UE17MC454
Unix and Shell Programming

Dr. S. Thenmozhi

Course Objectives

- To learn on essentials of shell programming
- Write simple Shell programs and manipulate system processes using shell scripting
- Automate many system tasks which may be part of complex scripts

Course Requisites

Text Book:

- Behrouz A. Forouzan, Richard F. Gillberg, UNIX and Shell Programming, Cengage Learning, Reprint 2011. (Unit I, Unit II , Unit IV)
- Venkateshmurthy, Introduction to UNIX & Shell Programming, Pearson Education, 2006. (Unit III , Unit V)

References:

- Sumitabha Das, UNIX Concepts and Applications, Tata McGraw-Hill, 3rd Edition, 2008.
- Eric Foster, John C. Welch, Micah Anderson, Beginning Shell Scripting, Wiley Publishing, 2005.

Evaluation

- Course – 3 credit course
- Course requirement
  - 60 marks for ISA
    - T1 -20 Marks, T2 – 20 Marks, Quiz – 10 Marks, Assignment – 10 Marks
  - 40 marks for ESA
  - No passing minimum

Course Outline

- Unit I : Introduction to Unix and File Systems
- Unit II : Introduction to Shells
- Unit III: Shell Programming
- Unit IV: Filters and Regular expressions
- Unit V: Awk

What are these???

What are these???

How does these flavours differ??

- Fundamental design
- Commands and features
- The hardware platform(s) (i.e., processors) for which they are intended
- Whether they are *proprietary software* (i.e., commercial software) or *free software*

Proprietary Flavours

- AIX - developed by IBM for use on its mainframe computers
- BSD/OS - a commercial version of BSD developed by Wind River for Intel processors
- HP-UX - developed by Hewlett-Packard for its HP 9000 series of business servers
- IRIX - developed by SGI for applications that use 3-D visualization and virtual reality
- QNX - a real time operating system developed by QNX Software Systems primarily for use in embedded systems
- Solaris - developed by Sun Microsystems for the SPARC platform and the most widely used proprietary flavor for web servers
- Tru64 - developed by Compaq for the Alpha processor

Free Flavours

- Linux - the most popular and fastest growing of all the Unix-like operating systems
- FreeBSD - the most popular of the BSD systems
- NetBSD - features the ability to run on more than 50 platforms, ranging from acorn26 to x68k
- OpenBSD - the most secure of all computer operating systems
- Darwin - the new version of BSD that serves as the core for the Mac OS X

Why Rapid Growth?

- Very low cost (i.e., free),
- The ability to operate on a wide range of platforms (everything from a supercomputer to a wristwatch)
- The availability of the source code so that it could be easily and freely modified and improved by its users and
- Greater ease of use.

Linux Vs Unix

Linux	Unix
The source code of Linux is freely available to it's users.	The source code of Unix is not available for the general public
Linux primarily uses Graphical User Interface with an optional Command Line Interface	Unix primarily uses Command Line Interface.

Linux Vs Unix

Linux	Unix
Linux is very flexible and can be installed on most of the home based PCs.	Unix has a rigid environment of the hardware. Hence, cannot be installed on every other machine.
Linux is used on home based PCs, Mobile Phones, Desktops	Unix is mainly used in Server Systems, Mainframes and High End Computers.

Unit 1

Computer System

- Computer System has Hardware and Software
- Hardware: Devices that can be recognized by touch
- Input, output, CPU, Secondary storage devices
- Software: combination of programs written to make computer a multipurpose machine
  - Application software: software written to solve users problems
  - System Software: Consists of set of programs that serve computer/ support computer process
- Operating System is a system software that manages the computer

Why Unix?

- **Portable** – can be portable to different environments
- **Multiuser** – supports multiple users concurrently share hardware and software
- **Multitasking** – run more than one job at a time
- **Networking** – users can operate the system remotely
- **Organized File System** – Everything is treated as a File and are organised in a well defined structure
- **Device Independence** – input and output devices are treated as ordinary files
- **Utilities** – utility programs to operate easily
- **Services** – system administrators provided with special utilities

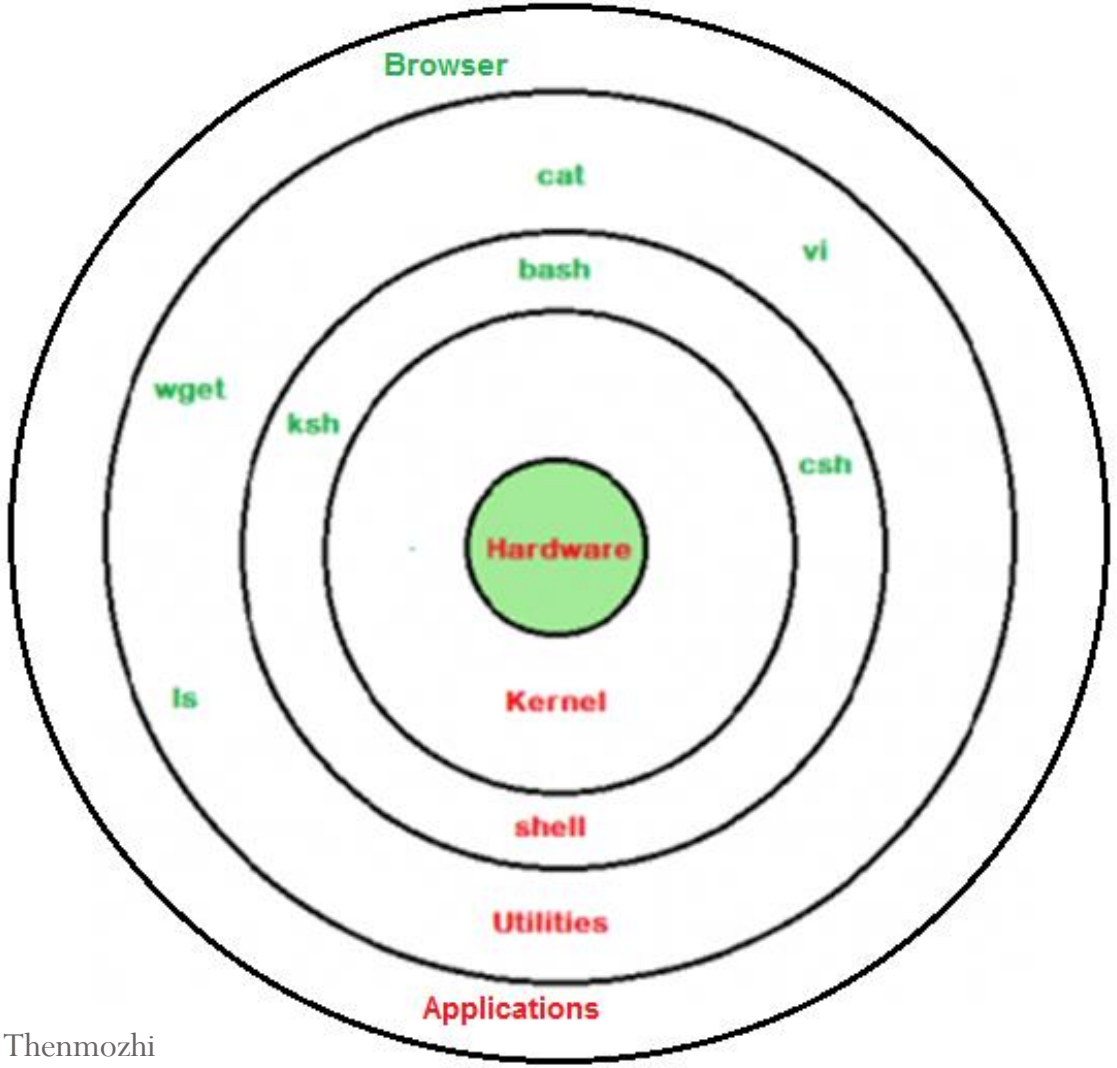
Unix Environment

- Unix is a multiuser, multiprocessing, portable OS. It is designed to facilitate Programming, text processing and communication.
- **Personal Environment**
  - Although Unix supports multiuser environment, many install in personal computers. Hence in mid 1990's Linux, a free Unix system was introduced.
  - Apple system X released in 2001, has Unix as its kernel.

Unix Environment

- **Time-Sharing Environment**
  - Many users are connected to one or more computers.
  - All computing is done in central computer
  - Their terminals are nonprogrammable.
  - The output devices and auxiliary storage devices are shared by all users
- **Client/Server Environment**
  - Splits the computing function between a central computer and user computer
  - Users computer – client
  - Central Computer – server
  - Work is shared between Client and Server. Computation responsibility moved to Server

Unix Structure



- Four Major Components
  - Kernel
  - Shell
  - Utilities
  - Application Programs
- **Kernel**
  - Heart of the Unix System
  - Process Control and Resource Management
  - System calls to perform services

- **Shell**
  - Part that is visible to the user
  - Receives and interprets the command entered by the user
  - If the command requires a utility, it requests the kernel to execute the utility
  - If it requires to execute an application program, it requests kernel to execute the application program
  - Major parts of shell – Command interpreter and Shell programming
  - A shell script is a file that contains shell commands that perform a useful function. It is also known as shell program.
  - Bourne Shell – Steve Bourne at AT&T Labs – Oldest Shell –Enhanced version – Bourne again shell(Bash) used in Linux.
  - C Shell – Bill Joy at Berkeley – Commands look like C statements. Tcsh is compatible in Linux
  - Korn Shell – David Korn at AT&T labs – compatible with Bourne Shell

- **Utilities**
  - Utility is a standard Unix program that provides a support process for users.
  - Common utilities – text editors, Search Programs and Sort Programs
  - Larger utilities – text editors – vi, emacs, pico
  - Simple utilities – ls etc..
- **Applications**
  - Not a standard part of Unix System
  - Extended capability to the system
  - Utilities started as application and they are now part of the system

Common Commands

- **Date and Time – date**
  - GMT – Greenwich Mean Time
  - PST – Pacific standard Time
  - IST – Indian Standard Time
  - UST – Universal Std. Time

Usage: $date –u (GMT/UST)

$date "+Today's date is: %D. The Time is: %T"

Options: D – Date in mm/dd/yy

T - hour:minute:second

Common Commands

- **Who's Online – Who**

  Options - -u –how long online

  -H – with header that explains each column

  Whoami – returns your userid
- **Calendar - cal**
- **Change Password - passwd**
- **Print Message – echo**
- **Online Documentation – man**

  Option: -k Search on Keyword usage: man –k searchkeyword
- **Print – lpr**

  Usage: lpr file1 file2

  Option : -P (printer) Usage: lpr –P lp1 file1 file2

Other Useful Commands

- **Show the name of the terminal – tty**
  - Each Terminal is treated as a file
  - Name of the terminal is the name of the file
  - Usage: tty
- **Clear the terminal - clear**
- **Calculator – bc**
- **System name – uname**

Other Useful Commands

- **Set Terminal – stty**
  - Sets/unsets the input/output options of selected terminal
  - When terminal is not responding properly, stty can be used to reconfigure it
  - Usage: stty – Current common setting
  - Option: -a current terminal option settings

    -g displays in a format
  - Attribute and default values : erase ^h,

    eof ^d,

    kill ^u,

    intr ^c
  - Usage: stty eof ^x

- **Record Session – script command**

  Usage: script default file is **typescript**

  script <yourfilename>

  exit to end your recording

  cat <yourfilename> -- view contents

  lpr <yourfilename> -- to print the recorded session

  Option - –a <yourfilename> -- to append the same script

Filenames

- **File:** A named Collection of related data stored on an auxiliary storage device
  - Start with an alphabet
  - Use dividers to separate parts of name (_ , -,.)
  - Use an extension at the end of the filename (optional, but preferable for language programs)
  - Never start filename with a period(.) – because it is recognized as hidden file.
  - Some versions allow 14 chars some 255 characters
- **Wildcards:** token that specify one or more different characters to make a specific request
  - ? – single char * - multiple character [ ] – group of chars

File Types

- Regular
- Directory
- Character Special
- Block Special
- Symbolic Link
- FIFO
- Socket

File Type Designators

File Type	File Identification Symbol
Regular File	- (hyphen)
Directory File	d
Character Special	c
Block Special	b
Symbolic Link	l
Fifo	p
Socket File	s

Regular file

- It may be text or binary file
- Both the files are executable provided execution rights are set
- They can be read or written by users with appropriate permissions
- To Create: touch or cat command
- To remove: rm command

Directory file

- Folder that contains other files and subdirectories
- Provides a means to organize files in hierarchical structure
- To create : mkdir command
- To remove : rmdir command
- Root directory – highest level in hierarchy - /
- Home Directory – Beginning of the personal directory structure
  - Directory when we first log into system
  - Each user has a home directory
  - Represented by ~
- Working Directory
  - The directory which we are currently working
  - Represented by .

Directory file

- Parent Directory – immediate above the present working directory, Represented by .. (double dot)
- Paths and Pathnames
  - Path – route map to fetch the file
  - Pathname – Absolute and Relative pathnames
  - Absolute: full path from the root to the desired directory
  - Relative: Path from the working directory to the file

Device file

- **Block device file**

  Physical device which transmits data a block at a time

EX: **hard disk drive, floppy disk drive**

- **Character device file**

  Physical device which transmits data in a character-based manner

EX: **line printers, modems, consoles**

- To create : mknod command

  mknod /dev/cdsk c 115 5

  /dev/cdsk : name of the device

  c -- character device b -- block device

  115 — major device number

  5 — minor device number

Major device number : an index to the kernel's file table that contains address of all device drivers

Minor device number : indicates the type of physical file and I/O buffering scheme used for data transfer

FIFO file

- Special pipe device file which provides a temporary buffer for two or more processes to communicate by writing data to and reading data from the buffer
- Each Fifo file will have Max size of buffer
- The storage is temporary
- The file exists as long as there is one process in direct connection to the fifo for data access

- To create : mkfifo OR mknod

  mkfifo /usr/prog/fifo _pipe

  mknod -p /usr/prog/fifo_pipe

- A fifo file can be removed like any other regular file

Symbolic link file

- A symbolic link file contains a pathname which references another file in either the local or a remote file system
- To create : ln command with –s option

ln –s /usr/abc/original /usr/xyz/slink

cat /usr/xyz/slink

/*will print contents of /usr/abc/original file*/

Hard and symbolic links

- A hard link is a UNIX path name for a file
- To create hard link ln command is used

  ln /usr/abc/old.c /usr/xyz/new.c
- Symbolic link is also a means of referencing a file
- To create symbolic link ln command is used with option –s

  ln –s /usr/abc/old.c /usr/xyz/new.c

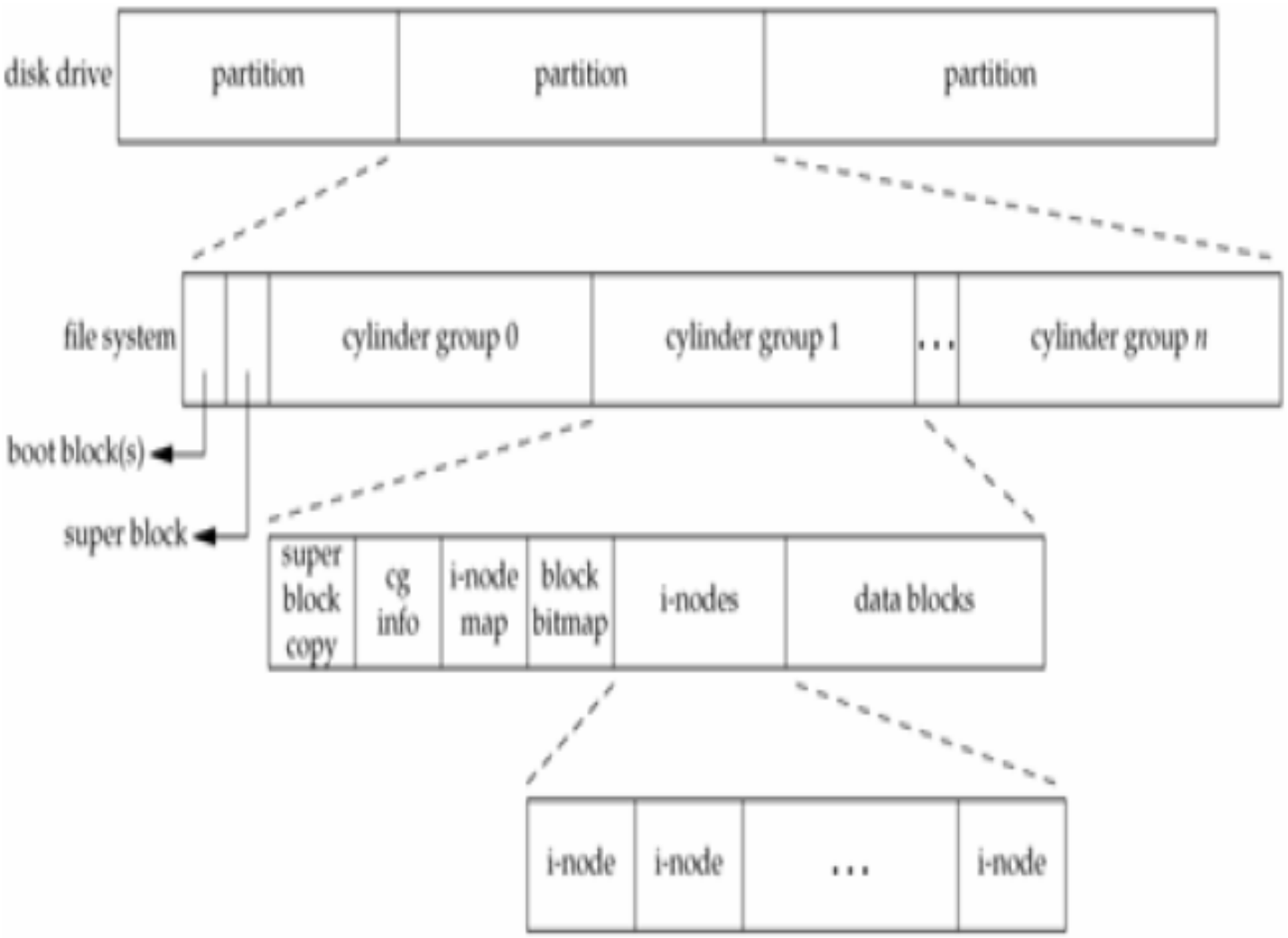
Difference : hard link and symbolic link

Hard link	Symbolic link
Does not create a new inode	Create a new inode
Cannot link directories unless it is done by root	Can link directories
Cannot link files across file systems	Can link files across file system
Increase hard link count of the linked inode	Does not change hard link count of the linked inode

File System

- A file system is a **logical collection of files on a partition or disk**. A disk drive can be divided into one or more partitions. Each partition can contain a file system.
- A file system is organized in the form of a directory structure with its own root
- Before any disk partition can be used, a file system must be built on it.
- File system is data structures written to disk that will be used to access and organize the physical disk space into files
- UFS - traditional BSD-derived UNIX file system
- PCFS - a file system to read and write DOS-formatted diskettes
- HSFS(High Sierra)/CD9660 - a file system to read CD file systems.
- NFS – a file system to enhance networked systems

- Every file system has four components
- **Boot block** – This block contains a small boot program and the partition table
- **Superblock** – It contains global information about the file system. It is the balance sheet of every file system. It consists of free list of i-nodes and data blocks that can be immediately allocated for creating a file
- **Inode block** – This region contains the i-node for every file of the FS. i-node contains file type, file access permission bits, size, pointers to file data blocks etc.
- **Data block** – all data and programs created by users reside in this area


disk drive
partition
partition
partition
file system
cylinder group 0
cylinder group 1
...
cylinder group n
boot block(s)
super block
super block copy
cg info
i-node map
block bitmap
i-nodes
data blocks
i-node
i-node
...
i-node

Operations Unique to Directories

- **Locate directory – pwd command**
  - You can locate your present working directory
  - No options and no attributes
  - Usage: pwd

Operations Unique to Directories

- **List Directory – ls command**
  - Lists the contents in a directory
  - Depending on options it can list files, directories or sub directories
  - pwd contents displayed – if no options and no attributes
  - Some Common options

-l long list

-d Working directory

-n user/group ids

-r reverse order

-t time sequence

-u last access

-i print inode

-p identify directories

-R recursive

-a list all

-ld long list directories

-nd list user and group ids

Operations Unique to Directories

- **Change Directory – cd command**
  - To change the working directory
- **Remove directory – rmdir command**
  - Deletes the directory
  - Can delete when the directory is empty
  - -r recursive –f forcible deletion

Operations Unique to Regular Files

- **Create File – vi, cat, touch**
- **Edit File – sed**
- **Display File – more**
  - -c clears screen before displaying
  - -f wrap long lines
  - -s squeezes multiple blank lines
  - -lines sets the number of lines in the screen
  - +nmbr starts output at the indicated line number
  - +/pattern Locates the first occurrence of the pattern and starts two lines before it
- **Print File**

Operations Common to Both

- **Copy – cp command**
  - -p preserve time and permission
  - -i interactive prompt
  - -r recursive (copy subdirectories)
- **Move – mv command**
  - -f force
  - -i interactive
- **Link – ln command**
  - -s symbolic
  - -f force
  - -i interactive

Operations Common to Both

- **Remove – rm command**
  - -f force
  - -r recursive
  - Wild card remove eg: rm –r f*
- **Find file – find command**
  - Search and locate files and directories based on conditions you specify
  - Find files by permissions, users, groups, file type , date and size

Finding Files with names

- Finding files using name in current directory
  - Eg: find . –name wel.txt
- Finding files using name in home directory
  - Eg: find /home/thenmozhi –name wel.txt
- Finding files using name and ignoring case
  - Eg: find /home/thenmozhi –iname wel.txt
- Finding directories using name
  - Find / -type –name assignment
- Finding all files in a directory
  - Eg: find . –type f –name “*.php”

Finding files with permissions

- Finding files with 777 permissions
  - Eg: find . –type f –perm 0777 –print
- Find files without 777 permissions
  - Find / -type f !perm 0777

Find files based on date and time

- Find last 50 days modified files
  - Eg: find / -mtime 50
- Find last 50 days accessed files
  - Eg: find / -atime 50
- Find last 50-100 days modified files
  - Eg: find / -mtime +50 –mtime -100
- Find changed files in last 1 Hour
  - Eg: find / -cmin -60
- Find modified files in last 1 Hour
  - Eg: find / -mmin -60 [accessed files use –amin]

Find Files based on size

- Finding all 50 MB files
  - Eg: find / -size 50M
- Find and delete 100MB files
  - Eg: find / -size +100M –exec rm –rf {} \;
- Find specific files and delete
  - Eg: find /-type f –name *.mp3 –size +10M –exec rm {} \;

Find files based on user and group

- Finding all files based on user
  - Eg: find /home –user student
- Finding all files based on group
  - Eg: find /home –group student
- Fidning particular files of user
  - Eg: find /home –user student –iname “*.txt”