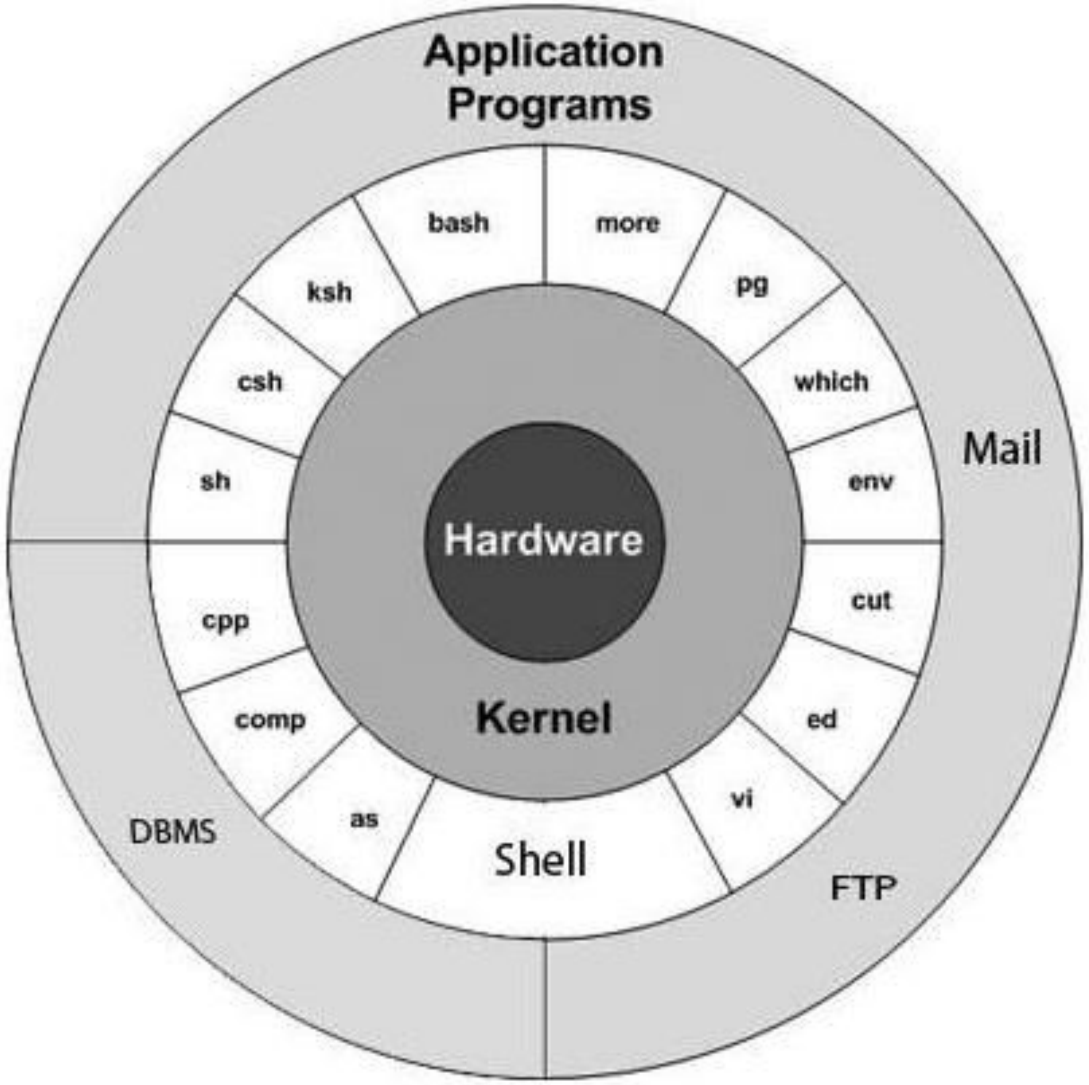
CSE305
COMPUTING PRACTICUM-II

OS Architecture



- Application Programs: (USER)
- Shell: Interprets the instructions/ commands typed by the user to machine language.
- Kernel: It interacts with hardware in machine language for managing resources. Has direct dealing with the hardware.

CHAPTER 1

ACCESSING THE COMMAND LINE

The bash shell

- A command line is a text-based interface which can be used to input instructions to a computer system.
- The Linux command line is provided by a program called the shell.
- The default shell for users in Red Hat Enterprise Linux is the GNU Bourne-Again Shell (bash).

CHAPTER 1

ACCESSING THE COMMAND LINE

The bash shell

- When a shell is used interactive, it displays a string when it is waiting for a command from the user.
- This is called the shell prompt. When a regular user starts as , the default prompt ends with a $ character.

```
[student@desktopX -]$
```

CHAPTER 1

ACCESSING THE COMMAND LINE

The bash shell

```
[student@desktopX ~]$
```

- 1. username
- 2. workstation of which that user is created/working.This is called the shell prompt.
- 3. ~ : home directory of the user, where user data is stored.
- 4. $: shell is ready to accept normal user commands.

When a regular user starts, the default prompt ends with a $ character.

CHAPTER 1

ACCESSING THE COMMAND LINE

The $ is replaced by a # if the shell is running as the super user, root .

- [root@desktopX ~]#

CHAPTER 1
ACCESSING THE COMMAND LINE

Shell basics

- Commands entered at the shell prompt have three basic parts:
- **Command** to run
- **Options** to adjust the behavior of the command
- **Arguments**, which are typically targets of the command

CHAPTER 1
ACCESSING THE COMMAND LINE

- Usage statements become much simpler to understand once a user becomes familiar with a few basic conventions :

- Square brackets, [ ] , surround optional items .

CHAPTER 1
ACCESSING THE COMMAND LINE

- Any thing followed by '...' represents an arbitrary-length list of items of that type.

- Multiple items separated by pipes, | means only one of them can be specified.
- Text in angle brackets, < > , represents variable data. For example, < file name> means "insert the file name you wish to use"

CHAPTER 1

Accessing the Command Line Using the Desktop

- The desktop environment is the graphical user interface on a Linux system . The default desktop environment in Red Hat Enterprise Linux is provided by GNOME 3.
- GNU Network Object Model Environment. (GNOME)

CHAPTER 1
Accessing the Command Line Using the Desktop

GNOME Shell is the graphical **shell** of the **GNOME** desktop environment with version **3**, which was released on April 6, 2011. It provides basic functions like launching applications, switching between windows and is also a widget engine.

- **GNOME** is part of the GNU project and part of the free software, or open source, movement.

- **GNOME** is a Windows-like desktop system that works on UNIX and UNIX-like systems and is not dependent on any one window manager

CHAPTER 1

Accessing the Command Line Using the Desktop

Parts of the GNOME Shell

These parts include the following :

1. top bar: The top bar provides the Applications and Places menus , and controls for volume, networking , calendar access, lock the screen, switch users, log out of the system , or shut it down etc.

2. Applications menu: to start applications, categorized into submenus. The Activities Overview can be started from this menu .

CHAPTER 1

Accessing the Command Line Using the Desktop

3. Places menu: This menu to the right of the Applications menu provides quick access through a graphical file manager to important menus in the user's home directory, to / , and to exports and file shares on the network.

4. window list: The bar that runs along the bottom of the screen. The window list provides an easy way to access, minimize, and restore all windows in the current workspace.

CHAPTER 1

Executing Commands Using the Bash Shell

Basic command syntax

- The GNU Bourne-Again Shell (bash) is a program that interprets commands typed in by the user.
- Each string typed into the shell can have upto thre e parts: the command , options (that begin with a - or --), and an arguments.

- If a user wants to type more than one command on a single line, a semi colon ; can be used as a command separator.

Contents

Shell Intro

- A system program that allows a user to execute:
  - shell functions (internal commands)
  - other programs (external commands)
  - shell scripts
- Linux/UNIX has a bunch of them, the most common are
  - `tcsh`, an expanded version of `csh` (Bill Joy, Berkley, Sun)
  - `bash`, one of the most popular and rich in functionality shells, an expansion of `sh` (AT&T Bell Labs)
  - `ksh`, Korn Shell
  - `zhs`
  - `...`

Command Format

- Format: command name and 0 or more arguments: `% commandname [arg1] ... [argN]`
- % sign means prompt here and hereafter.
- Arguments can be
  – options (switches to the command to indicate a mode of operation); usually prefixed with a hyphen (-) or two (--) in GNU style
  – non-options, or operands, basically the data to work with (actual data, or a file name)

Shell I/O

- Shell is a “power-user” interface, so the user interacts with the shell by typing in the commands.
- The shell interprets the commands, that may produce some results, they go back to the user and the control is given back to the user when a command completes.
- These system commands are often wrapped around a so-called system calls, to ask the kernel to perform an operation (usually privileged) on your behalf.

Command I/O

- Input to shell:
  - Command name and arguments typed by the user
- Input to a command:
  - Keyboard, file, or other commands
- Standard input: keyboard.
- Standard output: screen.
- These STDIN and STDOUT are often together referred to as a terminal.
- Both standard input and standard output can be redirected from/to a file or other command.
- File redirection:
  - < input
  - > output
  - >> output append

man

- Manual Pages
- The first command to remember
- Contains info about almost everything :-)
  - other commands
  - system calls
  - c/library functions
  - other utils, applications, configuration files
- To read about man itself type:

```
man man
```

date

Displays dates in various formats

- `date`
- `man date`

cal

- Calendar
  - for month
  - entire year
- Years range: 1 - 9999
- No year 0
- Calendar was corrected in 1752 - removed 11 days

`% cal`	current month
`% cal 2 2000`	Feb 2000, leap year
`% cal 2 2100`	not a leap year
`% cal 2 2400`	leap year
`% cal 9 1752`	11 days skipped
`% cal 0`	error
`% cal 2002`	whole year

clear

- Clears the screen
- There's an alias for it: Ctrl+L
- Example sequence:
  - `% cal`
  - `% clear`
  - `% cal`
  - `Ctrl+L`

sleep

- “Sleeping” is doing nothing for some time.
- Usually used for delays in shell scripts.
- `% sleep 2`   2 seconds pause

Command Grouping

- Semicolon: “;”
- Often grouping acts as if it were a single command, so an output of different commands can be redirected to a file:
- `% (date; cal; date) > out.txt`

alias

- Defined a new name for a command
- `% alias dt=“date”`

unalias

- Removes alias
- Requires an argument.
- `% unalias dt`

history

- Display a history of recently used commands
- `% history`
  - all commands in the history

exit / logout

- Exit from your login session.
- `% exit`
- `% logout`

shutdown

- Causes system to shutdown or reboot cleanly.
- May require superuser privileges
- `% shutdown –h now` - stop
- `% shutdown –r now` - reboot

ls

- List directory contents
- Has whole bunch of options, see man ls for details.
- `% ls`
  - all files except those starting with a "."
- `% ls -l`
- `% ls -a`

cat

- Display and concatenate files.
- `% cat`
  - Will read from STDIN and print to STDOT every line you enter.
- `% cat file1 [file2] ...`
  - Will concatenate all files in one and print them to STDOUT
- `% cat > filename`
  - Will take whatever you type from STDIN and will put it into the file `filename`
- To exit `cat` or `cat > filename` type Ctrl+D to indicate EOF (End of File).

Head/tail

Shows few entries of the data from a file

head /etc/passwd : first 10 entries by default

tail /etc/passwd : last 10 entries by default

touch

- By *touch* a file you either create it if it did not exists (with 0 length).
- Or you update it’s last modification and access times.
- There are options to override the default behavior.
- `% touch file`
- `Stat filename`
- `% man touch`

cp

- Copies files / directories.
- `% cp [options] <source> <destination>`
- `% cp file1 file2`
- `% cp file1 [file2] … /directory`
- Useful option: `-i` to prevent overwriting existing files and prompt the user to confirm.

mv

- Moves or renames files/directories.
- `% mv <source> <destination>`
  - The <source> gets removed
- `% mv file1 dir/`
- `% mv file1 file2`
  - rename
- `% mv file1 file2 dir/`
- `% mv dir1 dir2`

rm

- Removes file(s) and/or directories.
- `% rm file1 [file2] ...`
- `% rm -r dir1 [dir2] ...`
- `% rm -r file1 dir1 dir2 file4 ...`

script

- Writes a log (a typescript) of whatever happened in the terminal to a file.
- `% script [file]`
- `% script`
  - all log is saved into a file named typescript
- `% script file`
  - all log is saved into a file named file
- To exit logging, type:
  - `% exit`

mkdir

- Creates a directory.
- `% mkdir newdir`
- Often people make an alias of `md` for it.

cd

- Changes your current directory to a new one.
- `% cd /some/other/dir`
  - Absolute path
- `% cd subdir`
  - Assuming `subdir` is in the current directory.
- `% cd`
  - Returns you to your home directory.

pwd

- Displays personal working directory, i.e. your current directory.
- `% pwd`

`rmdir`

- Removes a directory.
- `% rmdir dirname`
- Equivalent:
  - `% rm -r dirname`

ln

- Symbolic link or a “shortcut” in M$ terminology.
- `% ln -s <real-name> <fake-name>`

chmod

- Changes file permissions
- Possible invocations
  - `% chmod 600 filename`
  - -rw------- 1 user group 2785 Feb 8 14:18 filename (a bit not intuitive where 600 comes from)
  - `% chmod u+rw filename` (the same thing, more readable)

which

- Displays a path name of a command.
- Searches a path environmental variable for the command and displays the absolute path.
- To find which `tcsh` and `bash` are actually in use, type:

```
which tcsh
which bash
```

whereis

- Display all locations of a command (or some other binary, man page, or a source file).
- Searchers all directories to find commands that match `whereis`’ argument
- `% whereis tcsh`

locate and find commands

```
locate  /bin/ls

find    /     [options]    /bin/bash
```

passwd

- Change your login password.

- It’s usually a paranoid program asking your password to have at least 6 chars in the password, at least two alphabetical and one numerical characters.

- Depending on a privilege, one can change user’s and group passwords as well as real name, login shell, etc.
- **`man passwd`**