Internal Assessment Test 1 – Sept. 2017

Scheme and Solutions

Sub:	UNIX Shell Programming					Sub Code:	15CS35	Branch: CSE &ISE
Date:	21-09-2017	Duration:	90 min's	Max Marks:	50	Sem / Sec:	CSE(3A,B,C) & ISE(3A,3B)	OBE

1 (a) Explain UNIX architecture with a neat diagram. [7]

The marks split up is as follows :

- Unix Architecture diagram – 3 Marks
- Division of Labour: the kernel & shell – 2 Marks
- The file system – 1 Mark
- System call – 1 Mark

Solution :

Unix Architecture Diagram

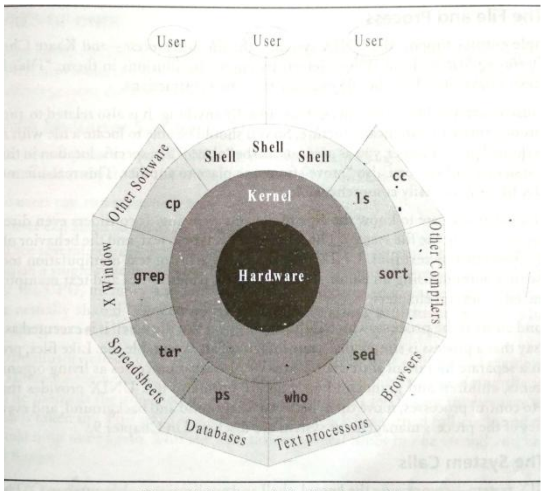


Fig. 2.1 The Kernel-Shell Relationship

Division of Labor: Kernel and Shell

Kernel

- Core of the Operating System. Interacts with the machines hardware.
- Collections of routines mostly written in C.
- It's loaded onto memory on system boot.
- User programs that need to access the hardware use the services of the kernel via use of system calls and the kernel performs the job on behalf of the user.
- Kernel is also responsible for managing system's memory, schedules processes, decides their priorities.

Shell

- It's an interface between the user and the kernel.
- Shell interacts through a set of functions called as system calls.
- Shell uses the services of the Kernel to perform jobs of the user.

- Even though there is only one kernel running on the system, there could be several shells in action.
- The shell is responsible for interpreting the meaning of meta characters if any, found on the command line before dispatching the command to the kernel for execution.

```
Ex: $echo Sun          Solaris
        Sun Solaris
```

  In the above example when the user enters echo with multiple spaces in between the input,the shell compresses all multiple spaces into single one. The echo command then runs with spaces compressed as above.

<u>The File and processes</u>

File

- Files have places and processes have life.
- File is just an array of bytes and can contain virtually anything.
- UNIX doesn't care to know the type of file you are using. It considers directories and devices as members of the file system.

Process

- File when it is executed as a program. "Time image" of an executable file.
- Processes also have a hierarchical tree structure *(Child parent relationship).*

<u>The system Calls *(read, write …).*</u>

- Handful of "C" functions which are used by shell and applications used to communicate with the kernel.
- All UNIX flavors have one thing in common *"They use the same system calls".*
- System calls are described in the POSIX specification.
- Linux uses the same system calls hence *"Linux is UNIX".*
- *"Write", "read"* is some of the systems calls used over all files.

(b) Explain the commands used to add, modify and delete a user in UNIX environment. [3]

- Add a user with ex – 1 Mark
- Delete a user with ex – 1 Mark
- Modify user with ex – 1 Mark

Solution : Useradd : adding a user

```
Ex: useradd –u 210 –g dba –c "THE RDBMS" –d /home/oracle –s /bin/ksh
                    –m oracle
```

The above ex creates a user oracle with UID 210 and group name dba.The home dir is /home/oracle and the user uses korn shell

- Usermod : Modifying users

  Ex :usermod –s /bin/bash oracle : sets bash as current shell for the user oracle
- Userdel : removing users

  Ex: userdel oracle : removes the user oracle

2 (a) Explain the salient features of UNIX operating system. [6]

Any six features of UNIX like multitasking, multiuser, programming tool kit, portable etc. Each carry 1 Mark

Solution :

1. Multiuser System
   - UNIX is a multiprogramming system. UNIX allows multiple users to concurrently share h/w and s/w.
2. Multitasking System
   - UNIX allows a user to run more than one program at a time.In fact more than one program will be running in background while the user is working in foreground.
3. *Building Block Approach(Solving huge problem combining small solutions)*
   - *Many UNIX tools are designed with the requirement that the output of one tool will be the input to other.UNIX makes this done through pipes and filters.*
     Ex : ls|wc
4. UNIX Toolkit *(Utilities)*
   - UNIX provides general purpose tools,text manipulation utilities(filters),compilers and interpreters, networked applications and system administrator tools.
5. Pattern Matching
   - Meta characters (*), regular expressions are supported.
6. Programming Facility
   - Shell programming, Supports a huge collection of compilers.
7. Documentation
   - Commands are well documented *(man page)*.

(b) Explain UNIX command structure in brief with an example.

Command structure :

- Verb -1 Mark
- Options - 2 Marks
- Arguments – 1 Mark

Solution :

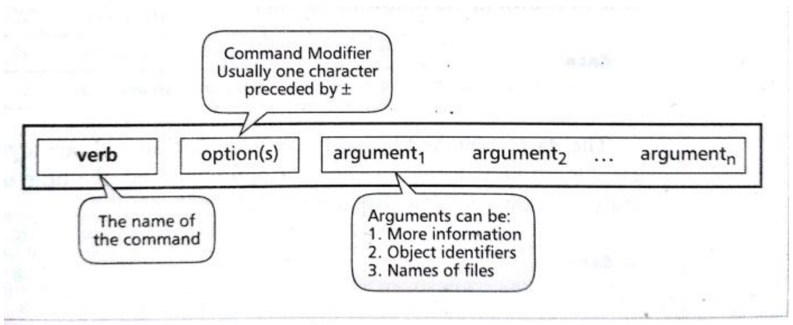


FIGURE 1.10 *General Command Format*

- Command, option/s or argument/s should be separated by at least one space/tab to enable the system to interpret them as words. We can have any number of whitespaces *(string of spaces).*
- Commands are case sensitive
- Options are special type of arguments that prefixed with a "-". There meaning is predetermined.
- Arguments are not pre determined and their need is command specific. They can range for 0 to n numbers.

```
$ls –l
```
ls is the command –l is the options.
```
$echo hello world
```
echo is the command and hello & world are arguments.

3 (a)	Explain UNIX **man** page with an example. [6]
	• Man page description – 6Marks

Solution :

- UNIX offers an online help facility in the man command.

```
$man ls     //Displays manual page of "ls" command
```

- Man page is divided into a number of compulsory and optional sections
- NAME, SYNOPIS, DESCRIPTION are seen in all man pages.

```
Example: $man wc
```

NAME

wc - print the number of bytes, words, and lines in files

SYNOPSIS

wc [-c | -m | -C] [-lw] [file ...]

DESCRIPTION

The wc utility reads one or more input files and, by default, writes the number of newline characters, words and bytes contained in each input file to the standard output. The utility also writes a total count for all named files, if more than one input file is specified.

OPTIONS

The following options are supported:
-c Count bytes.
-m Count characters.
-C same as –m.
-l Count lines.
-w Count words delimited by white spaces or new line characters ...

OPERANDS
The following operand is supported:
file A path name of an input file. If no file operands are specified, the standard input will be used.

EXIT STATUS
See largefile(5) for the description of the behavior of wc when encountering files greater than or equal to 2 Gbyte (2 **31 bytes)

SEE ALSO
cksum(1), isspace(3C), iswalpha(3C), iswspace(3C), largefile(5), .

(b) Briefly explain about internal and external commands in UNIX. Which command is used for identifying them and how? [4]

- Internal & External Command with ex – 3 Marks
- Type command - 1 Mark

Solution :
External command: Program or file having independent existence in "/bin or /usr/bin" directory is called external commands.
Ex. ls, ps etc.
Internal command: shell built-in commands that are not stored as separate files are called internal commands.
Ex. echo
Command "type": can be used to check whether the command is internal or external.
Syntax: **`$type echo`**

```
echo is a shell builtin
```

4 (a) Explain how to display and set terminal characteristics in UNIX OS. [5]

- Stty command – 1 Mark
- Displaying terminal settings – 2 Marks
- Changing the Settings – 2 Marks

Solution :
stty command is used to display and change terminal settings in UNIX.
Displaying terminal settings :

```
$ stty –a
        intr = ^c; quit = ^\; erase = ^?; kill = ^u;
        eof = ^d; eol = <undef>; swtch = <undef>;
        start = ^q; stop = ^s; susp = ^z;
```

The output is in the form keyword=value.The setting intr = ^c signifies [ctrl-c] interrupts a program. The erase character is [ctrl-h] and the kill character is [ctrl-u]
Changing the Settings :

- changing the interrupt key(intr): if we want to use [ctrl-c] as interrupt instead of [delete] use stty intr \^c - this indicates that interrupt key is ctrl-c

- Changing the End of file key(eof) : Instead of [ctrl-d] if we want to change [ctrl-a] as eof char then: stty eof \^a
- Turning off the keyboard input :
  Stty –echo turns off the keyboard input

(b) Explain the contents of /etc/passwd and /etc/shadow file with respect to UNIX OS. [5]

- /etc/passwd : 3 Marks
- /etc/shadow – 2 Marks

Solution :

/etc/passwd : All user information except the password encryption is stored in /etc/passwd. It contains seven fields as listed below.

Ex : useradd –u 210 –g 241 dba –c "THE RDBMS" –d /home/oracle –s /bin/ksh –m oracle

- Username : oracle
- Password : contains x
- UID : 210
- GID : 241
- Comment : Message as THE RDBMS
- Home dir : /home/oracle
- Login shell : /bin/ksh

/etc/shadow : For every line in /etc/passwd there is a corresponding entry in /etc/shadow.Encrypted password will be shown here in the second field.

Oracle:PRihghhDR:12032::::::

5(a) Explain the following commands with an example; [5]
i) echo ii) who iii) date iv) ls v) bc

- echo command with ex
- who command with ex
- date command with ex
- ls command with ex
- bc command with ex

(b) Explain various environments supported by UNIX with a neat diagram. [5]

- Personal Environment – 1 Mark
- Time sharing environment – 2 Marks
- Client Server Environment – 2 Marks

Solution :

Personal Environment – UNIX can be used in a standalone machine with personal setup.

Time-Sharing Environment – Thin client environment. Users have terminals which connect to workstation for the job execution *(Multiuser environment).*

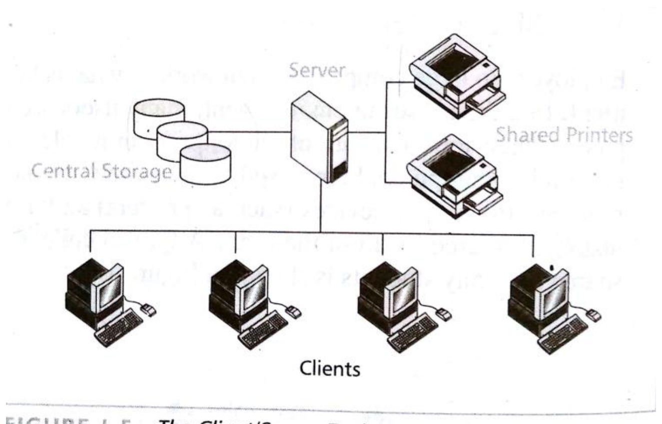


FIGURE 1.5 *The Client/Server Environment*

Client/Server Environment – Workstations *(Client)* connect with mainframe machines *(Server)* for high computational needs over network.

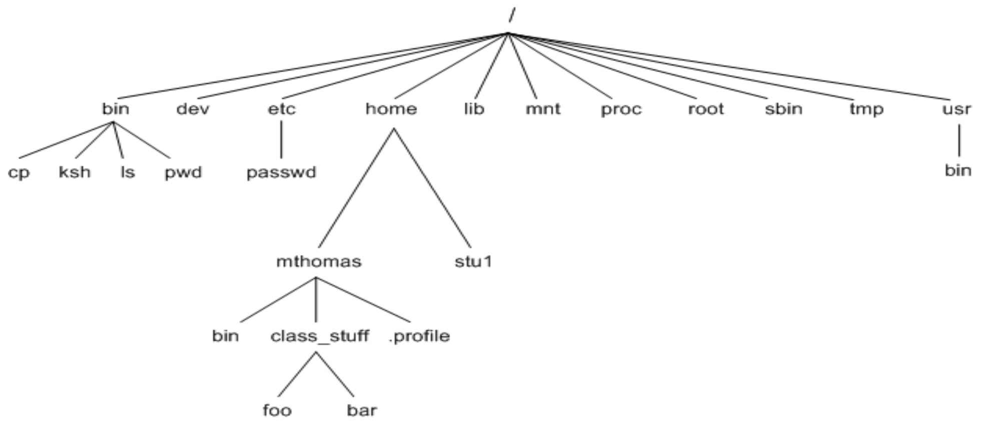
6(a) Explain parent-child relationship in UNIX file system with a neat diagram. [5]

- Parent child relationship diagram – 1 Marks
- Explanation of different layers – 4 Marks

Solution :

A file is the smallest unit of storage in the Unix file system. The Unix file system has a hierarchical (or tree-like) structure with its highest level directory called root (denoted by /, pronounced slash). Immediately below the root level directory are several subdirectories, most of which contain system files. Below this can exist system files, application files, and/or user data files.

.



The following system files (i.e. directories) are present in most Unix file systems:

bin - short for binaries, this is the directory where many commonly used executable commands reside

- dev - contains device specific files
- etc - contains system configuration files
- home - contains user directories and files
- lib - contains all library files
- mnt - contains device files related to mounted devices

• proc - contains files related to system processes
• root - the root users' home directory (note this is different than /)
• sbin - system binary files reside here. If there is no sbin directory on your system, these files most likely reside in etc
• tmp - storage for temporary files which are periodically removed from the file system
• usr - also contains executable commands

(b) Explain different file types available in UNIX Operating System. [5]

- Ordinary file - 2 Marks
- Directory file - 2 Marks
- Device File - 1 Mark

Solution :

Ordinary (Regular) File :
This is the most common file type. An ordinary file can be either a text file or a binary file.
A text file contains only printable characters and you can view and edit them. All C and Java program sources, shell scripts are text files. Every line of a text file is terminated with the newline character.
A binary file, on the other hand, contains both printable and nonprintable characters that cover the entire ASCII range. The object code and executables that you produce by compiling C programs are binary files. Sound and video files are also binary files.

Directory File :
A directory contains no data, but keeps details of the files and subdirectories that it contains. A directory file contains one entry for every file and subdirectory that it houses. Each entry has two components namely, the filename and a unique identification number of the file or directory (called the inode number).
When you create or remove a file, the kernel automatically updates its corresponding directory by adding or removing the entry (filename and inode number) associated with the file.

Device File :
All the operations on the devices are performed by reading or writing the file representing the device. It is advantageous to treat devices as files as some of the commands used to access an ordinary file can be used with device files as well.

7(a) Describe all the field of **ls -l** command output. [5]

- Ls -l command expl with ex along with 7 fields of long format – 5 Mark

(b) Explain the following commands with examples. [5]
i) cat ii) mv iii) rm iv) cp v) wc

- cat command with ex – 1 Mark
- mv command with ex – 1 Mark

- rm command with ex – 1 Mark
- cp command with ex – 1 Mark
- wc command with ex – 1 Mark

8 (a) Explain the following commands with examples. [6]
i) HOME ii) cd iii) pwd iv) mkdir v) rmdir vi) file

- Explanation of all the commands with examples – 6 Marks

(b) Differentiate between absolute and relative pathnames with suitable example.[4]

- Absolute Path explanation with example – 2 Marks
- Relative Path explanation with examples – 2 Marks

Solution : Absolute Path: An *absolute path* is defined as specifying the location of a file or directory from the root directory(/). In other words we can say *absolute path* is a complete path from start of actual file system from / directory.

example:
/home/user/Document/srv.txt
/root/data/dev.jpg
/var/log/messages
All are absolute Path.

Relative Path: ***Relative path*** is defined as path related to the present working directory(pwd). Suppose I am located in /home/user1 and I want to change directory to /home/user1/Documents. I can use ***relative path*** concept to change directory to Documents.

example:
here are two examples for changing directory, 1st by using relative path, 2nd by using absolute path.

```
$ pwd
/home/user1
$cd Documents/  (using relative path)
$pwd
/home/user1/Documents
```

9(a) Consider that the file **foo.txt** has permissions rw-r-xr--. Specify **chmod** expression in relative as well as absolute methods to achieve the following file permissions. [5]
i) rwxrwxrwx ii)r- -r - - - - - iii)- - - r - - r - - iv)r - - - w - - - x
v)- - - - - - - - -

i)chmod u+x,g+w,o+wx foo.txt   chmod 777 foo.txt

ii) chmod u-w, g-x, o-r foo.txt   chmod 440 foo.txt

iii) chmod u-rw, g-x foo.txt,   chmod 044 foo.txt iv)chmod u-w,g-rx+w o-r+x foo.txt   chmod 421 foo.txt

v) chmod u-rw,g-rx,o-r foo.txt   or chmod u-rw,g=w,o=x foo.txt   chmod 000 foo.txt

(b) Differentiate between the following commands output.

i) cat foo , cat > foo   ii) echo "$SHELL" , echo '$SHELL'

ii) $date +"%h %m", $date +"%a %A"   iv) ls –ld, ls –l

v) cd ~myfile , cd ~/myfile

Differentiating all the commands – 1*5 -5M

i) Cat foo – displays contents of foo
   Cat > foo – creates a file foo and save the contents into it.
ii) echo "$SHELL" – displays current shell
   echo '$SHELL' – displays $SHELL
iii) $date +"%h %m" – displays month name and number like Sep 09
   $date +"%a %A - displays weekday name abbreviated and full weekday name like Mon Monday
iv) ls –ld : displays long listing of directories
   ls –l : displays long listing of files
v) cd ~myfile – changes current directory to myfile
   cd ~/myfile – changes current directory to myfile home dir.

10a) What is the significance of following commands output in UNIX?
i) cd ../.. ii) mkdir ../bin   iii) rmdir ..   iv)ls ..   v) mkdir a/b/c   vi) cp –r bar1 bar2

---

i) Change the directory to root
ii) Change the directory from bin to root
iii) Remove the directories under root
iv) Lists all the files under root
v) Error. mkdir a b c is the correct one.
vi) Recursively copies bar1 to bar2

(b) Which is the command used for changing file ownership and group ownership in UNIX? Elaborate with an example.   [4]

- Changing file ownership – 2 Marks
- Changing group ownership – 2 Marks

Solution : Changing File Ownership :
chown :Changing ownership requires superuser permission, so use su command

```
ls -l note  -rwxr----x  1  kumar  metal  347  may 10 20:30  note
chown sharma note;
ls -l note
-rwxr----x  1  sharma  metal 347 may 10 20:30 note
```

Once ownership of the file has been given away to sharma, the user file permissions that previously applied to Kumar now apply to sharma. Thus, Kumar can no longer edit note since there is no write privilege for group and others. He can not get back the ownership either. But he can copy the file to his own directory, in which case he becomes the owner of the copy.

chgrp : This command changes the file’s group owner. No superuser permission is required.

```
ls –l dept.lst
-rw-r--r--  1  kumar  metal  139  jun 8 16:43 dept.lst
chgrp dba dept.lst;
ls –l dept.lst
-rw-r--r-- 1 kumar dba 139 jun 8 16:43 dept.lst
```