Linux From User’s Perspective

Haresh Dagale
Electronic Systems Engineering, IISc

What is Linux?

- UNIX like Open-Source Operating System.
- Refers to kernel/Operating System/Desktop
- Comes in many flavours or *Distributions*

.deb based	.rpm based	others
• Debian	• Redhat EL	• Slackware
• Ubuntu	• CentOS	• Gentoo
• Linux Mint	• Fedora	• DSL
• Linspire	• Suse	• Puppy Linux
• Xandros	• Mandirava	

Linux Means Many Things

Linux operating system is built around Linux Kernel developed by **Linus Torvald** and other open-source developers.

Version 0.01 was released in Sept 1991.

Uses basic programs developed by GNU to make it a useable productive machine.

GNU: Free Software Foundation, founded by **Richard Stallman** in 1985

Multiple Linux Distributions started distributing completely integrated system software stack that is easy to install and use

Debian, Redhat, Arch, Slackware…

LINUX

Linux OS from User's Perspective

Applications

- Development Tools
- Communication Tools
  - Browser
  - Email Client
- Graphics & Multimedia Software
- Design Tools

System Management

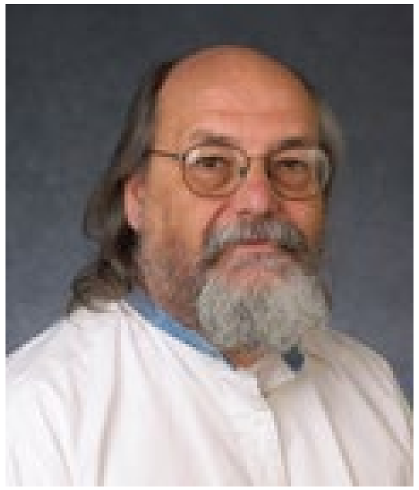
- Software Management
  - Add/Remove
- User Management
  - Add/Remove
- Security
  - Integrity
  - Confidentiality
  - Authenticity
- Privilege Levels
- File Systems

Runtime Management

- Hardware Management
  - CPU/Memory
- Process Management
- Network Stack
- Plug-n-play

Unix Origin

- Multi-user, Multi-process scalable OS of 70's
- Scalable and simple architecture made UNIX dominant OS for long time.
- Original AT&T Unix, whose development started in 1969[1] at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others
- University of California, Berkeley (BSD), Microsoft (Xenix), Sun Microsystems (SunOS/Solaris), HP/HPE (HP-UX), and IBM (AIX)
- POSIX and SUS Compliant Operating Systems
- In Unix "*Everything is a file*" *[stdout / stdin / stdin]*
- Initial Unix ran on PDP-11
- TCPIP integration and Socket Interface made it even more potent



Distributions

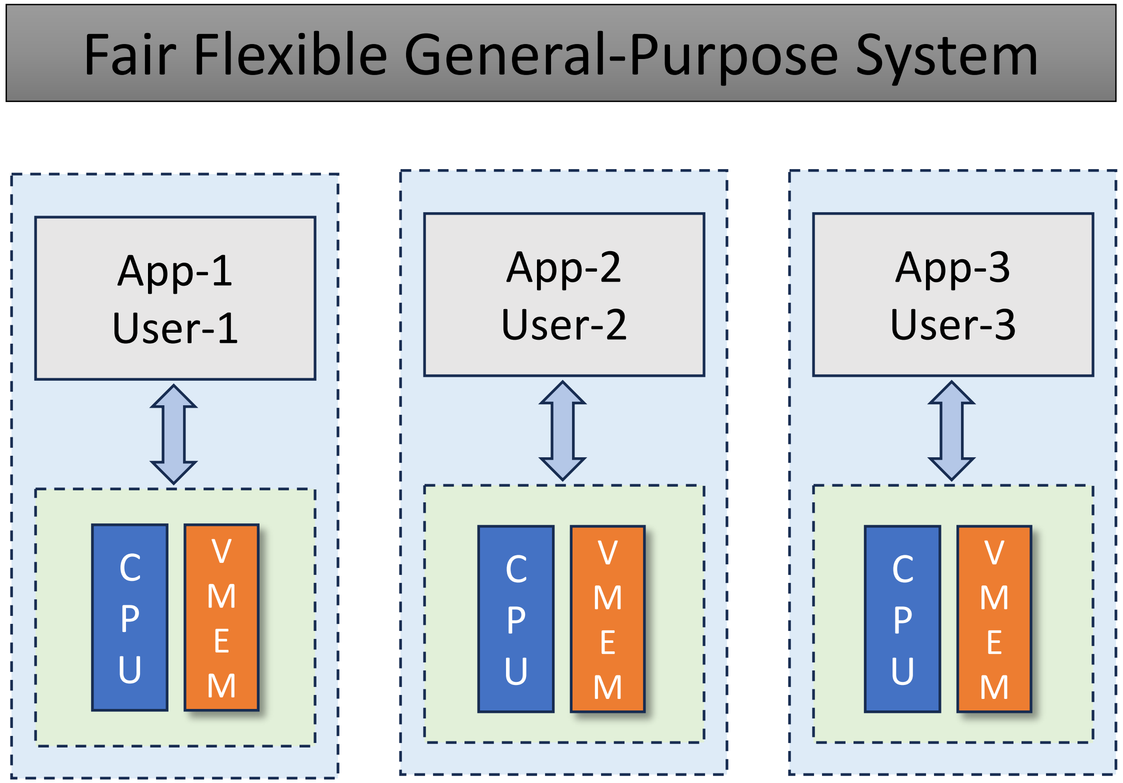
- How to manage the system?
- What default packages to installed or how to upgrade them?

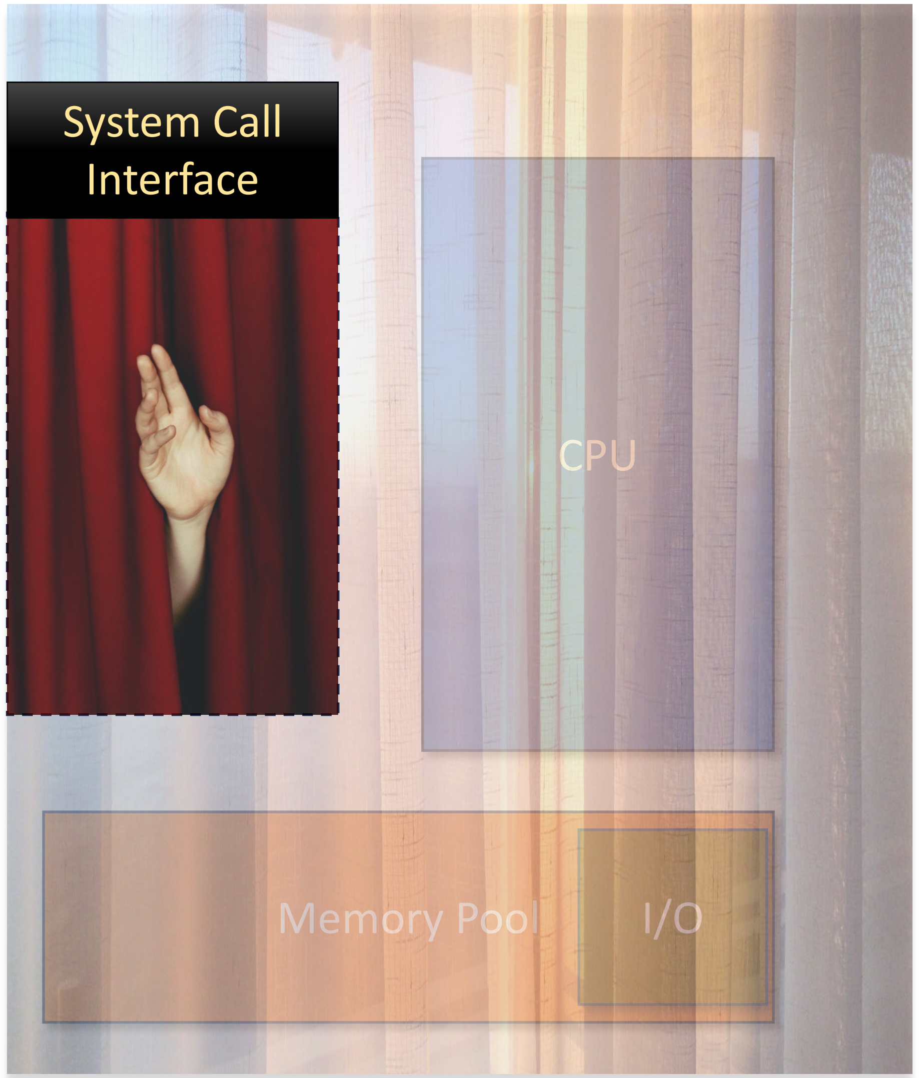
Manage system integrity, basically resolve library dependencies

Debian	Redhat	Arch
Apt/dpkg	Yum	Pacman
Version base	Version base	Rolling Type
Ubuntu/Mint/Raspien	Fedora/Suse/CentOS	Manjaro/ArcoLinux

LINUX

Virtualization in Desktop OS

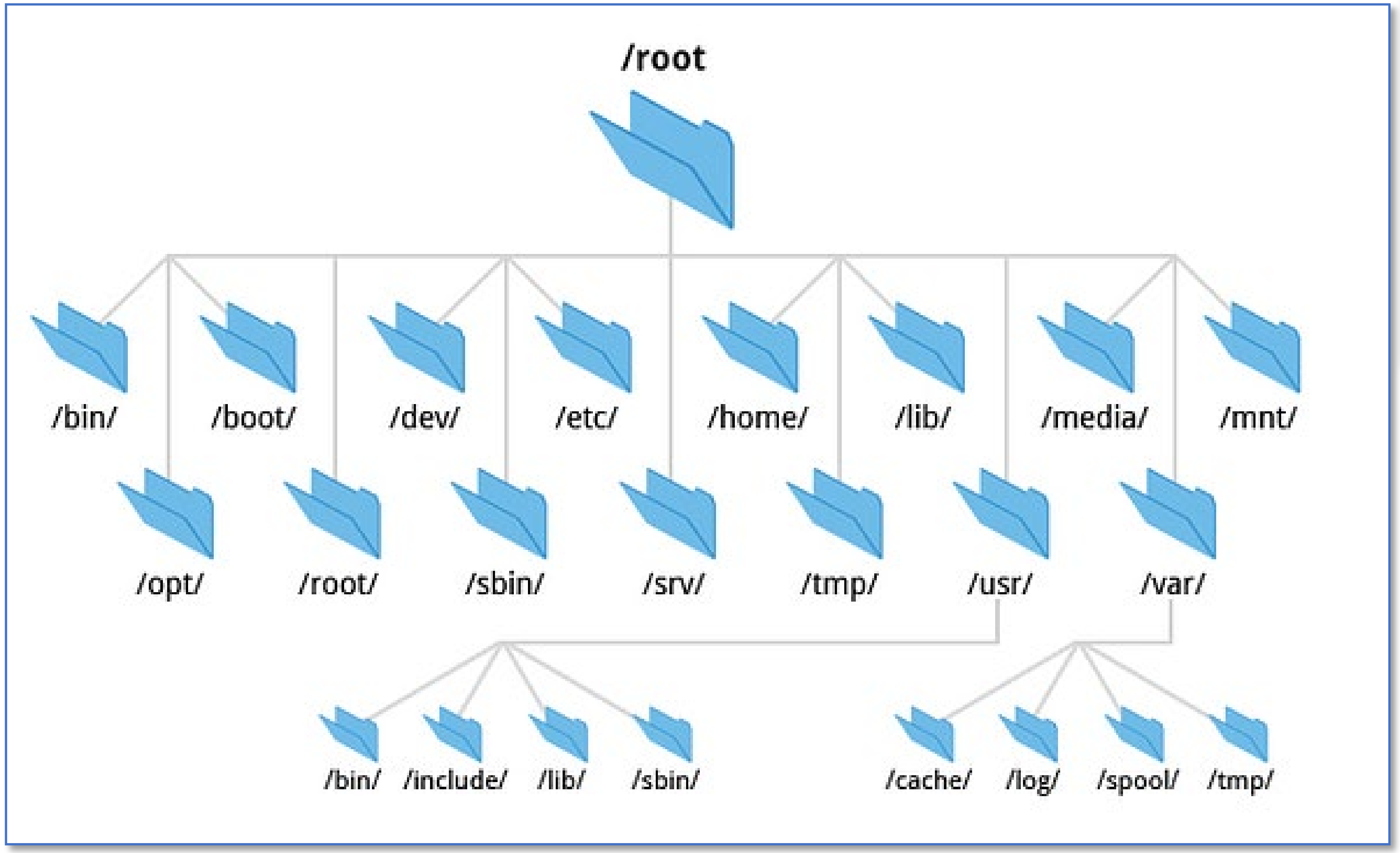


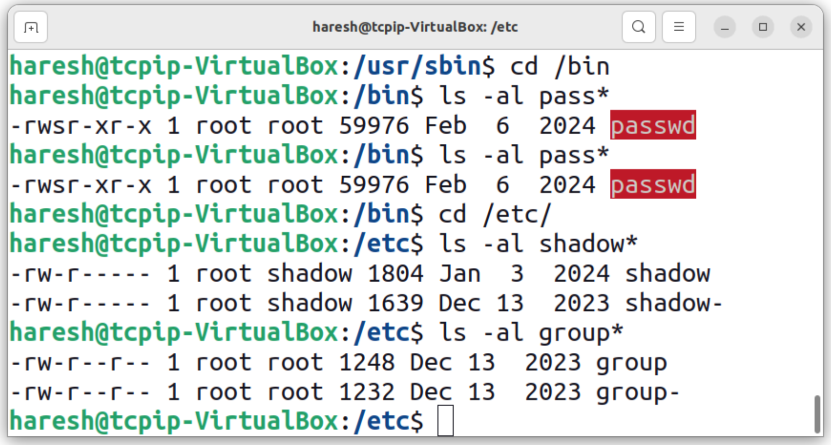


What makes Desktop OS non-realtime?

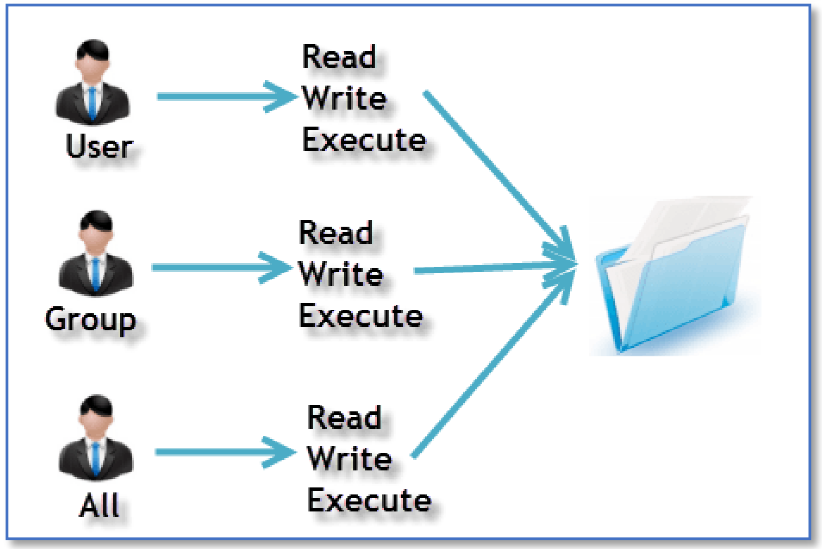
- Priorities of the tasks are not known before hand
- Scheduling depends on the number of active threads
- Involved non-deterministic disk accesses due to paging
- Interrupts are disabled during kernel operations with uncertain duration

File System Structure





```
haresh@tcpip-VirtualBox:/usr/sbin$ cd /bin
haresh@tcpip-VirtualBox:/bin$ ls -al pass*
-rwsr-xr-x 1 root root 59976 Feb  6  2024 passwd
haresh@tcpip-VirtualBox:/bin$ ls -al pass*
-rwsr-xr-x 1 root root 59976 Feb  6  2024 passwd
haresh@tcpip-VirtualBox:/bin$ cd /etc/
haresh@tcpip-VirtualBox:/etc$ ls -al shadow*
-rw-r----- 1 root shadow 1804 Jan  3  2024 shadow
-rw-r----- 1 root shadow 1639 Dec 13  2023 shadow-
haresh@tcpip-VirtualBox:/etc$ ls -al group*
-rw-r--r-- 1 root root 1248 Dec 13  2023 group
-rw-r--r-- 1 root root 1232 Dec 13  2023 group-
haresh@tcpip-VirtualBox:/etc$ 
```



Pipes and Redirection

‘>’ - Output of a program can be redirected to some file instead of stdout

'>>’ - Append output to the existing file

‘<‘ - Input to a program can be read from some file instead of stdin

‘|’ - Output of a program can be fed as an input to another program

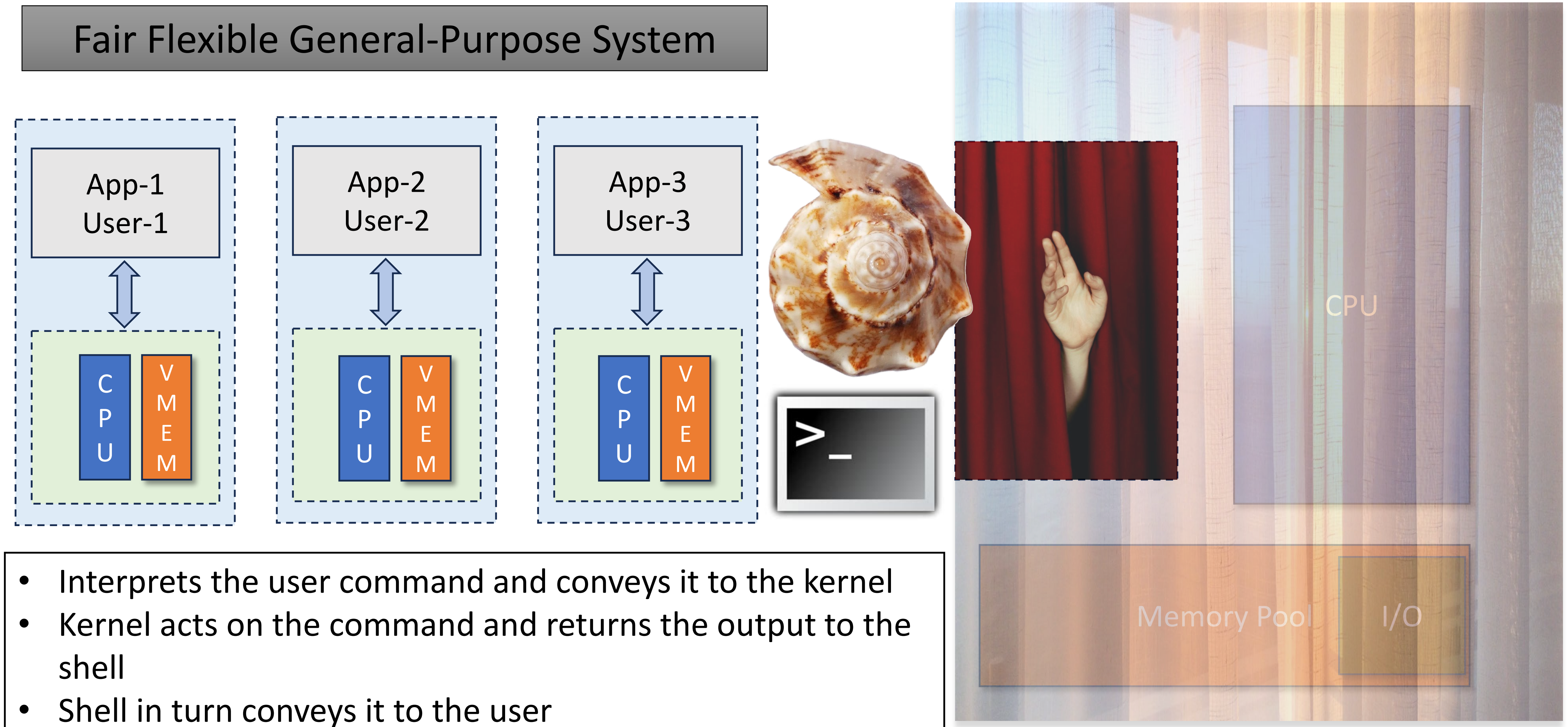
It is possible to connect two or more programs using pipes

```
$>who > temp

$>sort < temp

$>ls >> temp
```

Build program that only does one thing but in multiple ways and reliably.
Then combine these basic programs to achieve complex functionality

```
$>who | grep haresh

$>cc source.c 1 > &2 temp
```

Shell -- [BASH/CSH/KSH]



- Interprets the user command and conveys it to the kernel
- Kernel acts on the command and returns the output to the shell
- Shell in turn conveys it to the user

Shell Environment

```
DISPLAY=remote_host:0.0 ;
export DISPLAY

PATH="\$PATH:/home/user/b
in" ; export PATH
```

```
SHELL=/bin/bash
LANGUAGE=en_IN:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
PWD=/etc
LOGNAME=haresh
XDG_SESSION_DESKTOP=ubuntu-xorg
HOME=/home/haresh
LANG=en_IN
XDG_CURRENT_DESKTOP=ubuntu:
GNOMEVTE_VERSION=6800
TERM=xterm-256color
USER=haresh
PATH=/home/tcpip/anaconda3/bin:/home/haresh/.local/bin:/home/haresh/
bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/
usr/local/games:/snap/bin:/home/haresh/bin/:/home/haresh/ns/bake
GDMSESSION=ubuntu-xorg
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/bin
```

How Bash Executes User Command

Bash is running

Reads command “ls”

Bash calls fork()

Bash continues to run

- Calls “wait”
- Bash is waiting
- Comes out of wait once “ls” exits

Bash clone gets replaced by “ls”

- Exec(“/bin/ls”, …)
- Ls running
- Ls calls exit

Bash starts running again

What happens if bash is terminated when child is executing?

NOHUP signal

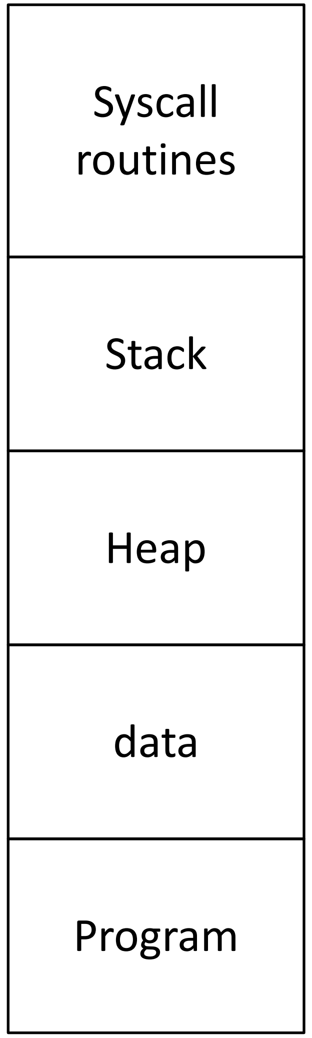
How To Start a New Program?

Process –

- A program with life! A program in execution (running) state.
- Needs memory for stack and to store data and program instructions.
- Runs in its own virtual world (environment).
- Competes for CPU time and other resources with other processes

Linux uses fork() and execve() to instantiate a new process

- Fork() replicates the given process. It is called once but returns twice.
- Execve() allows a running process to be replaced by new a program



Fork() example

```
int main(){
pid_t pid; int ret = 1; int status;

pid = fork();
if (pid == -1){

     // pid == -1 means error occurred
   printf("can't fork, error occurred\n");
   exit(EXIT_FAILURE);
}
else if (pid == 0){
           printf(“This is child process\n");
       while(1) {
                printf(“Do something new\n");
               }
       }
else {

       printf(“This is parent process\n");
       while(1) {
               printf(“Keep doing the old thing\n");
               }
```

Execve() Man Page

```
#include <unistd.h> int execve(const char *pathname,
char *const _Nullable argv[], char *const _Nullable
envp[]);
```

DESCRIPTION

execve() executes the program referred to by *pathname*. This causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.

Execv() Example

```
// the argv list first argument should point to
   // filename associated with file being executed
   // the array pointer must be terminated by NULL
   // pointer
   char * argv_list[] = {"ls","-lart","/home",NULL};


   // the execv() only return if error occurred.
   // The return value is -1
   execv("ls",argv_list);
   exit(0);
```

Difference between execv() and execve():
execve() allows additional process environment argument to be passed

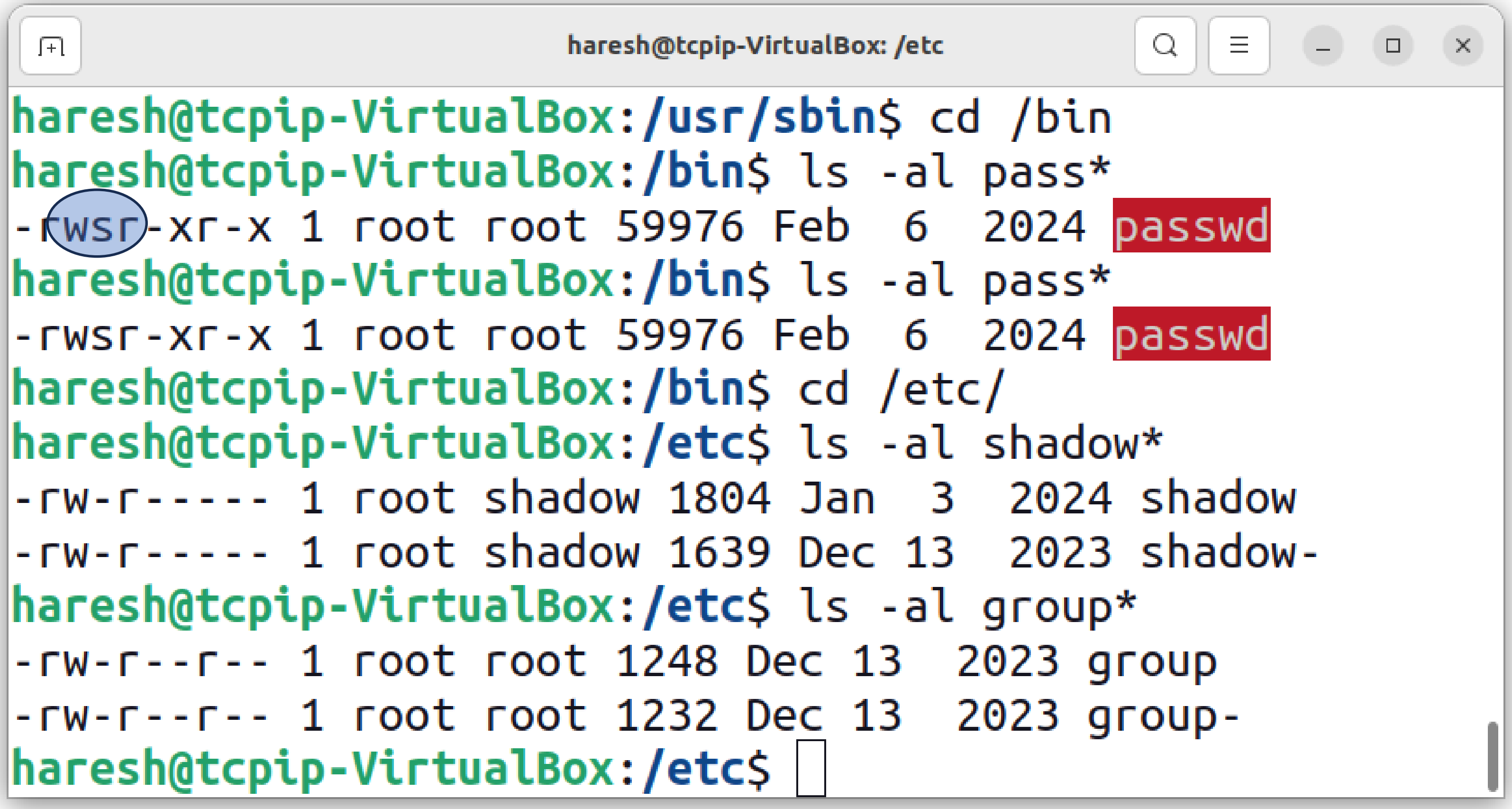
Runlevels

Linux has several modes of operation called 'runlevels’

Most common runlevels

- [0]: initial boot
- [1]: single user mode
  - used for troubleshooting and diagnostic purpose
  - minimal services, no networking
- [>2]: multi-user mode
  - Normal state, User can remotely login
- [5]: Graphical interface

Important Files



```
haresh@tcpip-VirtualBox:/usr/sbin$ cd /bin
haresh@tcpip-VirtualBox:/bin$ ls -al pass*
-rwsr-xr-x 1 root root 59976 Feb  6  2024 passwd
haresh@tcpip-VirtualBox:/bin$ ls -al pass*
-rwsr-xr-x 1 root root 59976 Feb  6  2024 passwd
haresh@tcpip-VirtualBox:/bin$ cd /etc/
haresh@tcpip-VirtualBox:/etc$ ls -al shadow*
-rw-r----- 1 root shadow 1804 Jan  3  2024 shadow
-rw-r----- 1 root shadow 1639 Dec 13  2023 shadow-
haresh@tcpip-VirtualBox:/etc$ ls -al group*
-rw-r--r-- 1 root root 1248 Dec 13  2023 group
-rw-r--r-- 1 root root 1232 Dec 13  2023 group-
haresh@tcpip-VirtualBox:/etc$
```

setuid bit

- /etc/fstab
- /etc/passwd
- /etc/shadow
- /etc/group
- /boot/grub/menu.lst
- /etc/inittab
- /etc/hosts

with Files

Basic Commands

- ls
- cd
- pwd
- cat

Coping and Renaming

- cp
- rm
- mv
- ln

Create and Delete directories

- mkdir
- rmdir

Search

- find
- locate

Useful Commands

- echo $HOME
- wc *filename*
- more *filename*
- less *filename*
- diff *file*1 *file*2
- touch *filename*
- file *filename*
- ping *remote ip address*
- lsof _ _
- wget *http : //url*

- dd if=/dev/hda of=mbr.image bs=512 count=1
- route add/del -net/-host *a.b.c.d* netmask 255.. gw *ip*
- netstat --tcp, --all, --statistics, --route

Managing Your System

Monitoring Processes

- ps aux
- pkill *matching_string*
- top

Disk Management

- du
- df
- fdisk and fsck

User Management

- adduser
- deluser

Software Package Management

- Debain/Ubuntu: apt-get, synaptic
- Fedora: yum

- A running program is called process
- Signalling to process

  kill -singal pid

Application Software

- User Interface

  X Server : X.org,
  Desktop: GNOME, KDE, XFce, LXDE, WindowMaker

- Office Software

  LibreOffice, LaTex, Openoffice, Koffice

- Internet Tools

  Web Server/browsers: Apache, Firefox, Konqueror

  Email: Postfix, Sendmail, qmail, Evolution, Thunderbird, Zimbra VoIP: Ekiga, LinPhone ...

- Multimedia Software

  Audio: Xmms, Rhythmbox,Amarok, Audacious Video: vlc, xine,totem, mplayer

- Graphic Tools

  Gimp, xfig, Scribus

System Software

- Developement Environment

  gcc, gdb, Eclipse, CodeBlock, VC++
- Virtual Machine Software xen, kvm
- Alien Software Support

  Wine, CrossOver Office
- Web Cache -- Squid
- Network Monitoring & Security

  iptables, Snort, MetaSploit, BurpSuite

init

- The first program to start and last to stop
- Initializes and terminates programs and services essential for system operations
- A Daemon process that runs during the entire lifetime of the system
- Ancestor process to all processes.
- Adopts orphaned process
- Kernel panics, If init does not start

Systemctl

Used to introspect and control system state of “systemd” and service manager. Used for activation, deactivation, enabling, disabling, restarting etc.

- ***systemctl status [unit]***: shows the status of unit.
- ***sudo systemctl stop [unit]***: deactivates a unit.
- ***sudo systemctl start [unit]***: activates a unit.
- ***sudo systemctl disable [unit]***: removes the symbolic link if it has an install section.
- ***sudo systemctl enable [unit]***: creates a symbolic link and it should be run before activating activating a unit else won’t run the systemd the install section.
- ***systemctl list-units***: lists all active units.
- ***sudo systemctl reload [unit]***: reloads a particular unit when there’s an edit in its configuration file.
- ***sudo systemctl restarts [unit]***: restart the unit by stopping and starting the unit. This method causes more disruption of service unlike ‘***systemctl reload [unit]***’ that only reloads the configuration file.
- ***sudo systemctl daemon-reload***: reloads all unit configuration files.

Pipes and Redirection

' >' Output of a program can be redirected to some file instead of stdout

' >>' Append output to the existing file

' <' Input to a program can be read from some file instead of stdin

'|' Output of a program can be fed as an input to another program

It is possible to connect two or more programs using pipes

- who > temp
- sort < temp
- ls >> temp

- who | grep mahesh
- cc source.c 1 > &2 temp

Administration

- su or su -
- fsck
- mkfs
- adduser
- userdel
- mii-tool
- ifconfig eth0 up/down
- ifup / ifdown
- sudo
- apt-get update — upgrade — install / remove [–purge] — *pkg$_n$ame*
- apt-cache search *matchstring*