the absolute path.

A relative path name specifies the path relative to another, usually following the current working directory the user is at. The "." is a special directory entry used to indicate the current directory whilst ".." is another special directory entry used to indicate the parent of the current directory. For example, if the user is at the directory "zhiyang/OpenOffice.org1.1.0/user", a relative path to get to the "acroread.txt" file is:

../../Acrobat5/Reader/sparcsolaris/bin/acroread.txt

The "../.." allows the user to ascend two levels up the file hierarchy. Thereafter, the user descends to the required file.

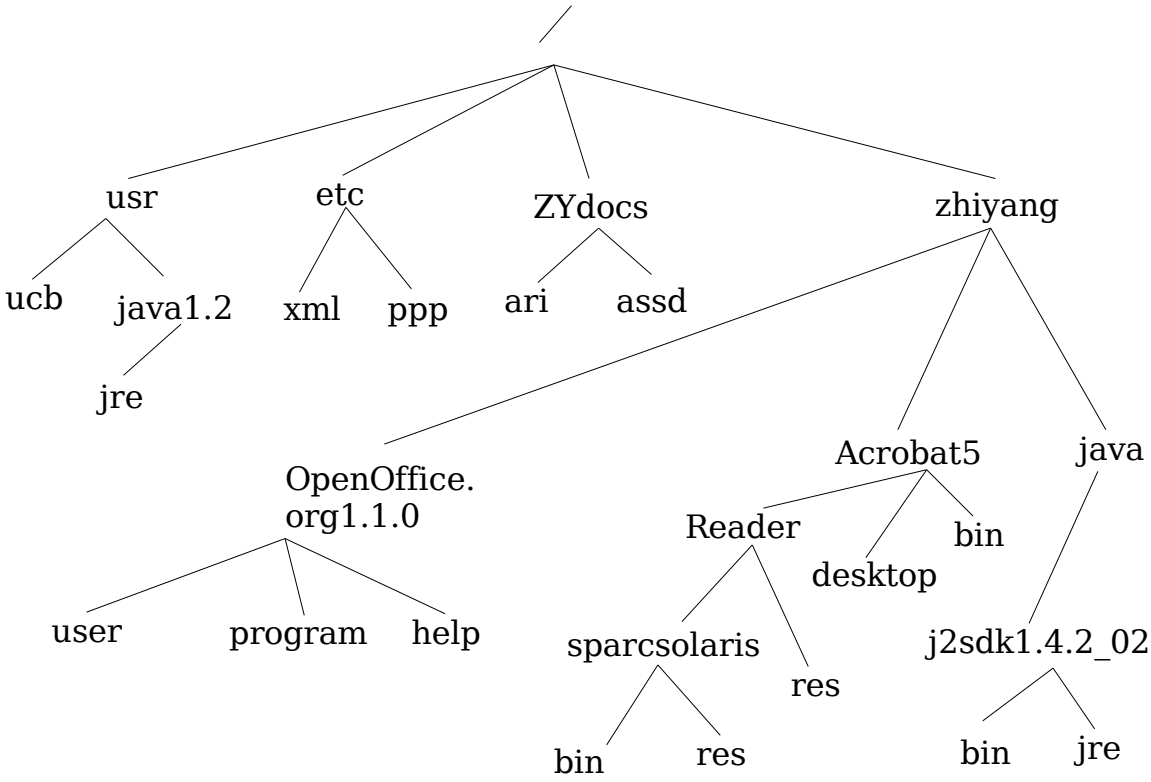


Figure 2.2 A pictorial description of the UNIX file structure.

2.3 UNIX Directories, Files and Inodes

With the exception of the root directory, every directory and file is listed in its parent directory. The parent directory of the root directory is the root directory itself. A directory is a file that contains a table listing the files contained within it. In that table, each filename is assigned to the inode number in the list. An inode is a special file designed to be read by the kernel; it provides information (such as its permissions, ownership, the physical location of the data blocks on the disk containing the data as well as dates of creation, of last access and modification) about each file.

The UNIX file system does not require any particular structure (it can be ASCII, binary, or a combination of both) for the data in the file itself. That is, UNIX does not yield information about the type of the files in a directory since it treats files of all types as the same.

2.4 UNIX Programs

A **program**, or **command**, interacts with the kernel to provide the environment and perform the functions called for by the user. A program can be a shell script, which is an executable shell file written in ASCII, a built-in shell command or complied source (that is the object code file).

A **shell** is a command line interpreter and facilitates the user interaction with the kernel. System programs are complied C source code in binary that provide UNIX functions like **sh**, **csh**, **date**, **who** and **more**.

3.0 Getting Started

3.1 Logging In

To gain access to UNIX, the user must satisfy login requirements to help the system recognise that the user as an authorised user. After connecting with a UNIX system, the user is prompted for a **login** name, which is a unique name belonging to the user on the system, followed by a **password**, which is a changeable code known only to the user.

The user should type in the username at the **login** prompt and the current password at the **password** prompt. Do **not** interchange the order of the login username and password entries. **UNIX is case sensitive.** Therefore, **do not** leave the CAPS LOCK on (to avoid confusion) or change the case of the alphabets in the login username or password.

Please duly note that if a user has exceeded the memory space quota allocated to him by the system administrator, the user will not be able to log in. This user will have to seek help from the system administrator. As a result, users must always ensure that they do not exceed their disk quota by using the ***du*** command, mentioned in Section 4.1.

Always log out after each UNIX session to protect your terminal and prevent unauthorized users from taking advantage of your access. Type “exit” at the shell prompt to log out using the Bourne shell. Type “logout” if a C shell is used. If logout is successful, the login prompt will display on the screen.

Types of login

1. Console

   The user logs into the UNIX system at a locally connected UNIX console.

2. Remote

   a) rlogin – Working on a UNIX environment, the user is able to perform a remote login to the system.

      Its syntax is:

      rlogin -l username hostname

   b) telnet – A user is able to use his remote computer to function as a working terminal.

      Its syntax is:

      telnet hostname

   c) Using emulators, like Exceed, a user is able to login to the UNIX system

Password

The password issued to you initially by the system administrator must be changed to ensure that the security of that user account is not compromised.

The ***passwd*** command is used to change your password. You must enter your old password as well as to key in and verify your new password. The system will not change the user's password if there exists a typographical error in the old password or discrepancy in the new password entries.

Passwords selected must be memorable and unobvious to avoid hacker break-ins. Passwords should be **kept confidential** (do not record your password anywhere except in your memory). That is, do not use passwords such as:

- A word (or words) in any language
- A proper name

Do not use:

- Information that can be found in your wallet
- Information commonly known about you (car license, pet name, etc)
- Control characters such as: '*', '[', ']' and '?'

Passwords mixed with a mixture of character types (alphanumeric, numeric, special), a mixture of upper case and lower case, with a minimum of six characters will be hard to crack... For example, try “cH#i54.orLA”. Do change your password often!

Passwords forgotten will be replaced by a temporary password issued by the system administrator that you should change.

Exiting

^***D*** – This indicates the end of the data stream; a user can log off. The latter is disabled on many systems. Note that the '^' symbol is used to represent control. That is, this command is ***Ctrl-D.***

^***C*** – This is used to interrupt the current process and bring up the shell prompt on the shell terminal again.

logout – Leave the system

exit – Leave the shell

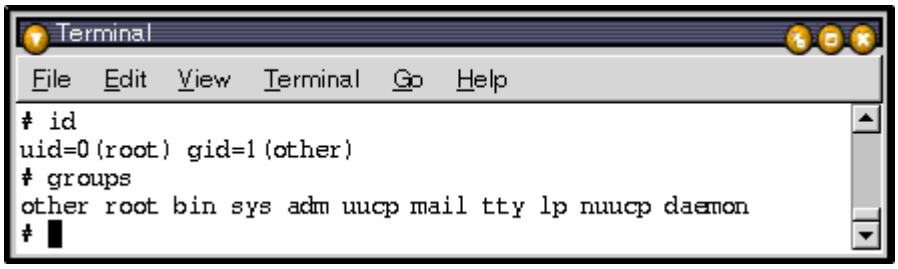
Identity

The UNIX system identifies each user by the user's user number (userid) and group numbers (groupid) assigned to that user by the system administrator. Users do not need to know their userid or groupid since UNIX translates username and groupname into userid and groupid automatically at login.

Users may belong to more than one group. The user's primary group is the one associated with your username in the password database file set up by the system administrator, whom also has a group database file to assign users with rights to additional groups on the system.

The ***id*** command is used to determine the userid of a user whilst the ***groups*** command displays the list of groups that user belongs to. Note that some systems allow the former command to display either the primary group information or additional group information.

For example,



3.2 UNIX Command Line Structure

A **command** is a program that tells the UNIX system to do something. Its syntax is:

command [options] [arguments]

where an option, usually preceded by a hyphen, modifies the way the command performs whilst an argument indicates what the command will perform its action on, usually a file or a series of files.

When a command is used with multiple options, the options may be concatenated or separated. That is, the syntax of a command with multiple options can either be:

command -[option1][option2][option3]

or

command -option1 -option2 -option3

For example:

ls -alp

or

ls -a -l -p

Options requiring parameters are usually specified separately. Note that not all UNIX commands follow the same format. Hence, some commands will not allow options to be concatenated or require a hyphen to precede the option.

Options and syntax for a command are listed in the **man page** for the command. The syntax for that is:

man command_name

For example:

man ls

3.3 Control Keys

Control keys are used to perform special functions on the command line or within a text editor. Hold down the **Control** key and another key simultaneously to type these control keys. An example is the **Ctrl-C** command key aforementioned in Section 3.1.

Note that with control keys, the case do not matter.

Some other control keys are:
^***S*** – Tell the terminal to stop accepting input
^***Q*** – Tell the terminal to start accepting input
^***U*** – Erase the entire input line

3.4 Getting Help

UNIX has a manual, called the **man pages**, available on-line to explain the usage of the UNIX system and its commands. The syntax for the **man** command, as aforementioned in Section 3.2, is

man [options] command_name

Two common options are:

-k List command synopsis line for all keyword matches

-a Show all matching man pages in SVR4, which is System V Release 4 of the joint developed UNIX system. Thus, Solaris 2.8 allows this option to be used.

For example, type ***man find*** to find out more about the ***find*** command.

An alternative to the man pages is the help option available for some commands. For example, ***xterm -help*** allows the user to find out more information about the xterm command.

3.5 Directory Navigation and Control

The UNIX file system is set like up a tree branching out from its root. The **root** directory is symbolised by the forward slash (/). System and user directories are organised under the **root**. All users normally log into their own **home** directory belonging to their account. Users are able to create other directories under their home directories.

Command/Syntax	Consequent Actions
cd	Change directory; **cd** – Changes directory to the user's home directory. Note that no arguments are used. **cd /** – Changes directory to the system's root. **cd ..** – Returns to the parent directory of the current directory. **cd ../..** – Goes up two directory levels **cd subdirectory_name** – Change to the subdirectory with the name "subdirectory_name" **cd ../subDir** – Change to the subdirectory "subDir" of the current directory's parent

Command/Syntax	Consequent Actions
ls	List the contents of the directory; **ls** – List the visible contents of the directory. That is, files beginning with a dot (.) will not be listed since they are normally not visible; **ls -a -l** – List all the contents of the directory, including hidden files (those files that begin with a dot that are usually not listed), in the long format.
mkdir	Makes a directory; **mkdir subDir** – A subdirectory, named “subDir”, is created in the current working directory **mkdir ../subDir** – A subdirectory, named “subDir”, is created in the current working directory's parent
rmdir	Removes an empty directory. If the directory is not empty, the directory needs to be emptied before it can be removed. This command cannot be used to remove the current working directory; you need to go to parent of the current working directory to do so. **rmdir subDir** – Remove the subdirectory, named “subDir”, from the current working directory
pwd	Displays the location in path (that is, the file system hierarchy), which is the current (present) working directory.

Note that these commands are similar to those of DOS.

Also note the following examples for the ***cd*** command:

cd /full/path/name/from/root – Changes directory to absolute path named (note the leading slash)

cd path/from/current/location – Changes directory to path relative to current location (no leading slash)

cd ~username/directory – Changes directory to the named username's indicated directory (Note: the ~ is not valid in the Bourne shell)

3.6 File Maintenance Commands

Use the following commands to create, copy, remove and change permissions on files:

The ***cp*** command is used to copy one file to another. Its syntax is:

cp [options] old_filename new_filename

If the file new_filename does not exist at the time this command is called, a file named "new_filename" is created. Else, the file named "new_filename" is overwritten. The file named "new_filename" is a copy of the file named "old_filename". That is, these two files have identical contents and are independent of each other. Either file can be modified or edited as required. Each of these files has its own inode, data block and directory table entry.

Two common options for the ***cp*** command are:

-i Interactive (prompt and wait for confirmation before proceeding)

-r Recursively copy a directory

-p Preserve the owner and group id, permissions modes as well as modification and access times in addition to duplicating the contents of the source files

To copy a file named “file1” in its subdirectory “subDir” into another file, “file2”, in the current directory's parent, do:

cp subDir/file1 ../file2

The ***mv*** command is moved from one file into another. Its syntax is:

mv [options] old_filename new_filename

If the file new_filename does not exist at the time this command is called, a file named “new_filename” is created. Else, the file named “new_filename” is overwritten. The contents of the file “old_filename” is moved into the file “new_filename” and “old_filename” is destroyed. That is, a file (old_filename) is renamed (to new_filename).

Two common options for the ***mv*** command are:

-i Interactive (prompt and wait for confirmation before proceeding)

-f Disenable prompts, even when copying over an existing file.

The ***rm*** command is used to remove a file. Its syntax is:

rm [options] filename

Some common options of this ***rm*** command are:

-i Interactive (prompt and wait for confirmation before proceeding)

-r Recursively remove a directory; first by removing the files in it followed by removing the subdirectories underneath it.

-f Disenable prompts (Overwrites the ***-i*** option)

For example:

rm -r -f subDir will remove the subdirectory subDir as well as all of its (subDir's) subdirectories (and files) without prompting.

rm *.java will remove all “Java” files in the current directory

Be very careful when overwriting a file or deleting it. UNIX does NOT have a "unremove" command. The system administrators may not necessary have backups of all your files. Hence, data lost will mean that you have to redo some of your work!

The ***rm***, ***mv*** and ***cp*** commands can be used to remove, move or copy files from parent directories or subdirectories to other subdirectories and/or parent directories by using ../ to move up one directory level or ./subDirName to access the subdirectory "subDirName".

The touch command allows users to set the access and modification times of each file. Its syntax is either of these:

touch [-acm] [-r ref_file | -t time] file

touch [-acm] [date_time] file

The file operand is a path name of the file(s) whose access times will be modified. File is created if the file(s) does not exist. The time operand is used to specify the time of the file, either by using the corresponding time fields of the file referenced by the -r ref_file option, by the date_time operand or the current time.

Its options are:

-a Change the access time of the file

-c Do not create a specified file if it does not exist. No diagnostic messages concerning this condition will be written.

-m Change the modification time of file

-r ref_file Use the corresponding access and/or modification times of the file named by ref_file instead of the current time

-t time Use the specified time instead of the current time. The time will be a decimal number of the form:

[[CC]YY]MMDDhhmm [.SS]

The date_time operand is used to specify the specified date and time instead of the current time. This operand is a decimal number of the form:

MMDDhhmm[YY]

For example:

touch -acm -t 03152357 UNIXcourse01.sxw

sets the access and modification time of the file "UNIXcourse01.sxw" to 2357 hours on the 15th of March, 2003.

The ***ls*** command can be used to list all the subdirectories and files of the current working directory. It can also be used to list file permissions as well by using:

ls -l

The output of this command has the following format:

-rwxr-x--- user unixgroup size Month day hh:mm filename

The first three characters following the first dash represent the file access permissions of the user whilst the next three represent that of the group assigned to that file and the last three represent that for the other users on the system.

r indicates read permission, ***w*** indicates write permission, **x** indicates execute permission whilst '–' represents no permission is given.

The ***chmod*** command is used to change file permissions on an item (file, directory, etc). Its syntax is:

chmod abc [argument list] numeric mode

chmod [who]op[perm] [argument list] symbolic mode

[argument list] contains the file, list of files or directories that you want to change. To indicate more than one file or subdirectory, append one after another with a space in between them.

For the numeric mode:

abc indicates the permissions that you want to set for the user (owner), group (that the owner belongs to) and other users.

Each of the permission types is represented in this mode by a numeric equivalent.

4 is used to represent read, 2 represents write and 1 represents execute. Sum these numerical equivalents to determine the permissions desired for the user (indicated by the character 'a'), group ('b') or other users ('c').
For example,

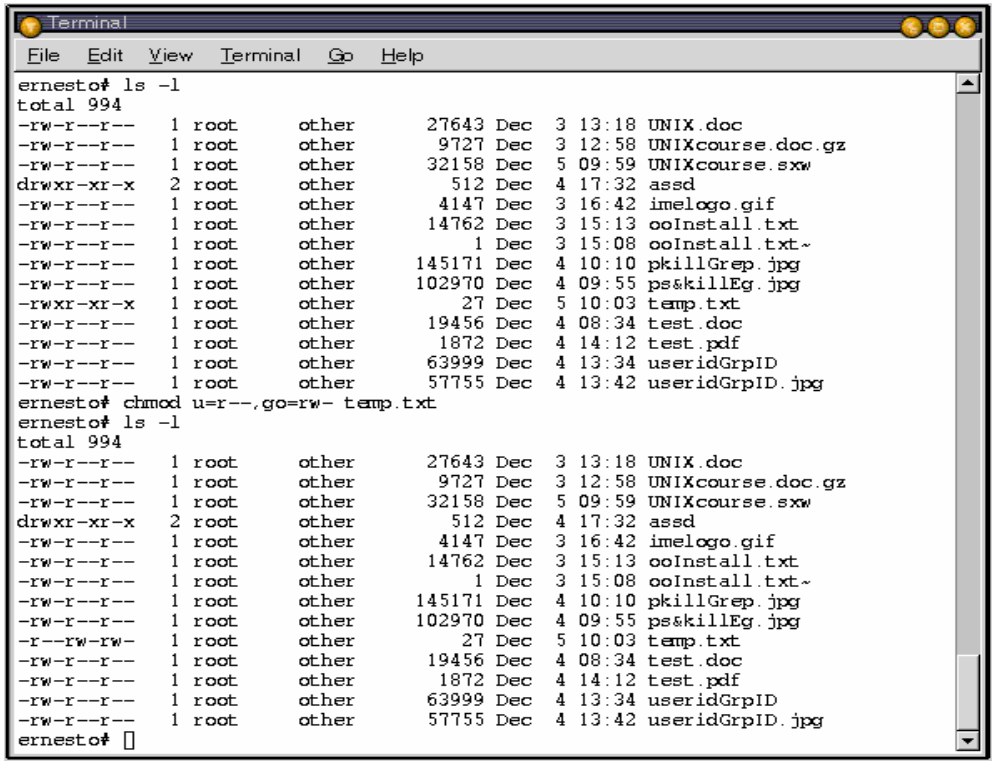
chmod 750 temp.txt test.class

allows the user to read, write and execute (since a = 4 + 2 + 1 = 7) the files “temp.txt” and “test.class”, allows the group to read and execute these files (since b = 5 = 4 + 1) and does not give other users any permission (since c = 0).

For symbolic mode,
[who] is used to indicate the user (u), group (g) or other users (o)
op indicates whether permissions are added to (+) or removed from(-) the current permissions given for that argument list or set (=) for that argument list.
[perm] indicates what permission – read (r), write (w) and/or execute (x) - is given.
For example:

chmod u=rwx,g=r-x,o-r temp.txt

For example,



This allows the user to only read the file whilst the group and other users can read and write to the file “temp.txt”.

Its common options are:

-f force (no error message is generated if the change is unsuccessful)

-R recursively descend through the directory structure and change the file mode bits of each file named by one of the file operands. This allows users to change the file permissions on all items in the specified directory(s) without having to go through all of its subdirectories.

Hence, UNIX allows users, based on their preferences, to change the file mode bits of files ending with the same type in the current directory, a number of specified files and/or all items in a directory.

3.7 Display commands

The ***echo*** command is used to repeat, or echo, the argument you give it back to the standard output device whilst ending with a line-feed usually.
For example, try ***echo pass.txt***

The ***cat*** is used to concatenate the contents of a list of files and display it on the shell terminal. Note that, this does not modify the contents of the files used as arguments of this command.
For example,

cat file1, file2, file3

concatenates the contents of these files in the order they were listed in this command without modifying the contents of any of these files.

The ***more*** command allows you to page through the contents of a file one screenful or line at a time. Some UNIX systems allow ***less*** or ***pg*** to be substitute commands. The ***more*** command has the following syntax:

more [options] [+/pattern] [filename]

For example, "***more Factorial.java***" allows you to view the specified Java file on the UNIX shell terminal. Press the SPACEBAR key to read the next page/screen, each page means the amount of the text file's contents that can be fitted into the size of the shell terminal , of the file, Factorial.java, or press the ENTER key read the next line of the file. Press q to quit viewing the file. Press PAGE_UP to read the previous screen.

The ***view*** command also allows you to view text files. Type ***view filename*** to view a text file named filename. Use ^U or ^D to scroll up or down half a screen of text. Type ***ZZ*** to exit and display the command prompt again.

Users can also use the ***pg*** command to view a text file. Type ***pg filename*** to view a text

file named filename. Type ENTER to view the next screenful of text as well as to exit once the end of the text is reached. The (EOF) message is used to indicate the end of the textfile. Type q or Q to quit.

The ***head*** and ***tail*** commands can be used to display a determined number of lines from the top or the bottom of the file. The number of lines displayed is 10 by default. Use the ***-n*** option, where n is an integer greater or equal to zero, to specify the number of line you want to be displayed.

For example, ***head -7 Factorial.java*** displays the first seven lines of the specified Java file.

4.0 System Resources & Printing

4.1 System Resources

The ***ps*** command is used to determine which process is currently running (or not running) and to get the following detailed information about each process at the time this command is typed.

- PID (Process ID) – Data required to kill a process
- TTY (Control Terminal and Priority) – The controlling terminal for the process (the message, ?, is printed when there is no controlling terminal)
- Time – Cumulative execution time for the process
- CMD (Command) – The command name (the full command name and its arguments, up to a limit of 80 characters, are printed under the -f option)

Some commonly used options for the ***ps*** command are shown in Table 1:

*Table 1 Commonly used options for the **ps** command*

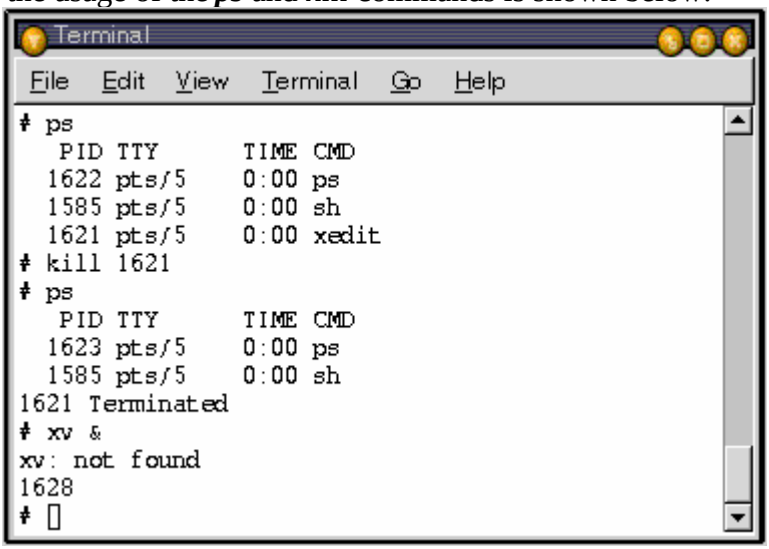
Option	Description
-a	List information about processes most frequently requested, except for process group leaders and those not associated with a terminal
-c	Prints information in a format, which affects the output of the -f and -l options when they are used, to reflect scheduler properties
-e	List information about every process that is currently running.
-f	Generates a full listing
-l	Generates a long listing

The ***kill*** command eliminates a process that is currently executing. It is recommended that this command be used only if the normal way of quitting a process/application does not work. The syntax for using the ***kill*** command is:

kill -signal PID

where signal is a number or name.

An example of the usage of the ***ps*** and ***kill*** commands is shown below:



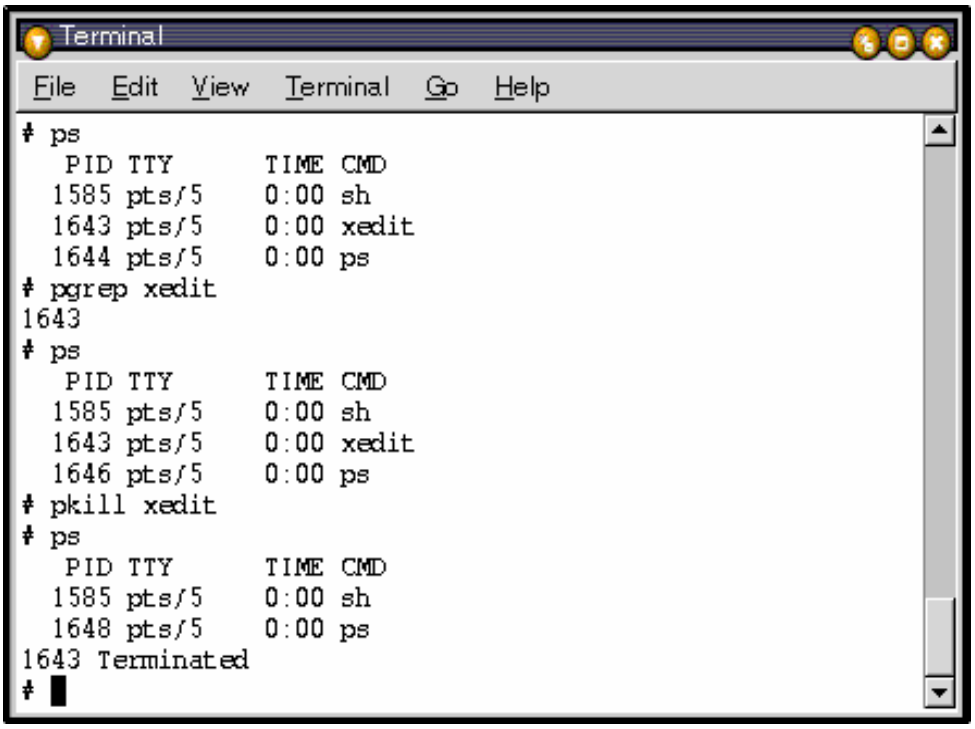
The ***pgrep*** command is used to examine the active processes on the system and report the process IDs of the processes whose attributes match the command-line argument.
An example of the syntax for this command is:

pgrep process-name

It allows the process ID of the specified process, known as “process-name above”, to be found.

The ***pkill*** command is used to terminate a process by its process name. Each active process on the system is examined. It is terminated when its name is found to match the specified process name.

An example of how the ***pgrep*** and ***pkill*** commands are used is shown below:



The ***date*** command is used to display the current date and local time as well as the time zone of that time. The ***-u*** option is used to display the time in Greenwich Mean Time.

The ***du*** command is used to report the amount of disk space in use for the files or directories specified. Its syntax is:

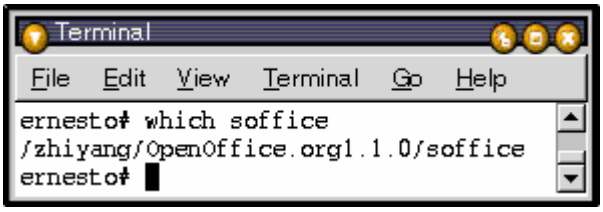
du [options] [directory or file]

When du is used alone, the amount of disk spaced used by the current working directory is displayed at the shell terminal. A ***-a*** option is used to display the disk usage for each file, not just subdirectories. The ***-s*** option displays a summary of the total disk usage only. The ***-k*** option displays the disk usage in kilobytes.

The ***quota*** command is an alternative command for the ***du*** command. It displays a user's UNIX file system disk quota and usage. The ***-v*** option is used to display the user's quota on all mounted file systems where quotas exist.

The ***which*** command reports the path (or the alias that is found first in your path) to the command or the shell alias in use. Its syntax is: ***which command_name***

For example,



Hence, if a command cannot be found, the user can use the ***which/man*** command to check if the relevant path is set in the path variable.

The ***who*** command reports who is logged in at the present time.

The ami (am i) option displays the username of the user who is currently logged in.

4.2 Print Commands

The ***lpr*** or ***lp*** command is used to submit a specified file, or standard input, to a defined printer for printing. Each print job is given a unique request-id that can be used to follow or cancel the job while it is in the queue. Their syntax are:

lpr [options] filename

lp [options] filename

Some commonly used options of the ***lpr*** or ***lpr*** command are:

-n Where n is an integer greater than 0. It indicates the number of copies to be copied.

-d destination Where the destination is the printer name of the desired printer.

The ***lpq*** or ***lpstat*** command is used to check the status of the user's print job. It is usually used with the ***-P*** or ***-p*** status respectively to indicate the status of a specific printer.

The ***lprm*** or ***cancel*** command is used to remove a user's print job from the print queue. A user cannot remove the print job of another user. The syntax of these commands are:

cancel [request-ID (of the print job)] [specified printer]

lprm -Pspecified_printer [request-ID (of the print job)] [username]

where specified_printer is the name of the printer where the print job is sent to.

The ***pr*** command is used to filter the file, and prints the header and trailer information surrounding the formatted file to the terminal. Options, some of which are listed below, are used to specify the number of pages to be printed, lines per page, columns, line spacing, page width, etc. to print along with header and trailer information. Its syntax is:

pr [options] filename

+page_number	Start printing with page page_number of the formatted input file.
-column	"column" indicates the number of columns that the formatted file will be printed to in the terminal
-d	Indicates the formatted file is displayed with double spacing
-l	Indicates the number of lines per page to be printed
-t	Indicates that the header and trailer on each page will not be printed
-w	Indicates the width of the page.

5.0 Shells

The UNIX shell acts as an interface between the user and the operating system by acting as a command-line interpreter that reads typed commands and passes them on for further action to be taken by the UNIX operating system after translation.

Numerous shells are available from the network. Two commonly used shells are the ***Bourne*** shell, sh, and the ***C*** shell, csh. The former uses the ***$*** symbol as its shell prompt whilst the latter uses the % symbol. Type csh at the Bourne shell prompt to change from the Bourne shell to the C shell. The shell prompts the user for UNIX commands that are brief sets of letters, which can be followed by options and/or arguments.

Note that the shell prompt for the superuser is always the # symbol.

To enter multiple commands on the same line, type a semicolon (;) between each command, and press “Enter” after the final command. A “not found message” will appear, followed by the shell prompt, if a command unknown to UNIX is entered.

The ***xterm*** command allows a new shell terminal to be opened. For example,

xterm -sb &

opens a new shell terminal with scroll bars (due to the ***-sb*** option). The symbol ***&*** allows an application to remain open when the user accesses the shell terminal that was used to execute that application. In this case, the application is the shell terminal. That is, the command is run in the background.

In the C shell, the ***alias*** command can be used to assign a name to a function. The ***bg*** command is used to put a job into the background whilst the ***fg*** command is used to bring a job to the foreground. Typing ^***Z*** followed by ***bg*** brings an application to the background and typing ***fg*** brings it back to the foreground.

The ***set*** command is used to set a shell variable whilst the ***setenv*** command is used to set an environment variable for this and subsequent shells. Some commonly used environment variables are:

EDITOR	The path to your default text editor.
PATH	Paths to be searched for commands

You can set these variables to change your default text editor or to add a path to your UNIX operating environment so that a command need not be entered with the full path preceding it.

6.0 Useful Commands

The ***grep*** command is used to search for generalized regular expressions occurring in UNIX files. Its syntax is:

grep [options] regexp [file[s]]

Its common options are:

-i Ignore case

-c Report only a count of the number of lines containing matches, not the matches themselves

-v Invert the search, display only the lines that do not match

-n Display the line number along with the line on which a match was found

-l List filenames, but not lines, in which matches were found.

The ***diff*** command is used to compare two files, or directories, and display the differences (text/ASCII files only) between the two. Its output format is designed to report the changes necessary to convert the first file into the second. Its syntax is:

diff [options] file1 file2

Its common options are:

-b Ignore trailing blanks

-i Ignore the case of letters

-w Ignore <space> and <tab> characters

For example, this command compares two Java files:

diff MyFactorial.java MacFactorial.java

The ***compress*** command is a utility used to reduce the size of the named files. Except when the output is to the standard output, each file will be replaced by one with the extension .Z, while keeping the same ownership modes, change times and modification times.

Its syntax is:

compress [-fv] [-b bits] [file1 file2 file3 ...]

compress [-cvf] [-b bits] [file]

The ***-f*** option forces the compress command to be executed without prompting in the background, regardless of whether the files can be compressed or if a .Z file is made to be compressed. The ***-c*** option writes to the standard output without creating .Z files. The -v option writes to standard error messages concerning the percentage reduction or expansion of each file. The ***-b*** option sets the upper limit, which must be between 9 and 16 (16 is the default), for common sub-string codes. The larger the number, the smaller the compressed file will be. The file operand indicates the path name of a file to be compressed. If file is -, or if no file is specified, the standard input will be used.

If appending the .Z to the file pathname would make the pathname exceed 1023 bytes, the command will fail. If no files are specified, the standard input will be compressed to the standard output. The amount of compression obtained depends on the size of the input, the number of bits per code, and the distribution of common substrings.

The ***uncompress*** command is a UNIX utility that will restore files to their original state after they have been compressed using the UNIX compress utility. If no files are specified, the standard input will be uncompressed to the standard output. This utility supports the uncompressing of any files produced by compress.

The ***gzip*** command is also used to compress the size of the named files. Whenever possible, each file is replaced by one with the extension .gz whilst keeping the same ownership modes, access and modification times. If no files are specified, or if a file name is "-", the standard input is compressed to the standard output. Gzip will only attempt to compress regular files. In particular, it will ignore symbolic links.

For example, the command:

gzip *.jpg

gzips up all the .jpg picture files in the current working directory. That is, the .jpg picture files mypicture01.jpg and mypicture02.jpg become mypicture01.jpg.gz and mypicture02.jpg.gz.

Using the command ***gunzip*** takes a list of files on its command line and replaces each file whose name ends with .gz, -gz, .z, -z, _z or .Z and which begins with the correct magic number with an uncompressed file without the original extension. gunzip can currently decompress files created by gzip, zip, compress, compress -H or pack. Unlike uncompress, gunzip is sometimes able to detect a bad .Z file.

For example, the command:

gunzip *.jpg.gz

gunzips all the .jpg.gz compressed picture files in the current working directory. That is, the compressed, or rather gzipped, .jpg.gz picture files mypicture01.jpg.gz and mypicture02.jpg.gz in the current working directory become restored to their uncompressed format mypicture01.jpg and mypicture02.jpg.

The ***find*** command recursively descends the directory hierarchy for each path seeking files that match a Boolean expression written in the predicate list.

For example, the command

find spare/java/ambardar

searches for all the files in the sub directory of ambardar.

A user can use the File Transfer Protocol (**FTP**) to transfer files from your account in one server to that in another or from one PC workstation to another. To ftp into a server, type the following:

ftp server_name

A login prompt will appear and the user shall login and satisfy the authentication process. To get a file from your account in a server to your PC workstation, change directory to the directory that contains your files by using the UNIX ***cd*** command. To copy those files from your account in the server to your workstation, try:

get filename

This will put the file named “filename” from your account in the server into your workstation.

If you want to put a file from your PC workstation, which runs Microsoft Windows, into your account in the server, use MS-DOS to go to the appropriate directory in the PC workstation and ftp into the server as aforementioned. Go to the directory where you want to transfer the files to and try:

put filename

This will put the file named “filename” from your PC workstation into your account in the UNIX server.

When you have finished transferring the files you desire, type ***bye*** to return to ***MS_DOS*** (if you are in Microsoft Windows) or ***xterm/Terminal*** (if you are logged onto a UNIX workstation).

Typing ***binary*** will transfer files in the binary format whilst typing ***ascii*** will do so in the ASCII format. Test files are usually transferred in the ASCII format.
For example, typing

binary

will result in transferring files in the binary format until the user types the ascii command to change the file transfer type to ASCII.

If a user wants to change the local working directory, use the lcd command. For example, if the user wishes to enter the subdirectory varaguna/mcb/genetics, type:

lcd varaguna/mcb/genetics

To transfer multiple files, use the ***mput*** or ***mget*** commands. For example,

mput*.java

transfers all Java files in the local working directory to the remote working directory whilst the following command

mget Genetics.pdf Cardiology.pdf

transfers Genetics.pdf and Cardiology.pdf from the remote working directory to the local working directory.

The ***mkdir*** and ***rmdir*** commands allow users to create or remove directories from the remote working directory. For example, the command

mkdir mariska

creates a directory called mariska whilst the command

rmdir bogner

removes the empty directory bogner.

Similarly, to telnet into your account in the UNIX server, a user needs to login as the user would with a UNIX workstation. ***Telnet*** is a Internet telecommunications protocol that enables a computer to function as a terminal working from a remote computer. That is, it allows a computer to function as a terminal emulator. The command to telnet into a UNIX server is:

telnet UNIX_server_name

Upon successful login, the user can use UNIX commands just like the user is able to on a UNIX Terminal or xterm window.

To log in remotely to the UNIX servers from a UNIX environment, use the ***rlogin*** command. Its syntax is:

rlogin [-l username] hostname

A remote login from your terminal is established to the remote machine named hostname. FTP only allows file transfer whilst rlogin and telnet allows you to access the remote machine via a terminal

To export display:

- Type “xhost +” at the shell prompt
- Telnet to server
- Set DISPLAY, an environment variable, to that of the IP address of that server

The difference between these ps commands ***/usr/bin/ps*** and ***/usr/ucb/ps*** are:

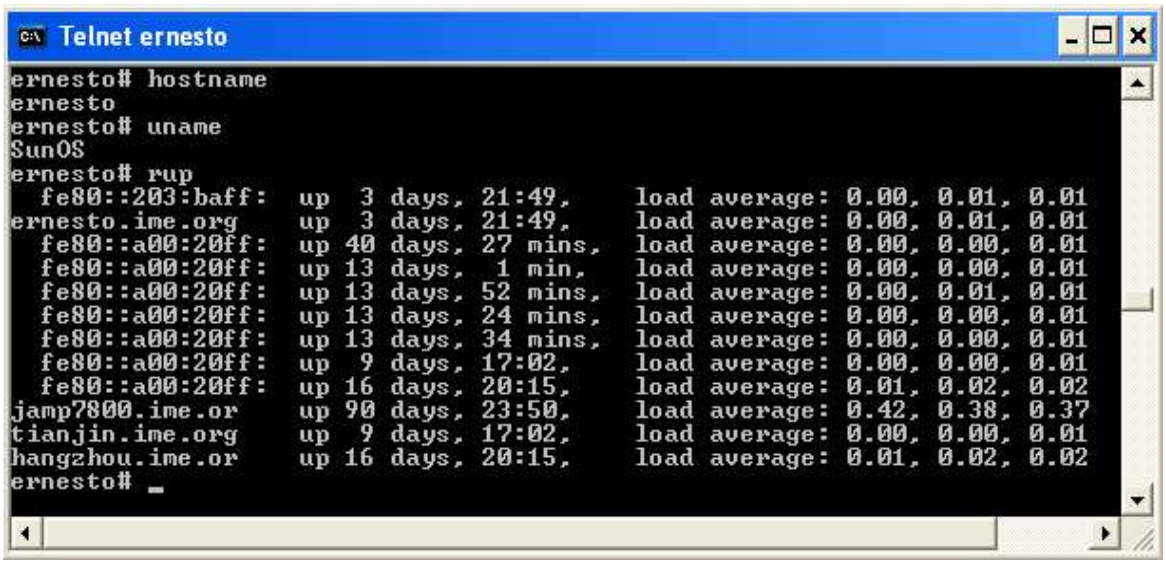
- /usr/bin/ps – includes PID, TTY, TIME and CMD
- /usr/ucb/ps – includes the state of the process as well

The ***hostname*** command is used to set/print the name of current host system.

The ***uname*** command is used to print the name of current operating system.

The ***rup*** command is used to show the host status of remote machines.

For example:



7.0 Special UNIX features

UNIX provides many utilities for doing common tasks or obtaining desired information. Two important features of UNIX utilities are I/O redirection and piping.

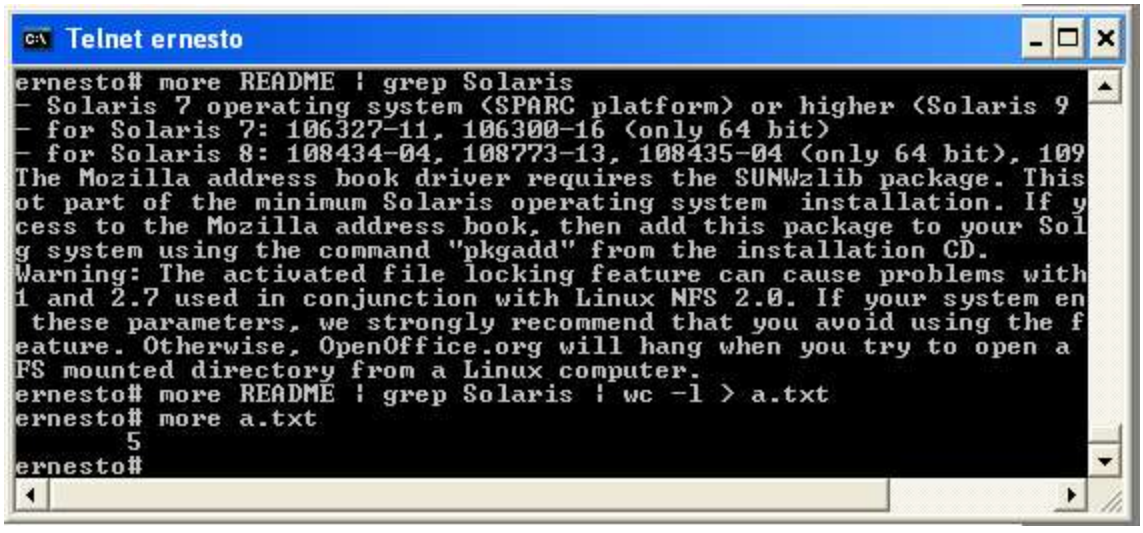
In file redirection, output redirection takes the output of a command and places it into a named file whilst input redirection reads the file as input to the command. The > symbol is used to redirect output whilst the < symbol is used to redirect input. The >> symbol is used to append output whilst the | symbol is used to pipe output to another command. That is, the | symbol filters output of the first command to be the input of the second command. Its syntax is:

command1 | command2

For example,

java < input_file MyDictionary > output_file

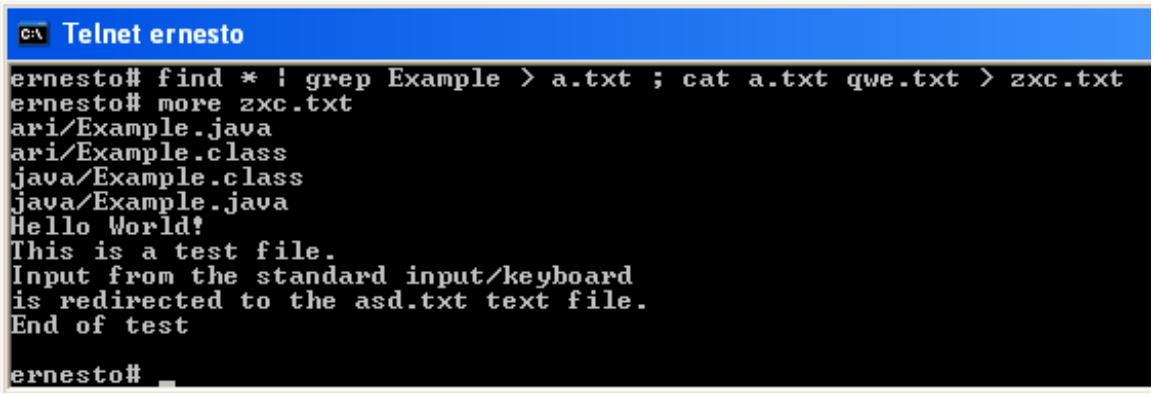
The MyDictionary Java Class (Object) file is executed and it takes input from the file named “input_file” and puts the output of the execution in the file named “output_file”.

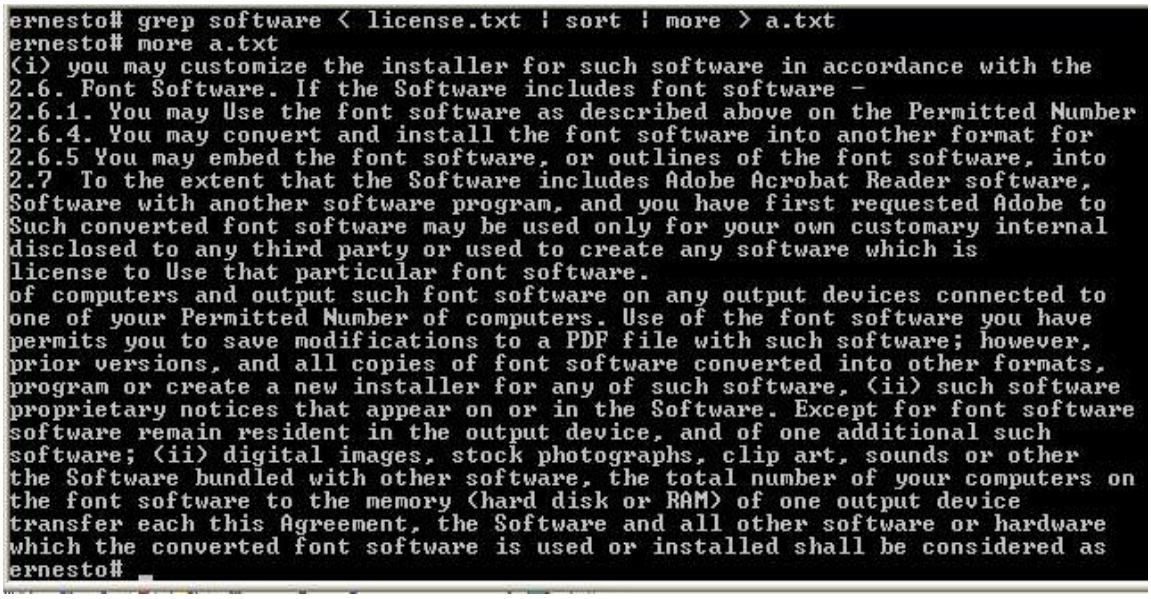


finger | grep csxt | sort | pr | lpr -Pcsdn1

- finger lists the users on the system
- grep searches its input for lines containing the string "csxt"
- sort takes lines of input and sorts them alphabetically
- pr formats input into pages, with header information
- lpr sends its input to the printer.

grep csxt < users_file | sort | pr > output_file





When the system hangs, telnet into your system account and kill the appropriate process.

As a last resort, contact your system administrator.

The shell and some text editors allow ***wild cards*** or ***metacards*** and replace them with pattern matches. The ***? s***ymbol is used to match any single character at the indicated position. The ***** symbol is used to match any string of zero or more characters. ***[abc...]*** is used to match any of the enclosed characters. ***[a-e]*** is used to match any characters in the range a, b, c, d and e. **~** is used to refer to the home directory of the current user in the C shell. ***~user*** is used to refer to the home directory of the specified user in the C shell only.

Note that when an application is not terminated properly, a ***core*** file of about 30 to 80 MB will be deposited in the current working directory of the shell terminal from which this command was run. This ***core*** file is ***USELESS***. Delete it! However, take care not to delete a core directory that contains your working and useful files (e.g. core design files)! To ensure that a core file is deleted instead of a core directory, use the ***more*** command to view the contents of the file.

Files like ****.pdv*** and ****.nfs*** are useless temporary files that CADENCE creates to backup your files. If CADENCE is not shut down properly, they will not be automatically destroyed. Thus, you need to manually delete then from your directory.

8.0 UNIX Command Summary

Commands	Consequent Actions
^C	This is used to interrupt the current process and bring up the shell prompt on the shell terminal again
^D	This indicates the end of the data stream; a user can log off
^S	Tell the terminal to stop accepting input
^Q	Tell the terminal to start accepting input
^U	Erase the entire input line
passwd	Change your password
logout	Leave the UNIX system
exit	Leave the shell
id	Determine the userid of a user
groups	Display the list of groups that the user belongs to
man	Bring up the UNIX user manual
cd	Change directory
ls	List the contents of the directory
mkdir	Make a directory
rmdir	Remove an empty directory
pwd	Display the location in path
cp	Copy one file to another
mv	Move the contents from one file into another
rm	Remove a file
chmod	Change file permissions on an item (file, directory, etc)
echo	Repeat the argument the user gives it back to the standard output device whilst ending with a line-feed usually.
cat	Concatenate the contents of a list of files and display it on the shell terminal
more/less/pg	Page through the contents of a file one screenful or line at a time
head	Display a determined number of lines from the top of the file
tail	Display a determined number of lines from the bottom of the file
ps	Determine which process is currently running (or not running) and to get the following detailed information about each process at the time this command is typed
kill	Eliminate a process that is currently executing
pgrep	Examine the active processes on the system and report the processIDs of the processes whose attributes match the command-line argument

Commands	Consequent Actions
pkill	Examine the active processes on the system and terminates the process, which process Ids/attributes match the command-line argument
date	Display the current date and local time as well as the time zone of that time
du	Report the amount of disk space in use for the files or directories specified
quota	Display a user's UNIX file system disk quota and usage
which	Report the path to the command or shell alias in use
who	Display who is logged in at the present time
lpr/lp	Submit a specified file/standard input to a defined printer for printing
lpq/lpstat	Check the status of the user's print job
lprm/cancel	Remove a user's print job from the print queue
pr	Filter the file, and print the header and trailer information surrounding the formatted file to the terminal
xterm	Open a new shell terminal
alias	Assign a name to a function
fg	Put a job into the background
bg	Bring a job to the foreground
set	Set a shell variable
setenv	Set an environment variable for this and subsequent shells
grep	Search for generalized regular expressions occurring in UNIX files
diff	Compare two files or directories and display the differences between the two
compress	Reduce the size of the named files by using adaptive Lempel-Ziv coding
uncompress	Restore files to their original state after they have been compressed using the UNIX compress utility
find	Recursively descend the directory hierarchy for each path seeking files that match a Boolean expression written in the predicate list
ftp	Transfer files from the user's account in one server to that in another or from one PC workstation to another
put	Copy specified files from your workstation to your account in the server
get	Copy specified files from your account in the server to your workstation
bye	Finish the file transfer process and return to MS-DOS/xterm/Terminal
telnet	Enable a computer to function as a terminal working from a remote computer

9.0 References

1. Byrd, J 1997, *A Basic UNIX Tutorial*, Department of Computing and Communications, Idaho State University. Viewed 09/12/03, <http://www.isu.edu/departments/comcom/workshops/unix/>
2. Campus Computing [Online] 1999, *The UNIX Cook Book*, Campus Computing, Department of Information & Access Technology Services, University of Missouri-Columbia. Viewed 09/12/03, <http://iatservices.missouri.edu/>
3. Department of Computer Science [Online] 2003, *Using UNIX on the Sun Rays*, Department Computer Science, The University of Adelaide. Viewed 09/12/03, <http://www.cs.adelaide.edu.au/for/students/handbook>
4. Fiamingo, FG, DeBula, L & Condron L 1998, *Introduction to UNIX*, University Technological Services, The Ohio State University Press, Ohio
5. SunService 1996, *Solaris 2.X System Administration Essentials; Student Guide*, Sun Microsystems, Inc, Palo Alto
6. Winsor, J 2000, *Solaris: System Administrator's Guide*, 3rd Guide, Sun Microsystem Press, Prentice Hall, Palo Alto

10.0 Index

A

B

C

D

E

F

G

H

I

K

L

M

O

P

R

S

T

U