

RJS FIRST GRADE COLLEGE

KORMANGALA

Subject: UNIX OPERATING SYSTEM

Unit – 1

Introduction: History, salient features, Unix system architecture, Unix command format, Unix internal and external commands, Directory commands, File related commands, Disk related commands, general utilities. Unix File System: Boot inode, super and data block, in-core structure, Directories, conversion of pathname to inode, inode to a new file, Disk block allocation.

2marks questions

Q1.List four feature of Unix?

Following are some of the important features of Linux Operating System.

1.Portable

2.Open Source

3.Multi-User

4.Multiprogramming

Q2.Define kernal and shell?

Kernel :- The kernel is the heart of the operating system. It interacts directly with the hardware and most of the tasks like memory management, task scheduling and file management.It is often called Operating system.

Shell :-

Shell acts as an interface between the user and the kernal.The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. So shell is called command line interpreter(CLI).The shell uses standard syntax for all commands.

The shell invoke a command line prompt which is usually \$ or % where the user can type a unix commands.

Q3.What are the Functions of kernal?

The major tasks of kernel:

- Process Management
- Device Management
- File Management
- Virtual Memory
- Networking

- Network File Systems
- Multiprocessor support
- Lightweight process (thread) support

Q4.What is a wild card?Mention the use of wild card?

Wildcards (also referred to as meta characters) are symbols or special characters that represent other characters. You can use them with any command such as ls command or rm command to list or remove files matching a given criteria, receptively.

- An asterisk (*) – matches one or more occurrences of any character, including no character.
- Question mark (?) – represents or matches a single occurrence of any character.
- Bracketed characters ([]) – matches any occurrence of character enclosed in the square brackets. It is possible to use different types of characters (alphanumeric characters): numbers, letters, other special characters etc.

Q5.Mention the types of shells?

Types of shells:

1.C Shell 2.Bourne Shell 3.Korn Shell 4.Bourne again shell are the most famous shells which are available with most of the Unix variants.

The shell invoke a command line prompt which is usually \$

or % where the user can type a unix command

Q6.Difference between bc and xcal?

bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

If you enter `bc`, it will only do integer calculations. However, if you call it with the `-l` flag (that is, `bc -l`), it will do floating-point calculations. For more information on using this calculator, at the Unix prompt, enter:

```
man bc
```

xcalc is a graphical scientific calculator that can emulate a TI-30 or an HP-10C scientific calculator. You can run `xcalc` only in an X Window session. Easy to use

Q7. Write about bc command?

bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

If you enter `bc`, it will only do integer calculations. However, if you call it with the `-l` flag (that is, `bc -l`), it will do floating-point calculations. For more information on using this calculator, at the Unix prompt, enter:

```
#bc
```

The bc command supports the following features:

- Arithmetic operators
- Increment or Decrement operators
- Assignment operators
- Comparison or Relational operators
- Logical or Boolean operators
- Math functions
- Conditional statements
- Iterative statements

Q8. What are the different types of Unix commands?

1. External commands

2.Internal commands

1.External commands:

External command in Unix is one that exists independently as a separate file.

Most Unix commands are external in nature

Examples:Cat and ls

2.Internal commands

Internal commands do not exist independently.They are the part of another program or routine.

So internal commands are also known as builtin commands.

Example:echo ,printf commands

Q9.Write about cal and who command?

Cal command:

Cal command is used to print the calendar of a required month or a required year.When cal command is used with out any argumens the calendar of the current month of the current year is printed.

\$cal 2009

2009 calendar will be printed here

Who command: Who command provides the login details of all the current users in 3 column formate.

Q10.Write the UNIX command formate?

UNIX command lines can be simple, one-word entries like the date command. They can also be more complex: you may need to type more than the command name.

A UNIX command may or may not have *arguments*. An argument can be an option or a filename. The general format for UNIX commands is:

```
command option(s) filename(s)
```

- Enter commands in lowercase.

- *Options* modify the way in which a command works. Options are often single letters prefixed with a dash (-). Multiple options in one command line **can** be set off individually (like **-a -b**), or, in some cases, you can combine them after a single dash (like **-ab**).
- The argument *filename* is the name of a file that you want to use.

5 marks questions

Q1. List and explain the salient features of Unix operating system?

Following are some of the important features of Linux Operating System.

1. Portable

2. Open Source

3. Multi-User

4. Multiprogramming

5. Hierarchical File System

6. Shell

7. Security

1. Portable – Portability means software can work on different types of hardware in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.

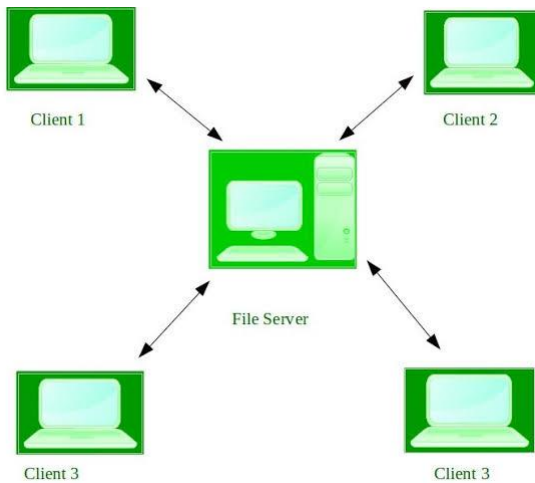
2. Open Source – Unix source code is freely available and it is a community-based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

3. Multi-User – Unix is a multiuser system means multiple users can access and share system resources like memory/ ram/ application programs/ printers/ speakers etc at the same time.

There are several types of terminals that can be connected to the server. They are

- **Dumb Terminals** (consist of keyboard & monitor do not have hard disk or memory)
- **Terminals Emulator** (consist of own memory, MP and disk drive it transmit its job to server for processing)

- **Dial in terminals** (Connect to server through telephone line)

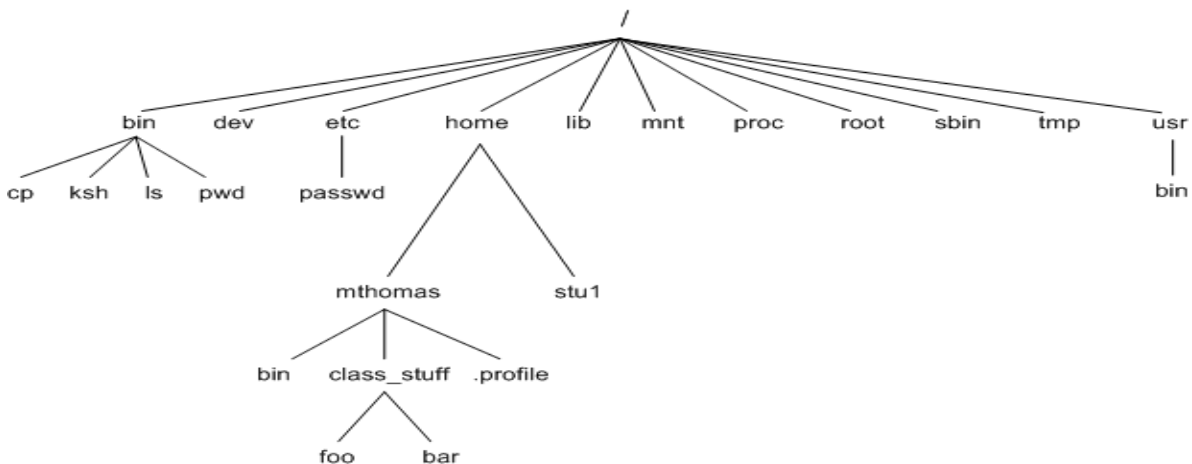


4. Multiprogramming/Multitasking system –

1. Unix is a multiprogramming system means multiple applications can run at same time. Single user can run multiple tasks concurrently.

2. Editing a file, printing another one on printer, send email to a friend and browse the www—all without leaving the application.

5. Hierarchical File System – Linux provides a standard file structure in which system files/ user files are arranged.



6. Shell – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.

7. Security – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Unix has inherent provision for protection of data

1.User Level:First at the system start up level it provide login names and password for user

2.File Level:Second file protection is provided by granting different permission to each file. Read(R) = only to read ; write (w)= permission only to write in a file

3.File encryption:Unix support file encryption(unreadable format) When required you can decrypt(readble formate)

Some other features:-

- Distributed data processing capabilities
- Open Source code
- Shell Scripts
- Powerful Pipes and Filters
- E-Mail Facility
- Support to most programming languages

Q2.Explain Unix architecture of Unix?

The Unix operating system is a set of programs that act as a link between the computer and the user.

The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or the **kernel**.

Users communicate with the kernel through a program known as the **shell**. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

A Unix operating system consist of three layer

- Hardware
- Kernel mode

- User mode

The major component of Unix system are

- Hard ware
- Kernel
- Shell & GUI
- File system

The main concept that unites all the versions of Unix is the following four basics –

- **Hard ware:** Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** – The kernel is the heart of the operating system. It interacts directly with the hardware and most of the tasks like memory management, task scheduling and file management. It is often called Operating system

The major tasks of kernel:

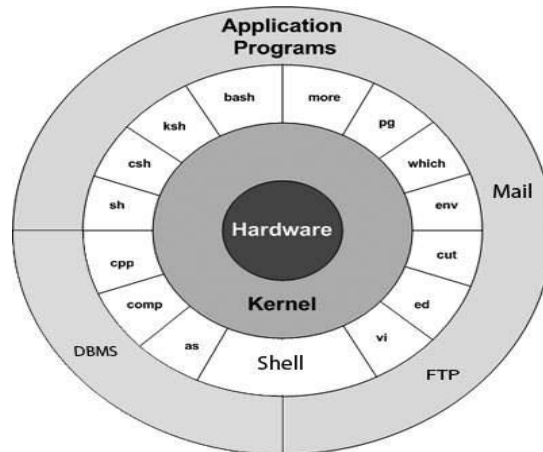
- Process Management
 - Device Management
 - File Management
 - Virtual Memory
 - Networking
 - Network File Systems
 - Multiprocessor support
 - Lightweight process (thread) support
- **Shell** – Shell acts as an interface between the user and the kernel. The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. So shell is called command line interpreter (CLI). The shell uses standard syntax for all commands.

Types of shells: 1.C Shell 2.Bourne Shell 3.Korn Shell 4.Bourne again shell are the most famous shells which are available with most of the Unix variants.

- **Commands and Utilities** – There are various commands and utilities which you can make use of in your day to day activities. **cp**, **mv**, **cat** and **grep**, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.

- **Files and Directories** – All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **file system**..

Diagram for Unix architecture



Q3. Write about bc command?

bc command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.

If you enter `bc`, it will only do integer calculations. However, if you call it with the `-l` flag (that is, `bc -l`), it will do floating-point calculations. For more information on using this calculator, at the Unix prompt, enter:

```
#bc
```

The **bc** command supports the following features:

- Arithmetic operators
- Increment or Decrement operators
- Assignment operators
- Comparison or Relational operators

- Logical or Boolean operators
- Math functions
- Conditional statements
- Iterative statements

1. Arithmetic Operators

Examples:

```
Input : $ echo "12+5" | bc
```

```
Output : 17
```

```
Input : $ echo "10^2" | bc
```

```
Output : 100
```

2. Assignment Operators

Examples:

```
Input: $ echo "var=10;var" | bc
```

```
Output: 10
```

3. Increment Operators

Examples:

```
Input: $ echo "var=10;++var" | bc
```

```
Output: 11
```

4. Decrement Operators

Examples:

```
Input: $ echo "var=10;--var" | bc
```

```
Output: 9
```

5. Comparison or Relational Operators

Examples:

```
Input: $ echo "10>5" | bc
```

```
Output: 1
```

```
Input: $ echo "1==2" | bc
```

```
Output: 0
```

6. Logical or Boolean Operators

Examples:

```
Input: $ echo "10 && 5" | bc
```

```
Output: 1
```

```
Input: $ echo "0 || 0" | bc
```

```
Output: 0
```

7. Mathematical Functions

The built-in math functions supported are :

- **s (x)**: The sine of x, x is in radians.
- **c (x)** : The cosine of x, x is in radians.
- **a (x)** : The arctangent of x, arctangent returns radians.
- **l (x)** : The natural logarithm of x.
- **e (x)** : The exponential function of raising e to the value x.
- **j (n,x)** : The bessel function of integer order n of x.
- **sqrt(x)** : Square root of the number x. If the expression is negative, a run time error is generated.

8. Conditional Statements

Example:

Syntax:

```
if(condition) { statements } else { statemnts }
```

Q4. Write about ls command in detail?

To list the files and directories stored in the current directory, use the following command –

```
$ls
```

Here is the sample output of the above command –

```
$ls
```

```
bin      hosts lib   res.03
```

```
ch07     hw1  pub   test_results
```

```
ch07.bak hw2  res.01 users
```

```
docs     hw3  res.02 work
```

The command `ls` supports the `-l` option which would help you to get more information about the listed files –

```
$ls -l
```

```
total 1962188
```

```
drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml
```

```
-rw-rw-r-- 1 amrood amrood 5341 Dec 25 08:38 uml.jpg
```

```
drwxr-xr-x 2 amrood amrood 4096 Feb 15 2006 univ
```

```
drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia
```

```
-rw-r--r-- 1 root root 276480 Dec 9 2007 urlspedia.tar
```

```
drwxr-xr-x 8 root root 4096 Nov 25 2007 usr
```

```
drwxr-xr-x 2 200 300 4096 Nov 25 2007 webthumb-1.01
```

Here is the information about all the listed columns –

First Column – Represents the file type and the permission given on the file. Below is the description of all type of files.

Second Column – Represents the number of memory blocks taken by the file or directory.

Third Column – Represents the owner of the file. This is the Unix user who created this file.

Fourth Column – Represents the group of the owner. Every Unix user will have an associated group.

Fifth Column – Represents the file size in bytes.

Sixth Column – Represents the date and the time when this file was created or modified for the last time.

Seventh Column – Represents the file or the directory name.

Q5. Write the following general purpose commands in details?

1 cal 2.date 3.echo 4.printf

1.cal commands

cal command is a calendar **command** in Unix which is used to see the calendar of a specific month or a whole year. if used without option, it will display a calendar of current month and year. **cal** : Shows current month calendar on the terminal.

```
cal [ [ month ] year]
```

- **cal** : Shows current month calendar on the terminal.

```
dharam@dharam-H110MHC: ~
dharam@dharam-H110MHC:~$ cal
December 2018
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

- **cal 08 2000** : Shows calendar of selected month and year.

```
dharam@dharam-H110MHC: ~
dharam@dharam-H110MHC:~$ cal 08 2000
August 2000
Su Mo Tu We Th Fr Sa
                1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

cal -3 : Shows calendar of 1- previous,2- current and 3- next month

```
dharam@dharam-H110MHC: ~
dharam@dharam-H110MHC:~$ cal -3
November 2018      December 2018      January 2019
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
                1  2  3                1                1  2  3  4  5
 4  5  6  7  8  9 10  2  3  4  5  6  7  8  6  7  8  9 10 11 12
11 12 13 14 15 16 17  9 10 11 12 13 14 15 13 14 15 16 17 18 19
18 19 20 21 22 23 24 16 17 18 19 20 21 22 20 21 22 23 24 25 26
25 26 27 28 29 30  23 24 25 26 27 28 29 27 28 29 30 31
                30 31
```

2.Date command

It is used to Display Current Date and Time.It is also used to change the value of, the system's time and date information.

The syntax is:

date

date "+format"

Out put of date command

```
Tue Oct 27 15:35:08 CDT 2009
```

For the following commnd:

date '+DATE: %m/%d/%y%nTIME:%H:%M:%S'

sample out put:

```
DATE: 10/27/09
```

```
TIME:15:50:44
```

3.echo Command

echo is one of the most commonly and widely used built-in command for Linux bash and C shells, that typically used in scripting language and batch files to display a line of text/string on standard output or a file.

The syntax for echo is:

echo [option(s)][string[s]

1. Input a line of text and display on standard output

```
$ echo Tecmint is a community of Linux Nerds
```

2. echo its value:

```
$ echo The value of variable x = $x
```

3. Using option ‘\b’ – backspace with backslash interpretor ‘-e’ which removes all the spaces in between.

```
$ echo -e "Tecmint \bis \ba \bcommunity \bof \bLinux \bNerds"
```

```
TecmintisacommunityofLinuxNerds
```

4.printf command

“*printf*” command in Linux is used to display the given string, number or any other format specifier on the terminal window. It works the same way as “printf” works in programming languages like C.

```
$printf [-v var] format [arguments]
```

Format Specifiers: The most commonly used printf specifiers are %s, %b, %d, %x and %f.

Examples:

- `%s` specifier: It is basically a string specifier for string output.

```
$printf "%s\n" "Hello, World!"
```

Output:

Hello

world

UNIT - II

Secondary Storage Management

Disk Space Management

Unit-II:

Secondary Storage Management: Formatting, making file system, checking disk space, mountable file system, disk partitioning, file compression. Special Tools and Utilities: Filters, Stream editor SED and AWK, Unix system calls and library functions, Processes, signals and Interrupts, storage and compression facilities.

2.Marks questions

Q1.What meant by disk management?

- User keep unwanted files on the disk due to which files tend to accumulate therefore making the system slow
- The allotted space for a file system may not be utilized fully or someone else may be in need of more space
- Whenever a new user is created, disk space has to be allocated to him
- Whenever new applications are created they need more space

Q2. What are various issues associated with managing disk space such as

There are various issues associated with managing disk space such as

- Formatting a disk
- Making file system
- Checking disk space
- Creation of partition on a disk
- Mounting a file system

5 Marks Questions

Q3What is ment by Formatting a disk

- Before you use a disk for backup purposes first you format it first
- A disk can be formatted in two ways
 - Low level format
 - High level format

a.Low level format (or) Physical format

- It is the first step in formatting
- It is dividing the disk track into specified number of sector
- Filling the data area of each sector with dummy bytes value
- This result in destruction of any existing data on the disk

b.High level format (or) Logical format

- It is the creation of a file system including a table of content for the disk
- It does not destroy data already on the disk
- A disk can be formatted in Unix using format command or fd format command
- The **fdformat** command is used to perform the lower level formatting of floppy disk
- Syntax:

`fdformat[-n]device`

Device – it is the location of the floppy disk

`$fdformat/dev/fd0`

o/p: it will perform low level format on floppy disk press enter key it will start formatting.

Q4.Explain how we can make Making file system in UNIX.

- Once partition on the hard disk is created by system administrator, he has to create a file system on this partition to make it usable
- **mkfs** command is the universal file system creation tool
- **mkfs** command is mostly used to create a new file system
- **newfs** command provides friendlier interface and call **mkfs** internally
- To create a file system in Linux with their own tree structure and root directory

- **Syntax**

```
mkfs[-t type][fs -option]filesystem[blocks]
```

- The item in square bracket is optional and filesystem is mandatory
- Filesystem is the name of the device
- It is mount point for new file system such as root directory, /usr or /home

- **Option**

- -t fstype(specifies the type of the file system to built)
- fs -option(file system specific option to be passed)
- **Sqrt(x)** – square root

```
$ mkfs /dev /fd0
```

Q5.How can we check the disk Space using Unix commands

In unix number of command that can aid you in this task of checking free space.

df and **du** command can be used by any user to report disk usage or free space in term of blocks

dfcommand – Display free space

```
$df[-option(s)] [device(s)]
```

- It is used to find the amount of space used by mounted file system
- It can report the amount of free space available for each file system
- It is commonly used to show not only space available and used but also what file system are mounted, the number of inodes still available or to install a new

program

- When used with no option and no argument, df generate the following information
 - First column display device name
 - Second column shows the number of block available
 - Third column display the number of used blocks
 - Fourth column shows the number of inodes available
 - Fifth column shows the percentage of blocks used
 - Last column represent the directory on the file system mounted

Option

- h (it provides easy to read output by reporting in large unit like GB,MB etc)
- k (display block available in 1k block)
- m(display block available in 1m block)
- i(display inode usage)

```
$df/dev/hda3
```

du command: Disk Usage Command

Syntax: ducommand[-option]directories

- It report the size of directory tree inclusive of all their content and size of the individual files
- It report the amount of space taken by each sub directory as well as current directory
- This command generate report in terms of block used

Options

- s (report on each user home directory)
- a(display the space that each file taking up)
- k(report the file size in units)
- ch(display file size as well as total capacity of file combined)

\$du-s*.txt

\$du-ch*.txt

Q6. Write about Unlimit command – user limit?

- faulty program or corrupted file may occupy huge amount of space
- It may run into several mega bytes of disk space, ultimately harming the file system
- Creation of such file can be avoided using ulimit command. This command is built in all shell
- This command imposes a restriction on the maximum size of the
 - file that a user is permitted to create
 - When used by itself, display the current setting

Q7. Write about Mounting and Un mounting of a file system?

Data store in device like

- Floppy disk
- CD ROMS
- Hard disk

— Migrate to unix from window

- You can access all file easily
- **However is not in case of linux**

— You can attach to some existing directory on your system before they can accessed

Mounting

The process of attaching a device to a directory is called mounting

- The device where the directory is attached is called mount point
- A device is mounted to a directory using the mount command
- After the device is mounted you can access the file on that device
- By accessing the directory where the device is attached
- Mounting a file system means that you are presenting the file system to the system to the end user

Syntax of Mounting Command

\$mount[-option] device_name mount_point

- It takes two argument
 - The name of the file system
 - The directory under which it is mounted
 - Before mounting an empty directory must be created in the file system
- The root directory of new system need to be mounted on this Directory

Two directories are created as default mount point

1./mnt - it is directory that use to mount occasionally it is going to test whether some device is really mountable

2./media – mount devices that are connected on a regular basics

\$mount/dev/cdrom/media

* If you are using mount command without argument it will display a list of all mounted devices

unmount command

- It is unmounting file system
- All mounted file system are unmounted automatically when a computer is shut down(USB)
- The reverse process of detaching a file system from another file system is known as unmounting
- Syntax

\$unmount[option]file system

Q8. Write the command for Disk Partitioning – fdisk command?

- It is the act of dividing a hard disk drive into multiple logical storage unit referred to as partition (Slices)
- It is created after the disk is formatted
- Linux uses fdisk command to create partition
- It is easy to name the partition device
- The first is represented by /dev/hda with partition hda1,hda2 etc

- The second hard disk will have the name /dev/hdb with similar extension

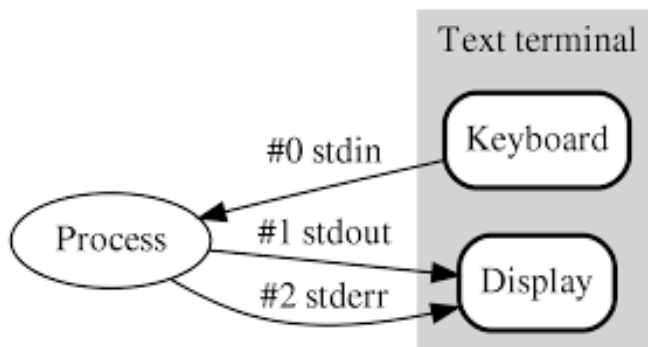
Q.9 What are the Standard Streams in Unix

- Most of the commands in unix take input from keyboard and send the output to terminal screen
- The command have been programmed to accept input from standard input file(keyboard) and produce output on a standard output file(terminal screen)
- The input and output files are stream of character which are given as input or sent as output
- In unix OS the standard streams are pre connected input and output channel between a computer program and its I/O devices
- The three I/O streams files are called standard streams
a).standard input(stdin),b).standard output(stdout) andc).standard error(stderr)
- **Standard input is textual data going into a program**
 - The redirection operator < ,input is expected from keyboard which started the program
 - The input also be given from output of another command using pipe (|) symbol
- **Standard output is the stream where a program writes its output data**
 - The redirection operator > , output is the text terminal which initiated the program
 - The output also be sent as input to another command using pipe (|) symbol
- **Standard error is output stream used by program to output**

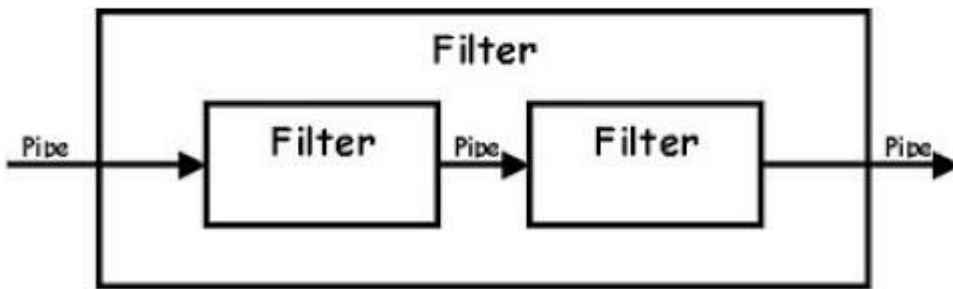
error message or diagnostics

Pipes in Unix

- You can connect two program together so that the output from one program become the input of the next program (pipe)
- The pipe is represented by vertical line character (|)
- The mechanism of redirecting the output of one command as input to another command directly without using intermediate files is called piping
- The sequence of commands using one or more pipes is called a pipeline



Q10. Write a note on Pipeline of two programs sent to standard output?



UNIX PIPES

A Complete tutorial on Pipes in Unix Programming

Pipes in Unix

- You can connect two program together so that the output from one program become the input of the next program (pipe)
- The pipe is represented by vertical line character (|)
- The mechanism of redirecting the output of one command as input to another command directly without using intermediate files is called piping
- The sequence of commands using one or more pipes is called a pipeline

Q11. Define Filter? Write some of filter Filter command?

Definition of Filters:

- A filter is a computer program to process a data stream
- Unix is rich with filter programs

- Filter commands accept some data as input perform some manipulation on it and produce output
- They perform action on the data they are appropriately called filters

A common use of filter is to modify output

1.Filter commands –

1. head command

- To display the beginning of a file

Syntax

`$head[-option][-number]file`

- It display the contents at the top of the file
- It display first 10 lines of the file
- **HEAD:** This command is used to display the lines from top of the file.

Options:-

- a) Head-n filename - To display n lines from top of the file.
- b) Head-nc filename – To display the number of character. If you want first 50 lines you can use head -50 filename or for 37 lines head -37 filename and so forth.

2.Tail command

- This command is used to display the lines from bottom of the file.

Options:-

- a) Tail-n filename - To display n lines from bottom of the file.
 - b) Tail+n filename – To display from the nth line to end of the file.
 - tail filename by default will display the last 10 lines of a file.
If you want last 50 lines then you can use
tail -50 filename.
- It does not perform any filtering action on its input. It gives out exactly what it takes
 - It can be used with any command that sends its output to standard output
 - When it is used with –a option, it append the redirected output to the specified file rather than overwriting

3.Cut command:

- cut command selects a list of columns or fields from one or more files.
- Option -c is for columns and -f for fields.
- It is entered as cut *options* [files]

for example if a file named testfile contains

- this is first line
- this is second line
- this is third line

Examples:

cut -c1,4 test file will print this to standard output (screen)

- It is printing columns 1 and 4 of this file which contains t and s (part of *this*).

Options:

-c *list* cut the column positions identified in list.

-f *list* will cut the fields identified in list.

-s could be used with -f to suppress lines without delimiters.

- a) cut-c 1, 4, 7 filename – To cut and display fields 1, 4, and 7 only if fields are separated by space or tab.
- b) cut-c 1-20 filename - To cut and display 1 to 20 characters in a file.

4.Paste Command

- paste command merge the lines of one or more files into vertical columns separated by a tab.
for example if a file named test file contains
- this is first line

- i ignores non-printing characters.
- n sorts in arithmetic order.
- o *file* put output in a file.
- +m[-m] skips n fields before sorting, and sort up to field position m.
- r reverse the order of sort.
- u identical lines in input file appear only one time in output.

6. Uniq command

- **uniq command** –when duplicate entries are found in file they can removed from file by using this command.
 - It takes only one sorted files as its argument
 - When the command is used without any option, the output is displayed without the duplicate entries

Options:

- -c print each line once and counting number of times
- -d print duplicate lines once.
- -u print only unique lines.

7. tr command – translating character

Syntax

tr[-option]expression1 expression2 standard_input

- It copies the standard input to the standard output with substitution or deletion of selected character
- It can also squeeze repeating character into a single character using certain option

Option

- d delete the specified character in expression1
- s replace the repeated character in the expression

8. Tee command

- Display output and redirect output
- Syntax

`tee[-option][file]`

- It is used when you want to redirect the output to another file and also see the output on the screen
- It uses standard output and standard output, which means that it can be placed anywhere in a pipeline
- It break the input into two component
 - One component is saved in a file
 - The other is connected to the standard output

9. Grep Command

- `grep` command is the most useful search command.
- You can use it to find processes running on system, to find a pattern in a file, etc.
- It can be used to search one or more files to match an expression.
- It can also be used in conjunction with other commands as in this following example, output of `ps` command is passed to `grep` command, here it means search all processes in system and find the pattern **sleep**.

Options:

-b option will precede each line with its block number.

-c option will only print the count of matched lines.

-i ignores uppercase and lowercase distinctions.

-l lists filenames but not matched lines.

Options:-

- a) `grep "billu" filename` - To search a word "billu" in the file and display that line.
- b) `grep -v "rock" | filename` - To search except the word rock in the file.
- c) `grep "e$"| filename` - To search for a line ending with e.
- d) `grep "lf"| filename` - To search for a line beginning with f.

Unix System call

Q12. Define System call? Write about Unix system calls related to files?

- System call is the interface between the process and the operating system
- It is the direct entries to the kernel
- It can only interact with commands, like shell, text editor, and other application program
- It is the only way to access kernel such as the file system, the multitasking mechanism and inter process communication

1. System call for low level I/O

- lowest level of the I/O is actually a direct entry into the operating system
- The system call are the programs that make a request to the os for the service (look function call in c)
 - The system call used for low level file I/O are
creat(name, permission) Open(name, mode) close(fd)

unlink(fd) read(fd, buffer, n)
write(fd, buffer, n) lseek(fd, offset, whence)

Create system call

- **No e to create a file in system call**
- If the file does not exist, the creat call create the specified filename(name) with the permission specified by second argument permission
- The permission are specified by an octal number with leading 0(zero)

fd= creat (name, permission)

2. open system call

- It is a system call to open a file
- Open return a file descriptor(fd)
- It is an integer specifying the position of this open file in the table of open files for current process

fd = open(name, mode)

- name argument contain the filename
- Mode specifies

access mode 0 – read

1-write 2 – both

-1 return error

3.close system call

- It is a system call to close the file
- It break the file name and descriptor(fd)
- Termination of the program with an exit statement or return statement in the main program, closes all open file

Close(fd)

4.unlink system call

- Remove the file from the file system itself

Unlink(fd)

5.read system call

Read file from the file opened for reading `read = read(fd, buffer,n)`

Where fd is the file descriptor

Buffer is an array(acting as data source or destination)

N is the number of bytes to be transformed

6.write system call

- Write data from the file opened for writing `nwritten = write(fd, buffer, n)`
- Where fd is the file descriptor

- Buffer is an array
- N is the number of bytes to be transformed

7.lseek system call

- Seek a specified position in a file
- It is used for random access while reading or writing a file
$$\text{pos}=\text{seek}(\text{fd},\text{offset}, \text{whence})$$
- Offset – is the position to move
- Whence is the position where the offset position is measured 0 – beginning 1- current 2 – end of file
- Pos – value return

B.System call for process control

- The unix system provide several system call to create and send program, to send and receive software interrupt, to allocate memory and do other job for a process
- Four system call for creating, ending, waiting for a process to complete.
- They are

fork()

Wait()

execl(), execlp(),
execvp() Exit()

fork()

- It is a system call that create a new process under UNIX OS
- Eg if a program contain fork() the execution of the program result in the execution of two processes (the parent process another child process).
- It is a time shared operating system the two process run concurrently

Syntax

proc_id=fork()

The value return by fork() is stored in proc_id (integer)

Each process will return their pid

To find out id of a process getpid() system call is used

Wait()

- *The parent wait for the child to terminate before continuing itself*
- Syntax

wait(&status)

- Status is the pointer to an integer where unix stores the value returned by the child
- The process to wait for a signal

- It is zero for normal termination and nonzero to indicate different kind of error

execl & execv

- The unix system call that transform an executable binary file into a process are the exec family of system call
- Execl (execute and leave) system call load a new executable into memory and associated it with the current process
- Execl takes the path name of an executable program(binary file) as its first arguments
- The execl and excev() the filename must be fully qualified path name of the executable binary file

execlp and execvp

- *The system call execlp is used to execute another program without returning*
- It halt the currently running program, execute the new program and then exit
- Execvp a variant of execlp is used when the number of argument are not known in advance

execvp(filename, argp)

argp is an array of pointer to the argument, the last pointer is null to indicate end of list

- Letter added to the end of exec indicate the type of argument
I – argn is specified as a list of argument v – argv is specified as a vector

p- user path is searched for command and command can be shell program

exit()

- The exit() system call end a process and return a value to its parents
- The prototype that exit() system call is

exit(status)

Status is the integer between 0 to 255

This number is returned to the parent via wait(), as the exit status of the process

If zero – success

If non zero - failure

C).System call for IPC

- The system call used for inter process communication

Pipe(files) dup(fd)

- Pipes
 - The pipe is the connection between two process
 - One process cannot read from buffer until the another has written to it
 - The Unix command line interpreter provide a pipe facility

\$ prog |more

Pipe system call

- The pipe is a system call that facilities inter process communication
- It open a pipe which is an area of main memory that is treated as a virtual file
- The pipe can be used by creating parent process as well as child process for reading and writing

One process can write to this virtual file or pipe and another related process can read from it

UNIT –III

SHELLPROGRAMMING

2Marks Questions

Q1. Define Shell?

Shell is an Interface between user and the kernel. It is also called Command Line Interpreter (CLI).

Prompt for executing Unix program is called *shell prompt* (\$, %, #)

Q2. Mention types of shell?

- **Bourne shell (sh):** \$
- **C shell (csh):** %
- **Korn shell (ksh):** \$
- **Bourne-Again shell (bash):** \$
- **#:** indicates any shell

Q3. Define Editor? What are the Shell Editors available in Unix?

Editors:Software package used to enter and execute any program

Unix editors : Vi, Vim, emacs, pico

Q4.What are modes of Vi Editor?

1. Commandmode
 - Hereeverycharacter typedisacommand
 - <Esc>:usedto returnto commandmode
2. Insertmode
 - Everycharacter typedis addedtothetextinthe file
 - Insert ori
 - <Esc>:usedto exitinsertmode
3. Exmode
 - Last-linecommand
 - Makesusertoentercommandsatthebottomofviscreen
 - Colon(:)isusedtoenterexmode

Q5.List the Types Shell Variables?

Types of shell variables are

- Systemvariables
- Localoruserdefinedvariables
- Read-onlyvariables

Q6.What is the Data type of shell variables bydefault?

- All shell variables are **string** type
- Content are stored in ASCII format
- By default, shell variables are initialized to null string; hence it is not required to declare or initialize them

- Ex:

```
Num=10  
echo $Num
```

Q7.What is meant by "COMMAND SUBSTITUTION"?

- Two commands can be connected either by using pipeline or by command substitution
- Back quote (` `) works only within double quotes and doesn't work within single quotes
- echo " Current date is `date` "

 - Current date is Tuesday Feb 18 11:55:59 IST 2014

- echo ` Current date is `date` `

 - Current date is `date`

Q8.Define Shell Script?

A script is a file that contains a set of shell commands.
Ex:fl.sh,fact.sh

5marks Questions

Q1.List and Write about types shell in details?

- **Bourne shell(sh):\$**
- C shell(csh):%
- Korn shell(ksh):\$
- Bourne-Again shell(bash):\$
- #:indicates any shell

1.Bourn Shell

- Symbol :\$

- Executable filename is **sh**
- Default shell for Unix
- Developed by Stephen Bourne

2.C-Shell

- Symbol : %
- Executable filename is **cs**
- Similar to C program
- Developed by Bill Joy
- **Advantage over Bourne Shell**
 - C shell can execute processes in background
- **TC shell (tcsh):** Free version of C-shell under Linux

3.Korn Shell

- Symbol : \$
- Executable file name is **ksh**
- Developed by David Korn
- Korn shell = Bourne shell + C shell

4.Bourne-Again shell

- Symbol : \$
- Executable filename is **bash**
- Developed by Fox and C Ramey
- Bash is a free ware shell
- Default for Linux

Q2.What are the steps that Shell process as command line processor ?

Steps

1. First it **parses** the command line, identifies and removes extra spaces, tabs etc
2. Evaluates variables prefixed with \$ sign
3. Executes commands with back quotes
4. Redirection is checked if any and connects the concerned files
5. Substitutes meta-characters like *
6. Looks for required files and commands, retrieves them and transfers them to the kernel for execution
7. PATH variable helps to find the path for the commands
8. Semicolon involves multiple commands

Q3.Define Editor? Write about in detail about the Editor?

Editors: Software package used to enter and execute any program

Unix editors : ed, Vi, Vim, emacs(ex), picoEditor

1. ed:
 - First editor on Unix
 - Developed by Ken Thompson
2. ex:
 - Extended editor
 - Developed by Bill Joy
3. vi:
 - **Visual editor**

- Screen-oriented version of ex
4. vim:
- **ViImproved**
 - Improved version of vi

Q4. Invoking vi and Quitting of vi editor?

1. Invoking vi editor

- vi filename.extension
- Filename occurs at the bottom of screen
- Options:
 - vi filename: edits filename starting at line 1
 - vi -r filename: recovers filename that was being edited when system crashed

vi+n filename: edits filename

2. Quitting vi

- While quitting vi, new or modified file is automatically saved
- Options:
 - :x/:wq/:q - quits vi, writing modified file
 - :w - writes modified file and remains in command mode

- :q!–quitsvi,without savingthelatest changes
- ZZ–savesandexits;thisisknownaslast command

Moving the cursor :

- Mouse does not work in Unix
- Arrows and keys has to be used
- Options:
 - h:movescursorleftonecharacter
 - l:movescursorrightonecharacter
 - j:movescursordownoneline
 - k:movescursoruponeline

Arrow keys

- Slow on lengthyfiles
- But sometimes they produce strange effects
- Options:
 - 0(zero)–moves cursor to star to current line
 - \$-moves cursor to end of current line
 - W-moves cursor to beginning of next word
 - b-moves cursor to beginning of preceding word
 - 0or1G-moves cursor to first line in file
 - nornG-moves cursor to linen
- **Capslock(^)beforealetterindicatesctrl**

- Options:
 - ^f:movesforwardonefullscreen
 - ^b:movesbackwardonefullscreen
- ^d:movesforwardonehalfscreen

Q5. Define Shell variables? What are types of Shell variables write in details?

- They have the ability to store and manipulate information within a shell program
- Rules for naming variables:
 - Can contain alphanumeric character and underscore(_)
 - Must start with alphabet or underscore
 - Case sensitive
 - No limit on length of variable name
 - Ex: No_of_std, NAME

Types of shell variables

- a) System variable or Environment variables or Predefined variables
- b) User defined or Local or user defined variables
- c) Read-only variables

a). System variables or Environment variables or Predefined variables

- Also known as environment variables
- Set either during booting or after logging in

- Written in uppercase only
- Ex: PATH,HOME,IFS,SHELL,LOGNAME,OSTYPE,PS1,PS2

List of System variables

- PATH : list of directories separated by colon(:)
- HOME : path of home directory
- SHELL : absolute pathname of user's shell program
- LOGNAME : stores user name
- OSTYPE : type of OS
- PS1 : holds primary prompt value (\$)
- PS2 : holds secondary prompt value (>)
- IFS:
 - Internal FieldSeparator
 - Holdstokenusedtoseparateastringintosub-strings
 - 3defaulttokens:space,tab,newline
 - **od**:usedtodisplaynon-printablecharacters
 - **-b**:displaysoctalvalue
 - **-c**:displayscharacteritself

Ex:\$echo\$IFS|od-bc011012 012

\t \n \n

b).User Defined Variables or Local Variables:

- Variables are defined and used by users

- Exist only during execution of shell program
- Local to the user's shell
- Not accessible to other users

Ex: read x, y, z x=10 y=20 z=30

c).Readonly Variables

- Values are fixed
- 'readonly' function is used to convert any variable to read-only variable
- Ex: echo Enter a number:

```
Readonly num #Enter a number:5
```

```
echo N=
```

```
$n
```

```
#N=5
```

```
readonly
```

```
n
```

```
n=n+10
```

```
echo N=$n
```

```
#N=5
```

Q6. what are the features of shell script?

- It is a complete programming language
- It consists of sequence of commands for selective execution, I/O operation and looping

- It runs in an interpretive mode, executing one statement at a time
- They are named just like any other files
- When a shell script is created for the first time, it will have only read and write permissions. Execution permission must be granted

Q7. How to execute a shell script?

1. shfilename.sh
2. chmod +xfilename.sh
./ filename.sh

Eg \$ chmod u +x prog1.sh

\$/prog1/.shwelcome

Q8. Write about Shell Input and output functions?

Shell input statement is `read` command:

- Shell allowstoprompt for user input
- Read statement is used to get input from user and store data into a variable
- Syntax:

read varname1, varname2.....

Shell out put statement echo statement:

echo "Enter 3 numbers" read a bc

```
echo "Entered numbers are $a $b $c"
```

```
read -p "enter your name: " first last  
echo "First name: $first" echo "Last  
name: $last"
```

Q9. How can we do Arithmetic in Shell Programme?

1. Integer Arithmetic using Expr command

2. Real Arithmetic using bc command

1. SHELL ARITHMETIC USING expr

- All variables in Unix are string variables
- expr command is used to convert string variables to numeric format
- Arithmetic operators:
 - +, -, *, /, %
- **bc** is used for floating point calculations

```
a=5 ;b=10
```

```
expr $a +$b
```

```
expr $a -$b
```

```
expr $a \*$b
```

```
expr $a /$b
```

```
expr $a %$b
```

RULES FOR ARITHMETIC OPERATIONS

- `expr` is used with `command` substitution using back quotes (`` ``) to assign values to variables
- `a=5; b=10`
- `a=`expr $a + 1`` or `a=$(expr $a + 1)`
- `echo $a`

RULES FOR ARITHMETIC OPERATIONS

- Multiple assignments can be done in a single line
- Operators in `expr` command must be followed and preceded by at least one blank space

`$ expr $a + $b + $c`

- Hierarchy of operators:

□ 1: / * %

□ 2 : + -

2.Real arithmetic's using bc command:

- For real arithmetic's, basic calculator (`bc`) is used
- Output of arithmetic calculation are piped to `bc`
- `scale` function is used to set number of decimal places after decimal point
- Ex: `a=5.5; b=10.5`

`scale=2`

```
c=`echo $a + $b | bc`
```

```
d=`echo $a \* $b | bc`
```

```
echo $c $d
```

Q10. Write about Positional Parameter in Shell programming?

- Information can be conveyed to a shell script through command line argument or shell script arguments
- These argument submitted with a shell script are known as positional parameters
- Bourne shell stores the first nine command line arguments in the variable

\$1, \$2....\$9

POSITIONAL PARAMETERS

Command line arguments submitted with a shell script

- Positional parameters automatically store values of command line arguments
- **set** command is used to assign values

Parameter	Meaning
\$0	Name of the current shell script
\$1-\$9	Positional parameters 1 through 9
\$#	The number of positional parameters
\$*	All positional parameters, "\$*" is one string

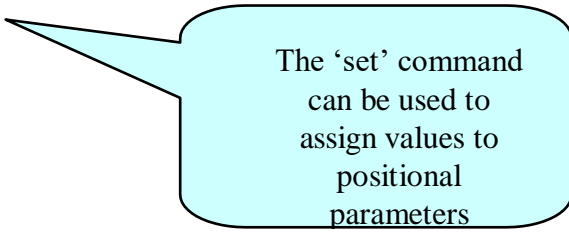
\$@ All positional parameters, "\$@" is a set of strings

\$? Return status of most recently executed command

\$\$ Process id of current process

\$ set tim bill ann fred

\$1 \$2 \$3 \$4



The 'set' command can be used to assign values to positional parameters

\$ echo \$*

tim bill ann fred

\$ echo \$# 4

\$ echo \$1 tim

\$ echo \$3 \$4 ann fred

\$ set `date`

Tuesday Feb 18 11:55:59 IST 2014

\$1 \$2 \$3 \$4 \$5 \$6

\$ echo \$*

Tuesday Feb 18 11:55:59 IST 2014

```
$ echo $#           :
```

```
6
```

```
$echo$1           :
```

```
Tuesday
```

```
$ echo $4         :
```

```
11:55:59
```

Exit command

- Terminates the execution of shell scripts
- If program is executed successfully, it returns non-zero; otherwise zero value is returned
- `$?` : variable that stores the status of exited command

Q11. Write about types of test command in shell script?

test expression

Or

[expression]

Ex: a=5;b=10

```
test $a-eq $b;
```

```
echo $?
```

```
[$a-eq$b];
```

```
echo $?
```

Test command is 3 types:

1. Numeric test
2. String test
3. File test

1. Numeric Test

<u>OPERATOR</u>	<u>MEANING</u>	<u>USAGE</u>
S		
-eq	Equal to	if [5 -eq 6]
-ne	Not equal to	if [5 -ne 6]
-lt	Less than	if [5 -lt 6]
-le	Less than or equal to	if [5 -le 6]
-gt	Greater than	if [5 -gt 6]
-ge	Greater than or equal to	if [5 -ge 6]

Output: True: \$?=0; False : \$?=1

2. String Test

- If string contains more than one word separated by white space, then they must be enclosed in double quotes

- Ex: str1="New Horizon College"
- While comparing such strings they must be enclosed in quotes
- Ex: ["str1" = "str2"]
File Test

3.FILE COMPARISON

<u>TEST</u>	<u>MEANING</u>
-e file	True if file exists
-f file	True if file exists and a regular file
-r file	True if file exists and is read-only
-w file	True if file exists and is writable
-x file	True if file exists and is executable
-s file	True if file exists and is non-empty
-d file	True if file exists and is a directory

Q12. Write about "LOGICAL OPERATORS" and its hierarchy

OPERATORS

TYPE

! expression

Logical NOT

expression1 -a expression2

Logical AND

expression1 -o expression2

Logical OR

HIERARCHY OF OPERATORS

OPERATOR

TYPES

!

Logical NOT

-lt, -gt, -le, -ge, -eq, -ne

Relational

-a

Logical AND

-o

Logical OR

UNIT IV: Conditional Control Structures

Introduction

2Marks question

Q1.How can we classify the control structure?

In to two ways:

1.**Conditional control structure** are also known as branching control structure or selection structures. Decision making can be carried out by using branching control structure or selection structures.

2.**Loop Control Structure** are required whenever a set of statement must be executed repeatedly. The repeated execution also need decision making to terminate the loop

Q2.List the types **Conditional control structure**?

1. **if –else statements**

2. **case-esac statements**

Q3.**Write the syntax of if else?**

If then else fi statement

if conditional expression then

true block

else

false block

fi

Q4.List the loop control structure?

- 1.While
- 2.Until
- 3.For loop

Q5.List the jumping statements.

- 1.Break
- 2.continue
- 3.Exit statements

5 Marks Questions

Q1.Write about Branching Control structures

- If then fi statement
- If then else fi statement
- If then elif else fi statement
- Case eacsc statement

1.If then fi statement

**if conditional
expression then**

true block

fi

statements are executed only if **command** succeeds,

i.e. has return status "0"

`$?= 0, if true`

`$?=1, if false`

Find largest of two numbers

Clear

`echo " enter two number"`

`Read a b large=$a`

`If [$b -gt`

`$large]; then Large=$b`

`fi`

2.If then else fi statement

if conditional expression then

true block

else

false block

fi

Example: Leap year or not

`echo enter a year read year`

`x=`expr $year % 4` If [$x -eq 0]`

`Then`

`echo $year is a leap`

```
year else
    echo $year is not leap year
fi
```

Ex:Odd or Even

```
clear
```

```
echo enter a number read n
if [expr $num % 2` -eq 0]
then
echo n is a even number
else
echo n is not a even number
fi
```

3.if then elif else fi statement

```
if [ condition1 ];
    then
    statement1
```

```
elif [ condition2 ];
    then
    statement2
```

```
elif [ condition3 ];
    then
    statement3
```

```
else
```

```
    default_statement
```

fi

- The word **elif** stands for “else if”

It is part of the if statement and cannot be used by itself

Example: Find whether a number is positive, negative or zero

```
echo enter a number
```

```
Read num
```

```
if [ $num -gt 0 ];  
then echo $num is  
positive elif [ $num  
-lt 0 ]; then echo  
$num is negative elif  
[ $num -eq 0 ]; then  
echo $num is zero
```

```
else
```

```
echo kindly enter a valid input fi
```

case --esac statement

- Used for a decision that is based on multiple choices

- Syntax:

case value in

pattern1) command-list1

;;

pattern2) command-list2

;;

patternN) command-listN

;;

***) default-list**

;;

esac

- The value is compared against the patterns until a match is found
- The case statement starts with the keywords case and ends with the keyword esac
- Block of commands attached to every pattern must be terminated with double semicolon(;;) but not compulsory with default pattern
- The default *) pattern gets executed when no match is found
- Case patterns (label) can be in any order

Unix commands using case statement

- 1) display list of files
- 2) display todays date
- 3) display calendar
- 4) display logged user
- 5) display current directory
- 6) quit

echo menu

echo 1.list of files
echo 2.todays date

echo 3.display month of calender

echo 4.logged user
echo 5.display current

directory

echo 6.quit

echo "enter the choice" read ch

case \$ch in

```
1)    ls;;
2)    date;;
3)    cal;;
4)    who;;
5)    pwd;;
6)    exit;;
*)    echo invalid choice
      ;;
```

esac

•

Q.2. Write about Looping control structures?

- Loops are required whenever a set of statement must be executed repeatedly
- The repeated execution also need decision making to terminate the loop
- The three types of looping they are

— while loop

— for loop

— until loop

1. while loop

To execute commands in “command-list” as long as “expression” evaluates to **true**

Syntax:

```
while [ expression ] do  
command-list done
```

Ex: Sum of digits

```
clear sum=0  
echo "enter a number" read num  
n=$sum  
while [ $num -gt 0 ] do  
rem=`expr $num % 10` sum=`expr $sum + $rem` num=`expr $num / 10` done  
echo the sum of digit of $n is $sum
```

EXAMPLE: Using while loop

COUNTER=0

```
while [ $COUNTER -lt 10 ] do
echo $COUNTER let COUNTER + =1
done
```

Until Loop

Purpose:

To execute commands in “command-list” as long as
“expression” evaluates to **false**

Syntax:

```
until [ expression ] do
command-list done
```

EXAMPLE: USING THE UNTIL LOOP

#!/bin/bash

```
COUNTER=20
until [ $COUNTER -lt 10 ] do
echo $COUNTER let COUNTER - =1
done
```

THE FOR LOOP

Purose:

- The exit statement is used to terminate a program

- Syntax

 - Exit

UNIT -V

UNIXSYSTEM COMMUNICATION

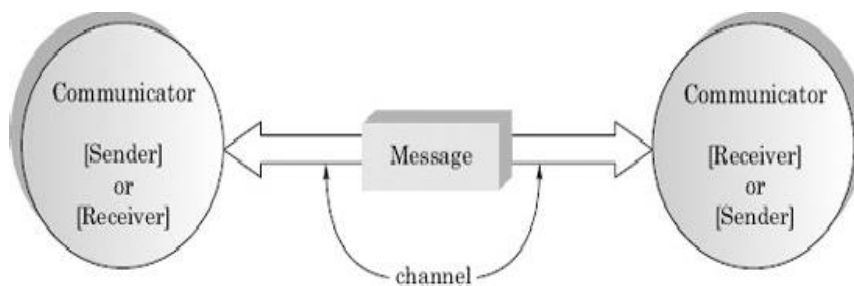
2Marks Questions

Q1.What is the use of communication?

- Electronicmailoremailiseasiestwayof communication onunix.
- Fast andcheap
- Used to Exange graphics,sound and video files.

Q2.What are the Elementsofcommunicationprocess?

A.1.Sender 2.Receiver 3.Message 4.Communication channel



System administration

2marks

Q3.What is the basic functions of system administrator?

- It is the installation and maintenance of the unix computer system.
- He is to maintain hardware and software of this system
- It include hardware configuration, software installation, reconfiguration of the kernel, networking etc
- So that the system admin can call as super user or root

- A super user on a unix system is one, who has unrestricted access to all files and command
- The login name is root

Q4. What is the user name of a System Admin?

Root is default user name of system admin.

He is also called as Super User

Q5. Write about File Encryption using crypt command?

- Crypt command is used to protect individual files from other including superuser
- It relies on simple substitution
- ~A changed to ^A
- It is used for both encryption and decryption

```
$crypt xyz <test> test.crypt
```

The above command crypt the file test using key xyz

```
$crypt xyz <test.crypt> test (decrypt)
```

```
$crypt < test > test.crypt key
```

Q6. What are the ways to acquiring the super-user status?

- **There are two ways to acquire super user status**
 1. To log into console directory as root
 2. To execute the **su** command after logging in under a different username

5marks questions

Q1.What is the the mesgcommand:mesg?

- Unix facilitates usersto send messages to other user's terminals who are logged in.
- This is possible only if other terminal has given a write permission.
- The **mesg** command is used to change the write permission of a user.

Options:

- y for yes
- n for no

example:

- \$mesg y #grants write permission
- \$mesg n #denies write permission
- \$mesg #displays current status of write permission of the terminal.

Q2.What is the function of write command?

Allows a two-way communication between two users who are recurrently logged in and who have given write permission.

Example:

```
$write prathiba
```

```
Going on a tour?happy journey.
```

```
-rama
```

```
<ctrl+d> #indicates end of message
```

```
Immediately this message appears on user pratibha's terminal with a beep sound.
```

```
Message from rama@maharanionpts/2 at 14:43.....
```

```
going on a tour?Happy journey.
```

```
Rama
```

EOF

- Conversation continues until one or both users decide to end it.
- Both users must be logged in else error message appears.

Q3. What is the use of finger command?

- Similar to **who** command.
- It lists the details of users who have logged in and given permission to accept messages.

Example:

```
$finger
```

```
Login  Name Tty      Idle  Logintime      office  officeph.
Rama   rama pts/1          Feb2813:57
Uma    uma.k pts/2          *Feb 2814:21
```

In this example

- **Login** shows login name of users
- **Name** shows full name of the users
- **Tty** shows device number of the terminals
- **Idle** shows idle time since user logged in
- **Logintime** shows time of logging in of the users
- **Office** and **officephone** columns shows address and phone number of the user

Q4. What is the use of the wall command?

- **Wall** stands for **write to all**
- This command can be used only by **superuser**

- Used to send message to all users on the system, known as broadcasting a message to all users. Irrespective of whether users have given write permission or not.
- Wall executable file is stored in the /etc directory.

Example:

```
$wall #message send by superuser
```

There may be power failure.

please save your files.

<ctrl d>

When the command is executed, UNIX displays the following message on all the Logged-in users

```
Broadcast message from root(pts/3)(sat 09 14:37:282016):
#DISPLAYED ON
ALL LOGGED IN USERS.
```

There may be power failure.
Please save your files.

Q5. Explain sending of Electronic mails in UNIX?

- Sending and receiving messages using computer and communication tools is known as electronic mail or e-mail.
- The mail command is the basic e-mail program
- Contains text editor to compose email.
- Can be used to send as well as receive mails.

Syntax:

```
$mail <options> addresses
message text
```

Example:

\$mail user1user2

Subject:seminar

..... **Rama <ctrld>**

- User1anduser2 are loginnames.
- If the receiver is not busy running a program, the following message is displayed on his screen.
you have newmail.
- If the user is not logged in when mail is sent to him,thenmessageisdisplayed
You havemail

Receiving amail

The mailcommandwithoutargumentis used to receive mails.

- Example:

\$mail

Mail version.....

“/var/spool/mail/rama” :2 messages 1 new1 unread

u 1.<uma> mon mar 03 10:40 labs

>N 2.<std1> mon mar 04

12:3 projects

&

- Received mails of a user are stored in a **mailbox**.Name of this will be his/her login name.Mailbox is found in /var/spool/mail directory.
- First line displays version of mail program
- Second line gives a summary of messages , with their status such as unread and new.It also indicates mail directory being used and number of messages in it .

- Next list of mails are shown. First character on each line on gives status of each mail. Like new(N),unread(U),(>) character indicates that message as current message
- The “& “ character in the list line is the mail prompt.
- Several actions like reading, saving, deleting, forwarding and quitting the mail program.
- Personal mailbox called ‘mbox’ is located in user’s home directory. Any message not deleted ,but read ,will be saved in this file when user quits mail program.

Internal mailcommands

- <Enter> =>displays currentmail
- N =>displaysmailnumberedN.
- d =>deletesthecurrentmail.
- dn =>deletes mailn.
- u =>undeletes currentmail
- wfilename
=>savescurrentmessageinfilename.
- r =>reply to currentmail
- rn =>replytosenderofmailn.
- muser=>forwardsthemailtouser.
- q =>quits the mailprogram.
- - =>printpreviousmessage.
- + => print nextmessage.

SystemAdministration

Introduction

5 marks Questions

Q1.What are the ways to acquiring the super-user status in details with example?

- **There are two ways to acquire super user status**
 - 1.To log into console directory as root
 - 2.To execute the **su** command after logging in under a different username

1.Root as the super user login

- Unix provide a special login name called root for system admin
- This account need not be created since it is in built in every unix system
- The password is set at the time of installation

login : root

password : ***** # not displayed
#

2.The su command

- By executing the su command any user can acquire the status of super user if she knows the root password

\$/bin/su- password :

#

- After the execution of the su command the system displays a (#) pound sign showing the user now becomes super user
- The (-) after su changes the shell environment to superuser environment ie home directory is changed to root
- It is also known as **substituteuser** instead of super-user command

Q2. What are the **Duties of system Administrator?**

- Starting and shutting down the system
- User management (account adding)
- Disk space management
- Backup and restore
- Management of file system, Devices and Network services
- Monitoring system activity and security
- Responsibility of the user
- Hardware responsibilities
- Providing assistance to user whenever required

Q3. What are the **Super user privilege?**

- The super user is the most powerful user on the system.
- He has complete control over the entire system
- He can change the attribute of any file (permission)
- He can remove any command using **rm** command
- He can change the password
- He can set the system **date**
- He can send message using **wall** command to all the user
- He can limit the size of the file using **ulimit** command

Q4. What are the **restrictions on restricted shell-rbash?**

- A standard shell allows user to move around in the file system, execute commands, change environment variables and so on.
- Normal privileges to the user need to be restricted
- So rbash is allotted to him. It is an executable program similar to sh but with minimum privileges
- The user is allowed to work only in his home directory
- The user cannot change the **PATH** variable
- The user is not allowed to create new files or append to existing files

UserManagement

- The normal duties of the system administrator includes adding or deleting a user or a group from the system
- There are three types of accounts on a Unix system
 - Root account
 - System account
 - User account

Q5. What are the different ways of Startup and shutdown Unix operating system?

1 Startup of Unix operating system

- As soon as the power on the system looks all peripherals and next to complete boot cycle
- First important thing is loading into kernel -/kernel/genunix in solaris and /boot/vmlinuz in linux into memory
- The kernel spawns the init(PID1) daemon
- A Unix system can be booted to a specific mode which is represented by a number or a letter called run level

Two modes possible to start up the Unix

1. Single User Mode
2. Multiple User Mode

Single user mode is system admin to perform important task. In this mode other users are not allowed to operate the system

Multi user mode individual file system to be mounted and system daemon started

2. Shutdown of Unix operating System:

- The system admin shutdowns the system at the end of the day
- Abrupt switching off the system may lead to a problem
- So shutdown command is used for this purpose
 - Inform the user with wall command about shutting system
 - Send signal to all running process

b.Extracting files from archive

- To extract file from the archive tar uses x-option

```
$tar -xvf prgs.tar
```

c.Viewing the archive

- The content of the file can be viewed by -t option

```
$tar -tvf prgs.tar
```

2.Cpio command

- The cpio command copy input and output is used to copy files to and from backup device

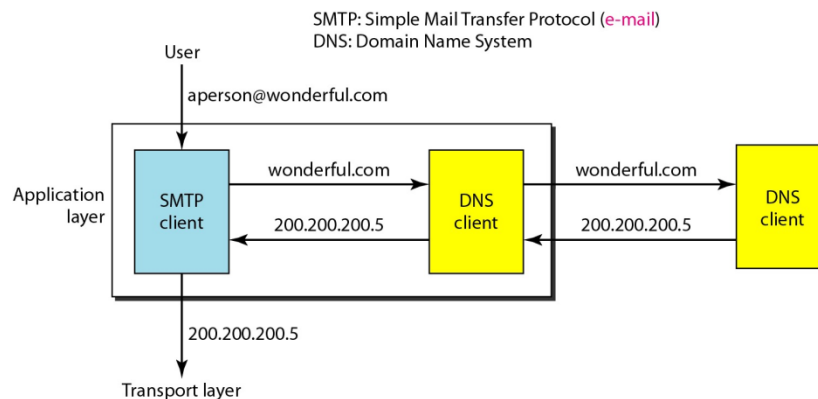
Option

- -o output
- To copy the current directory to 1.44 mb floppy
- `#ls | cpio -ov > /dev/fdoh1440`
- In above eg the files in current directory are piped to cpio through ls command and redirected to 1.44 mb floppy

DNS

Q7.What is Domain Name System Service(DNS)?What is its purpose.

Figure25.1 *Example of using the DNSservice*



DFS

Definition:

Implement a common file system that can be shared by all autonomous computers in a distributed system

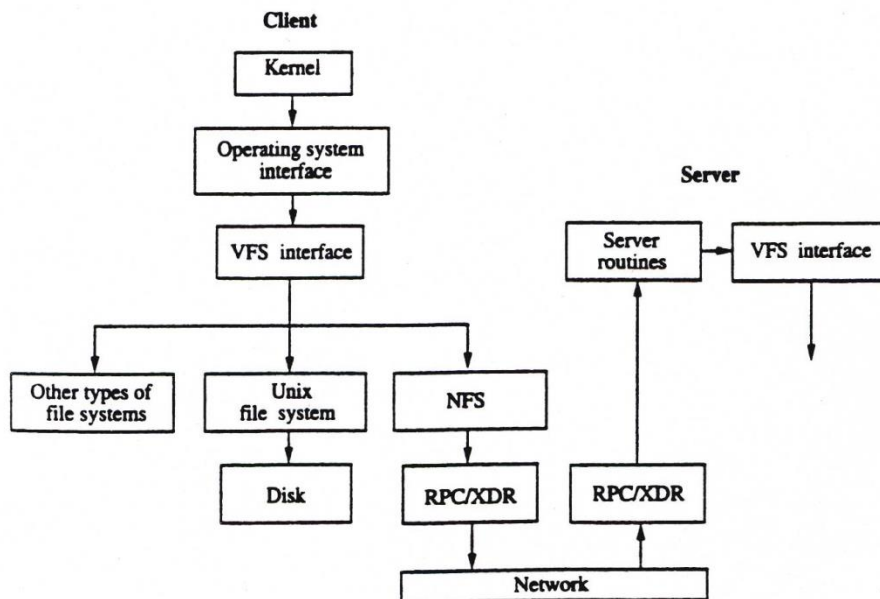
Goals:

Network transparency High availability

Q8. Write about Network File System (NSF)?

- Developed by Sun Microsystems to provide a distributed file system independent of the hardware and operating system
- Architecture
 - Virtual File System (VFS):
File system interface that allows NFS to support different file systems
 - Requests for operation on remote files are routed by VFS to NFS
 - Requests are sent to the VFS on the remote using
 - The remote procedure call (RPC), and
 - The external data representation (XDR)
 - VFS on the remote server initiates file system operation locally
 - *Vnode* (Virtual Node):
 - There is a network-wide *vnode* for every object in the file system (file or directory)-equivalent of UNIX *inode*
 - *vnode* has a mount table, allowing any node to be a mount node

Q.10 Write about NFS Architecture with neat diagram?



Naming and location: NFS (Cont.)

- Workstations are designated as clients or file servers
- A client defines its own private file system by mounting a subdirectory of a remote file system on its local file system
- Each client maintains a table which maps the remote file directories to servers
- Mapping a filename to an object is done the first time a client references the file. Example:

Filename: /A/B/C

- Assume 'A' corresponds to 'vnode1'
- Look upon 'vnode1/B' returns 'vnode2' for 'B' where 'vnode2' indicates that object is on server 'X'
- Client asks server 'X' to lookup 'vnode2/C'
- 'filehandle' returned to client by server storing that file

- Client uses *'filehandle'* for all subsequent operation on that file

Caching:NFS (Cont.)

Caching done in main memory of clients

- Caching done for: file blocks, translation of filename to *vnodes*, and attributes of files and directories

(1) Caching of file blocks

- Caching on demand with timestamp of the file (when last modified on the server)
- Entire file cached, if under certain size, with timestamp when last modified
- After certain age, blocks have to be validated with server
- Delayed writing policy: Modified blocks flushed to the server after certain delay

(2) Caching of filename to *vnodes* for remote directory names

- Speeds up the lookup procedure

(3) Caching of file and directory attributes

- Updated when new attributes received from the server, discarded after certain time

- Stateless Server

- Servers are stateless
 - File access requests from clients contain all needed information (pointer position, etc)
 - Servers have no record of past requests
- Simple recovery from crashes.