First mid Term Exam –Sep. 2017

Class: 3 IT　　　　　　　　Subject: UNIX Shell Programming
Time: 1 Hour　　　　　　　　Marks: 10
Attend any two questions. All question carry equal marks.

Q.1: What is access permissions in linux ? Explain with example.

Answer: On Linux and other Unix-like operating systems, there is a set of rules for each file which defines who can access that file, and how they can access it. These rules are called file permissions or file modes. The command name chmod stands for "change mode", and it is used to define the way a file can be accessed.

In general, chmod commands take the form:

```
chmod options permissions file name
```

If no options are specified, chmod modifies the permissions of the file specified by file name to the permissions specified by permissions.
permissions defines the permissions for the owner of the file (the "user"), members of the group who owns the file (the "group"), and anyone else ("others"). There are two ways to represent these permissions: with symbols (alphanumeric characters), or with octal numbers (the digits 0 through 7).
Let's say you are the owner of a file named myfile, and you want to set its permissions so that:

1. the user can read, write, and execute it;
2. members of your group can read and execute it; and
3. others may only read it.

This command will do the trick:

```
chmod u=rwx,g=rx,o=r myfile
```

This example uses symbolic permissions notation. The letters u, g, and o stand for "user", "group", and "other". The equals sign ("=") means "set the permissions exactly like this," and the letters "r", "w", and "x" stand for "read", "write", and "execute", respectively. The commas separate the different classes of permissions, and there are no spaces in between them.
Here is the equivalent command using octal permissions notation:

```
chmod 754 myfile
```

Here the digits 7, 5, and 4 each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers 4, 2, 1, and 0:

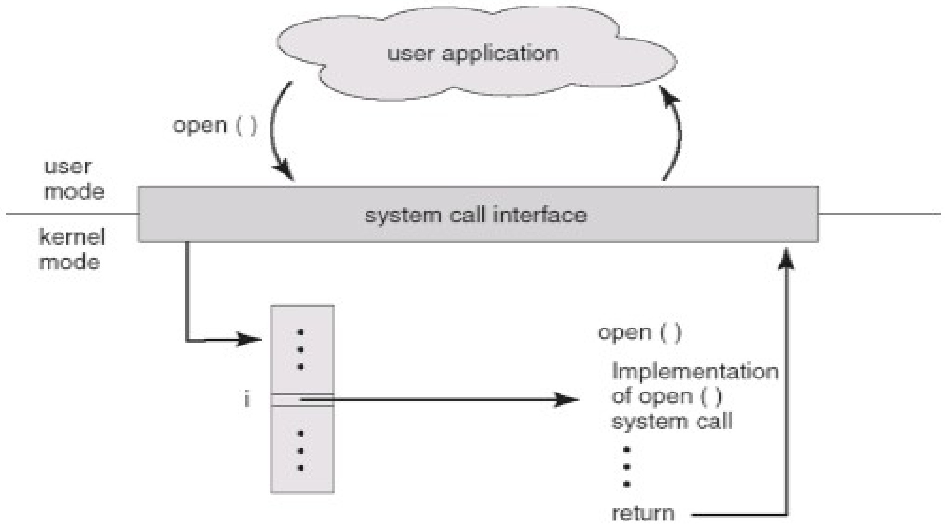
- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute", and
- 0 stands for "no permission."

So 7 is the combination of permissions 4+2+1 (read, write, and execute), 5 is 4+0+1(read, no write, and execute), and 4 is 4+0+0 (read, no write, and no execute).

Q.2: Explain the system call with example .

Answer - a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. System calls provide an essential interface between a process and the operating system.

- or example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.



Types of System calls

- Process control
- File management

- Device management
- Information maintenance
- Communications

1) Process Control:

- A running program needs to be able to stop execution either normally or abnormally.
- When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger.
- Following are functions of process control:

  i. end, abort

  ii. load, execute

  iii. create process, terminate process

  iv. get process attributes, set process attributes

  v. wait for time

  vi. wait event, signal event

  vii. allocate and free memory

2) File management :

- We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes.
- Once the file is created, we need to open it and to use it. We may also read, write, or reposition. Finally, we need to close the file, indicating that we are no longer using it.
- We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system.
- In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary. File attributes include the file name, a file type, protection codes, accounting information, and so on

  Functions:

  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes

3) Device Management:

- A process may need several resources to execute - main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available.

- The various resources controlled by the OS can be thought of as devices. Some of these devices are physical devices (for example, tapes), while others can be thought of as abstract or virtual devices (for example, files).
- Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with files.
- In fact, the similarity between I/O devices and files is so great that many OSs, including UNIX, merge the two into a combined file-device structure.
- A set of system calls is used on files and devices. Sometimes, 1/0 devices are identified by special file names, directory placement, or file attributes.

  Functions:

  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

Information Maintenance

- Many system calls exist simply for the purpose of transferring information between the user program and the OS. For example, most systems have a system call to return the current time and date.
- Other system calls may return information about the system, such as the number of current users, the version number of the OS, the amount of free memory or disk space, and so on.
- In addition, the OS keeps information about all its processes, and system calls are used to access this information. Generally, calls are also used to reset the process information.

Functions:

- get time or date, set time or date
- get system data, set system data
- get and set process, file, or device attributes

Communication

- There are two common models of interprocess communication: the message-passing model and the shared-memory model. In the message-passing model, the communicating processes exchange messages with one another to transfer information.
- In the shared-memory model, processes use shared memory creates and shared memory attaches system calls to create and gain access to regions of memory owned by other processes.
- Recall that, normally, the OS tries to prevent one process from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas.

- Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement than is shared memory for intercomputer communication.
- Shared memory allows maximum speed and convenience of communication, since it can be done at memory speeds when it takes place within a computer. Problems exist, however, in the areas of protection and synchronization between the processes sharing memory.

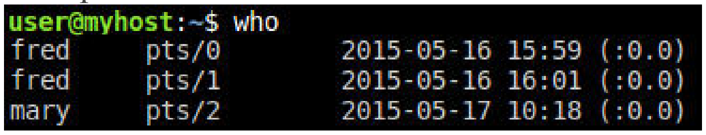
  Functions:

  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - Attach and Detach remote devices
  -

Q.3: Explain the following commands:

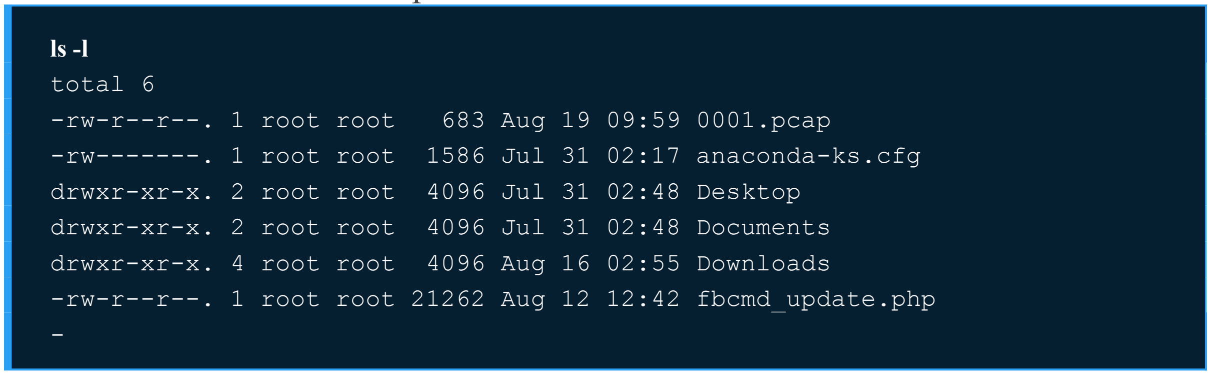
a) who : The who command prints information about all users who are currently logged in.

Displays the username, line, and time of all currently logged-in sessions. For example:



```
user@myhost:~$ who
fred     pts/0        2015-05-16 15:59 (:0.0)
fred     pts/1        2015-05-16 16:01 (:0.0)
mary     pts/2        2015-05-17 10:18 (:0.0)
```

b) ls-l shows file or directory, size, modified date and time, file or folder name and owner of file and it's permission.



```
ls -l
total 6
-rw-r--r--. 1 root root   683 Aug 19 09:59 0001.pcap
-rw-------. 1 root root  1586 Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Documents
drwxr-xr-x. 4 root root  4096 Aug 16 02:55 Downloads
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
-
```

c) *pwd* - print working directory, is a Linux command to get the current working directory.

```
$ pwd
/user/src
```

d) **grep**- which stands for "global regular expression print," processes text line by line and prints any lines which match a specified pattern.

Grep is a powerful tool for matching a regular expression against text in a file, multiple files, or a stream of input. It searches for the *PATTERN* of text that you specify on the command line, and outputs the results for you.

```
grep chope /etc/passwd
```

Search **/etc/passwd** for user **chope**.

```
grep "May 31 03" /etc/httpd/logs/error_log
```

Search the Apache error_log file for any error entries that happened on May 31st at 3AM.

Second mid Term Exam – Nov.2017

Class: 3 IT Subject: UNIX Shell Programming
Time: 1 Hour Marks: 10
Attend any two questions. All question carry equal marks.

Q.1: What is Shell? Explain the various type of shell.

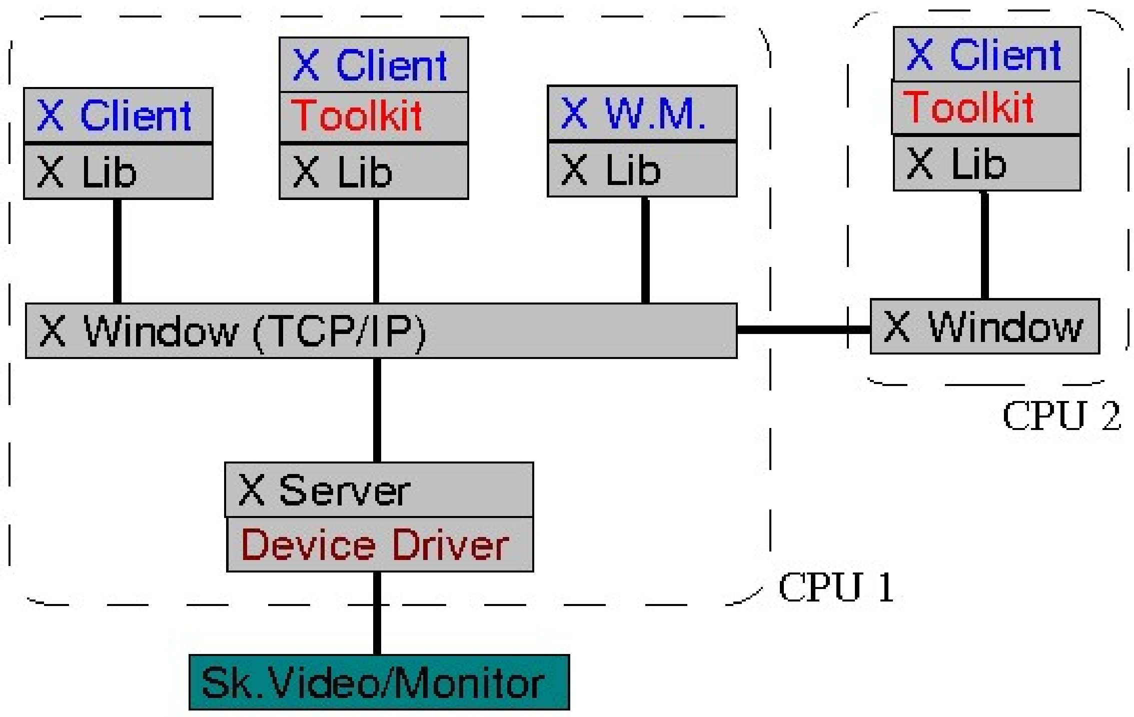
Answer: UNIX shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands.

Following are the different types of Unix shells:

a) B shell - /bin/sh – This is the default Unix shell for many Unix operating systems .
Bourne shell was written by S. R. Bourne and its more emphasis is to use it as a scripting language rather than an interactive shell .Some of the features are :
Provided support for environment variables using parameters and exportable C-shell /bin/csh was designed to provide the interactive features lacking in b shell such as job control and aliasing .
b) K shell /bin/ksh – was created by David Korn and has features of both B shell and C shell along with some additional features.
c) Bash – the Bourne again shell was developed by GNU project .It is based on B shell language and has features of C and K shells.
d) tcsh is the default shell of FreeBSD and its descendants. Essentially it is C shell with programmable command line completion, command-line editing, and a few other features.
e) Zsh is a shell designed for interactive use and it has many of the useful features of bash, ksh, and tcsh along with many new features.

Q.2 Explain the architecture of x-window system.

Answer: X-Window is a protocol based on TCP/IP stack which is used between XClient and XServer to communicate. XServer job is to receive requests from XClient as: "draw line from point to point", "draw rectangle from points to points", etc. so it sent to the device driver that it will trasform them to "hardware" codes for specific video card. XServer moreover has duty to manage input from keyboard and mouse, and transmits it to the XClient to process it. XClient job is to build the window, from borders to contents: push-buttons, menù, list-box, combo-box, etc. and sent them throw XLib function-base to XServer that it show them on monitor. Explained XClient and XServer jobs it's easy to understand that is not necessary they run on the same machine, but through X-Window protocol that is based on the TCP/IP it's very simple to connect XClient and XServer on a geographic network (internet) or a simple local network



Advantages of this architecture are enormous because we can use heterogeneous hardware and software to execute our application, for example: we have a xclient that needs a lot of power calc, but also show complex graphs, then we can execute xclient on a Cray and the xserver in a Silicon Graphics or PC, therefore the first one offers an enormous power to us of calculation while seconds has a good support video display.

Q.3 Write short notes on-

a) awk utility b) Concurrent Versions System

answer a) awk utility - Finds and Replaces text, database sort/validate/index

awk syntax :

awk 'Program' input-file1 input-file2 ... awk -f PROGRAM-FILE input-file1 input-file2

awk command searches files for text containing a pattern. When a line or text matches, awk performs a specific action on that line/text. The Program statement tells awk what operation to do; Program statement consists of a series of "rules" where each rule specifies one pattern to search for, and one action to perform when a particular pattern is found. A regular expression enclosed in slashes (/) is an awk pattern to match every input record whose text belongs to that set.

OPTIONS

Tag	Description
-F FS --field-separator FS	Use FS for the input field separator (the value of the 'FS' predefined variable).

EXAMPLES

- To return the second item($2) from each line of the output from an ls - l listing.

```
•   $ ls -l | awk '{print $2}'
•   13
•   3
•   17
•   7
```

- To print the Row Number (NR), then a dash and space ("- ") and then the first item ($1) from each line in sample.txt.

  First create a sample.txt file

```
Sample Line 1
Sample Line 2
Sample Line 3
$ awk '{print NR "- " $1 }' sample.txt
1 - Sample
2 - Sample
3 - Sample
```

- To print the first item ($1) and then the second last item $(NF-1) from each line in sample.txt.

```
•   $ awk '{print $1, $(NF-1) }' sample.txt
•   Sample Line
•   Sample Line
•   Sample Line
```

- To print non-empty line from a file.

```
•   $ awk 'NF > 0' sample.txt
```

- To print the length of the longest input line.

```
•   $ awk '{ if (length($0) > max) max = length($0) } END { print max }' sample.txt
•   13
```

- To print seven random numbers from zero to 100, inclusive.

```
•   $  awk 'BEGIN { for (i = 1; i <= 7; i++) print int(101 * rand()) }'
•   24
•   29
•   85
•   15
•   59
•   19
•   81
```

- To count the lines in a file

```
$ awk 'END { print NR }' sample.txt
```

Answer b) Concurrent Versions System

Concurrent Versions System (CVS) is a program that lets a code developer save and retrieve different development versions of source code . It also lets a team of developers share control of different versions of files in a common repository of files. This kind of program is sometimes known as a version control system . CVS was created in the UNIXoperating system environment and is available in both Free Software Foundation and commercial versions. It is a popular tool for programmers working on Linux and other UNIX-based systems.

CVS works not by keeping track of multiple copies of source code files, but by maintaining a single copy and a record of all the changes. When a developer specifies a particular version, CVS can reconstruct that version from the recorded changes. CVS is typically used to keep track of each developer's work individually in a separate working directory. When desired, the work of a team of developers can be merged in a common repository. Changes from individual team members can be added to the repository through a "commit" command.

CVS uses another program, Revision Control System (RCS), to do the actual revision management - that is, keeping the record of changes that goes with each source code file. The writers of the most popular CVS Frequently Asked Questions document are careful to emphasize that CVS is not a build system, a code configuration management system, or a substitute for other good development practices, but simply a way to control the versions of the pieces of a program as they are developed.