CASE STUDY ON UNIX :

What is UNIX?

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. . It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. UNIX is highly portable across hardware since it is written in C language.

UNIX allows only needed modules to be loaded in memory (modularity).It has an inverted tree like file structure, with files and directories created within the file structure. Each file can be protected using read, write and execute permissions for the user, group and others(security).. UNIX uses TCP/IP protocol. CAD/CAM applications best perform in a UNIX system, with its varied support for graphic cards.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which are not covered by a graphical program, or when there is no windows interface available, for example, in a telnet session.

Types of UNIX

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.Here in the School, we use Solaris on our servers and workstations, and Fedora Core Linux on theservers and desktop PCs.

History of UNIX

- 1969: Ken Thompson, Dennis Ritchie started working on a multi-user OS on PDP-7,Bell Labs. 1970: OS named as UNIX
- 1973: OS rewritten in C
- 1975: First Version of Berkeley Software Distribution (BSD)
- 1982: AT&T announced UNIX System III, first public release.
- 1983: AT&T announced UNIX System V, the first supported release. Installed base45,000. 1984: Berkeley releases 4.2BSD, includes TCP/IP. X/Open formed
- 1984: System V Release 2 introduced. 1, 00,000 installations worldwide.
- 1986: 4.3BSD released, including internet name server. Installed base 2, 50,000.
- 1987: System V Release 3 introduced. Around 7, 50,000 installations.
- 1988: Open Software Foundation formed.
- 1989: System V Release 4 ships unifying System V, BSD . 1.2 million installations.

  its varied support for graphic cards.

Layered Architecture

UNIX is a layered operating system. The innermost layer is the hardware that provides the services for the OS. The operating system, referred to in UNIX as the kernel, interacts directly with the hardware and provides the services to the user programs. Most well written user programs are independent of the underlying hardware, making them readily portable to new systems.

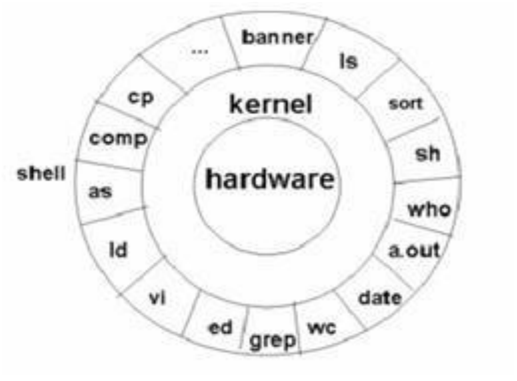


Fig 1.1

Layered Architecture of the UNIX System You can have many users logged into a system simultaneously, each running many programs. It's the kernel's job to keep each process and user separate and to regulate access to system hardware, including CPU, memory, disk and other I/O devices.

Aims and Objectives

- To use an example of a real OS (UNIX) to give context to the course
- UNIX is one of the most popular OSs on PCs to Mainframes
- *De Facto* standard for Open Systems

The Shell

- The shell is the interface between the command language user and the OS
- The shell is a user interface and comes in many forms (Bourne Shell, sh; Berkeley C Shell, csh; Korn Shell, ksh; Restricted Shell, rsh)
- User allowed to enter input when prompted ($ or %)
- System supports all shells running concurrently. Appropriate shell is loaded at login, but user can usually (except in sh, rsh) dynamically change the shell ▯ A UNIX command takes the form of

executable_file [-options] *arguments*

- The shell runs a **command interpretation loop** ○ accept command ○ read command ○ process command ○ execute command
- Executing the command involves creating a child process running in another shell (an environment within which the process can run). This is done by *Forking*.
- The parent process usually waits for the child to terminate before re-entering the command interpretation loop
- Programs can be run in the background by suffixing the command-line entry with an ampersand (&). Parent will not wait for child to terminate

The Processing Environment

Input and Output

- UNIX automatically opens three files for the process

STDIN - standard input (attached to keyboard)

STDOUT - standard output (attached to terminal)

STDERR - standard error (attached to terminal)

- Because UNIX treats I/O devices as special types of files, STDIO can be easily redirected to other devices and files

```
who > list _of _users
```

The Kernel

- Central part of the OS which provides system services to application programs and the shell
- The kernel manages processes, memory, I/O and the Timer - so this is not the same as the kernel that we covered in Lecture 3!
- UNIX supports multiprogramming
- Processes have their own address space - for protection
- Each process's *process environment* is composed of an unmodifiable re-entrant *text (code) region*, a modifiable *data region* and a *stack region*.
- The text region is shareable
- Processes can modify their environment only through calls to the OS

The File System

- UNIX uses HDS with *root* as its origin
- A directory is a special UNIX file which contains file names and their i-nodes (index nodes) Subdirectories appear as file entries
- Directories cannot be modified directly, but can are changed by the operating system when files and subdirectories are created and deleted
- File and Directory names must be unique within a particular directory (i.e., the path name must be unique)
- The File System is a data structure that is resident on disk
- It contains a *super block* (definition of the file system); an array of i-nodes (definition of the files in the system); the actual file data blocks; and a collection of free blocks Space allocation is performed in fixed-size blocks

The i-node

- Contains

  the file owner's user-id and group-id

  protection bits for owner, group, and

  world the block locator file size

  accounting information number of links to

  the file file type

The Block Locator

- Consists of 13 fields

- First 10 fields points directly to first 10 file blocks
- 11th field is an indirect block address
- 12th field is a double-indirect block address
- 13th field is a triple-indirect block address

Permissions

- Each UNIX file and directory has 3 sets of permission bits associated with it
- These give permissions for owner, group and world
- System files (inc. devices) are owned by root, wizard, or superuser (terminology!) Root has unlimited access to the entire installation - whoever owns the files!

Setuid

- When you need to change your password, you need to modify a file called /etc/passwd. But this file is owned by root and nobody other than root has write permission!
- The **passwd** command (to change passwords) is owned by root, with execute permission for world.
- The setuid is a bit which when set on an executable file temporarily gives the user the same privileges as the owner of the file
- This is similar in concept to some OS commands executing in Supervisor mode to perform a service for an otherwise unauthorised process

Process Management

- Description of Process Management in SunOS

Scheduling

- Priority-based pre-emptive scheduling. Priorities in range -20 to 20. Default 0.
- Priorities for runnable processes are recomputed every second
- Allows for ageing, but also increases or decreases process's priority based on past behaviour
- I/O-bound processes receive better service
- CPU-bound processes do not suffer indefinite postponement because the algorithm `forgets' 90% CPU usage in 5*n seconds (where n is the average number of runnable processes in the past 60 seconds)

Signals

- Signals are software equivalents to hardware interrupts used to inform processes asynchronously of the occurrence of an event

Interprocess Communication

- UNIX System V uses semaphores to control access to shared resources
- For processes to exchange data or communicate, *pipes* are used
- A pipe is a unidirectional channel between 2 processes
- UNIX automatically provides buffering, scheduling services and synchronisation to processes in a *pipe line*
- The presence of a pipe causes the processes in the pipe line to share STDIO devices

```
who | grep cstaff
```

- The output from **who** is directed to a buffer. **grep** will take its input from this buffer. The output from **grep** will be displayed on the terminal

Timers

- UNIX makes 3 interval timers available to each process
- Each counts down to zero and then generates a signal
- The first runs continuously
- The second runs while a process is executing process code
- The third runs while the process executes process code or kernel code

Memory Management

Address Mapping (Virtual Storage) - Paged MMS

- Virtual address V is dynamically translated to real address (P, D)
- Direct Mapping is used, with the Page Map held in a high-speed RAM cache
- Each Page Map Entry contains a *modified bit*, an *accessed bit*, a *valid bit* (if the page is resident in PM) and *protection bits*
- The system maintains 8 page maps - 1 for the kernel (not available to processes) and 7 for processes (contexts)
- 2 *context registers* are used - one points to the running process's page map and the other to the kernel's page map
- The replacement strategy replaces the page that has not been active for longest (LRU)

Paging

- SunOS maintains 2 data structures to control paging
- The *free list* contains empty page frames
- The *loop* contains an ordered list of all allocated page frames (except for the kernel)
- The *pager* ensures that there is always free space in memory
- When a page is swapped out (not necessarily replaced) the system judges whether the page is likely to be used again
- If the page contains a text region, the page is added to the bottom of the free list, otherwise it is added to the top
- When a page fault occurs, if the page is still in the free list it is *reclaimed*

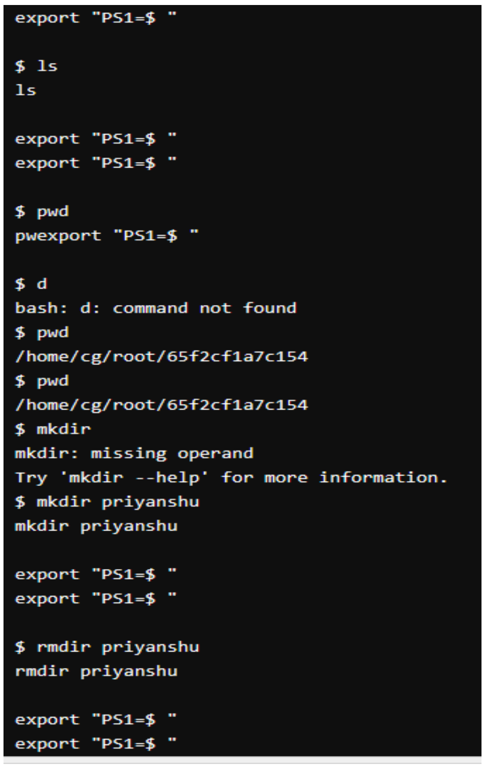
I/O

Data

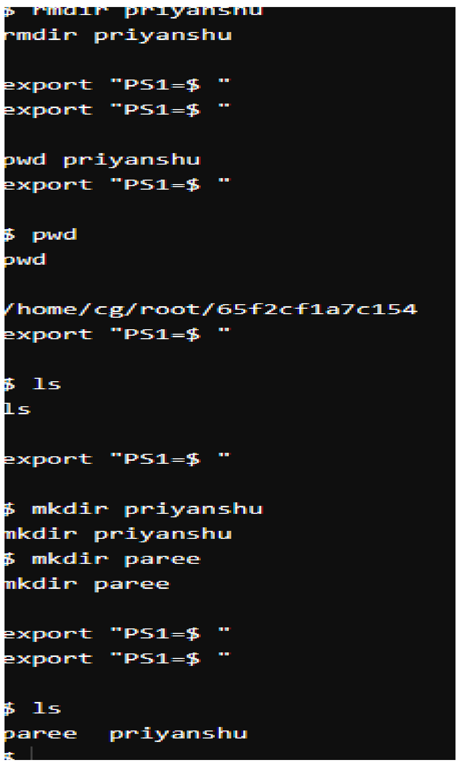
- All data is treated as a byte stream
- UNIX does not impose any structure on data - the applications do
- So data can be manipulated in any way - but programmers must explicitly structure the data

Devices

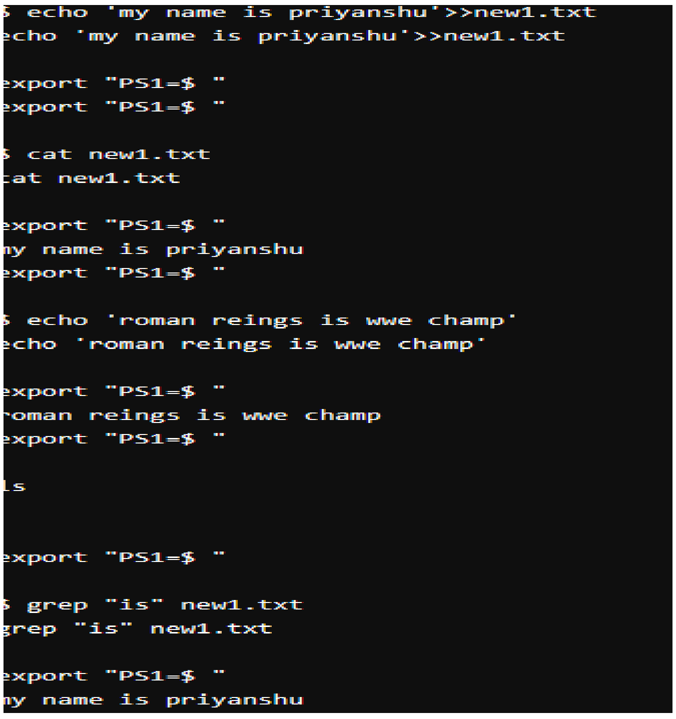
- A device is just a special type of file
- These files can have protection bits, so that a printer, e.g., cannot be read
- Permission to use sensitive devices, e.g., magnetic disk, is restricted to root and all other users have to use system calls to executable files which have their setuid bit set



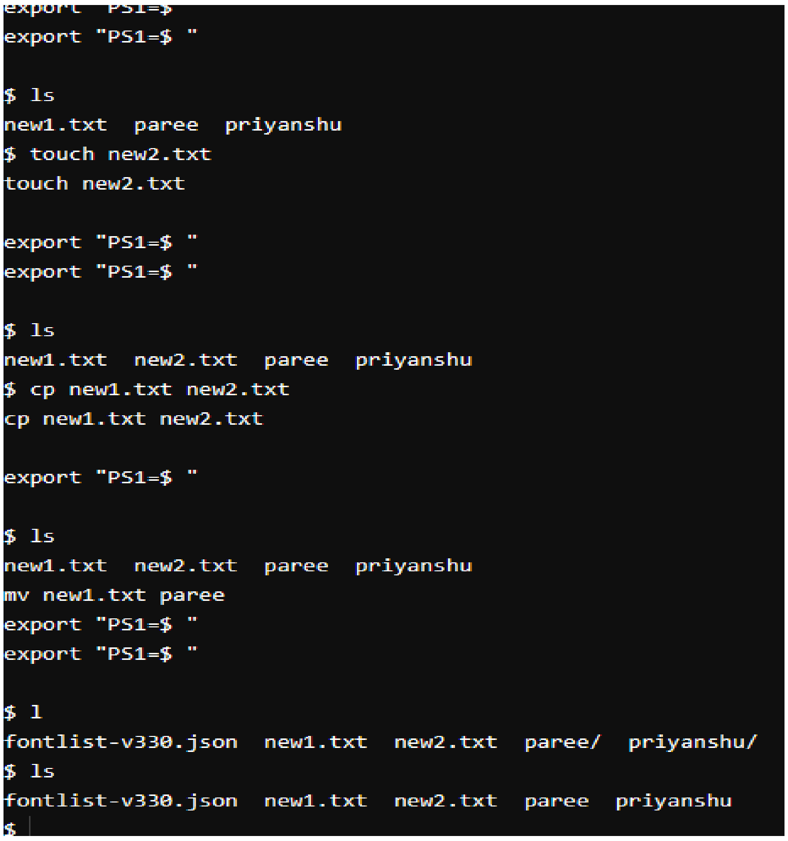
```
export "PS1=$ "

$ ls
ls

export "PS1=$ "
export "PS1=$ "

$ pwd
pwexport "PS1=$ "

$ d
bash: d: command not found
$ pwd
/home/cg/root/65f2cf1a7c154
$ pwd
/home/cg/root/65f2cf1a7c154
$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
$ mkdir priyanshu
mkdir priyanshu

export "PS1=$ "
export "PS1=$ "

$ rmdir priyanshu
rmdir priyanshu

export "PS1=$ "
export "PS1=$ "
```



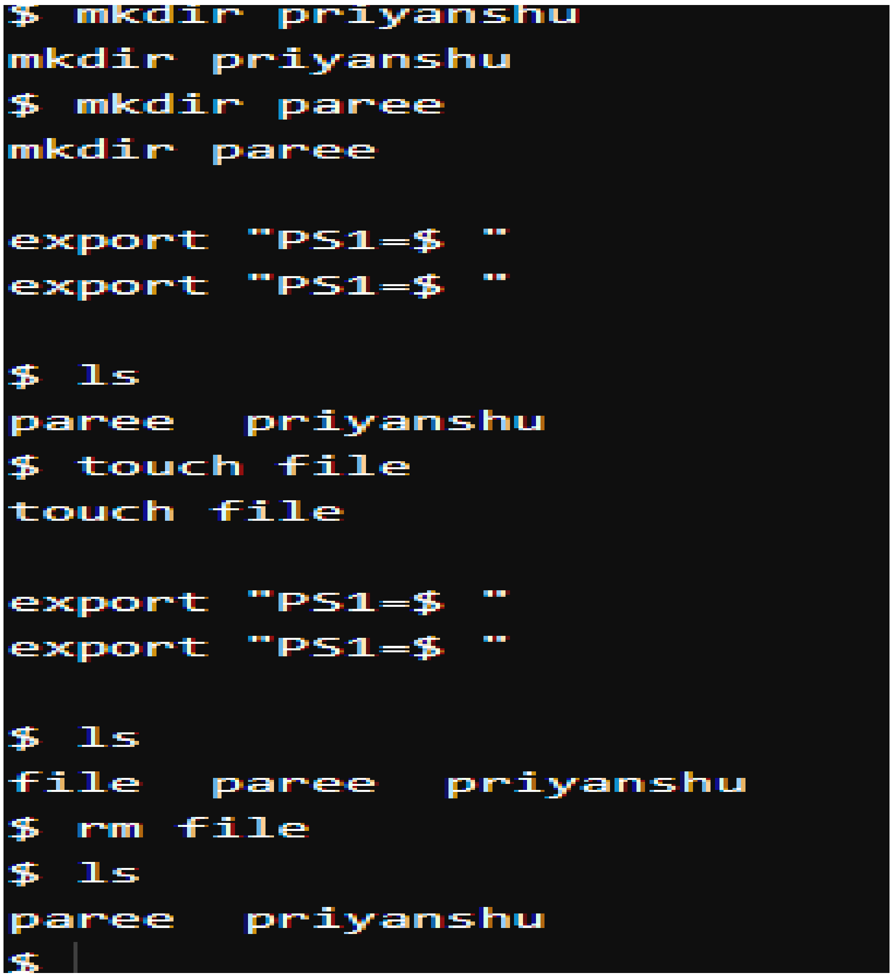
```
$ rmdir priyanshu
rmdir priyanshu

xport "PS1=$ "
xport "PS1=$ "

pwd priyanshu
xport "PS1=$ "

$ pwd
pwd

/home/cg/root/65f2cf1a7c154
xport "PS1=$ "

$ ls
ls

xport "PS1=$ "

$ mkdir priyanshu
mkdir priyanshu
$ mkdir paree
mkdir paree

xport "PS1=$ "
xport "PS1=$ "

$ ls
paree   priyanshu
$
```



```
$ echo 'my name is priyanshu'>>new1.txt
echo 'my name is priyanshu'>>new1.txt

xport "PS1=$ "
xport "PS1=$ "

$ cat new1.txt
at new1.txt

xport "PS1=$ "
ny name is priyanshu
xport "PS1=$ "

$ echo 'roman reings is wwe champ'
echo 'roman reings is wwe champ'

xport "PS1=$ "
oman reings is wwe champ
xport "PS1=$ "

ls


xport "PS1=$ "

$ grep "is" new1.txt
grep "is" new1.txt

xport "PS1=$ "
ny name is priyanshu
```



```
export "PS1=$
export "PS1=$ "

$ ls
new1.txt  paree  priyanshu
$ touch new2.txt
touch new2.txt

export "PS1=$ "
export "PS1=$ "

$ ls
new1.txt  new2.txt  paree  priyanshu
$ cp new1.txt new2.txt
cp new1.txt new2.txt

export "PS1=$ "

$ ls
new1.txt  new2.txt  paree  priyanshu
mv new1.txt paree
export "PS1=$ "
export "PS1=$ "

$ l
fontlist-v330.json  new1.txt  new2.txt  paree/  priyanshu/
$ ls
fontlist-v330.json  new1.txt  new2.txt  paree  priyanshu
$
```



```
$ mkdir priyanshu
mkdir priyanshu
$ mkdir paree
mkdir paree

export "PS1=$ "
export "PS1=$ "

$ ls
paree  priyanshu
$ touch file
touch file

export "PS1=$ "
export "PS1=$ "

$ ls
file  paree  priyanshu
$ rm file
$ ls
paree  priyanshu
$
```