28/12/22

Kernel, System Calls & init implementations

Boot Process

↳ 1st Stage -> BIOS -> Basic Input/Output System
↳ initializes the hardware & run some self-test to check hardware is good to go
-> main job is to load the bootloader

2) Bootloader 2nd Stage
↳ loads the Kernel into the memory based on kernel parameters our system might have

most common bootloader is -> grub bootloader

3rd Stage Kernel -> when the kernel is loaded it immediately initializes the memory & devices that are there in the system

-> main job is to load the init process or the mother process

4th Stage init/mother process
↳ it starts & stops all essential processes that run our system

* Linux can be classified into 3 levels of abstraction
1) Hardware -> CPU, memory, ports, hard disk etc

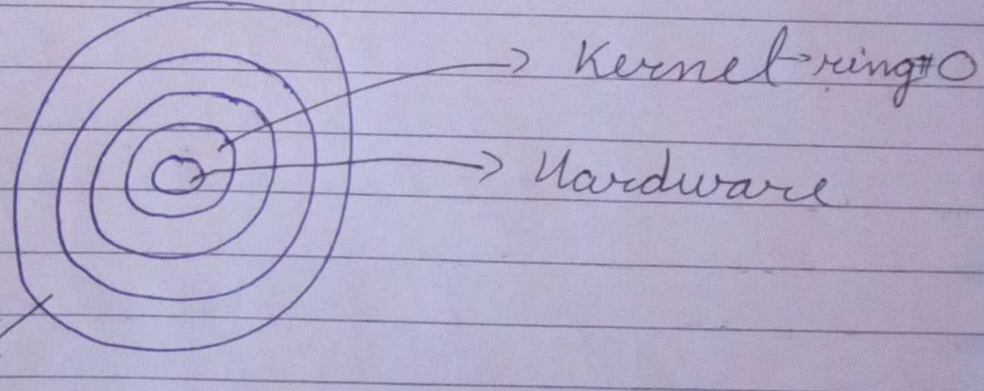
2) Kernel

2) Kernel
3) User Space

★ Privilege Levels -> also known as protection rings
- Kernel operates in Kernel mode
- User operates in user mode

Kernel mode has access the hardware & it controls everything in the system

Usermode is only allowed to access a very small amount of memory & CPU



★ System Calls
We (user) needs a system call to access the hardware
- let us perform some privileged instructions in Kernel mode
- (Kinda VIP pass)

-> also known as syscalls

3 major implementations of INIT/mother process

a) System V -> traditional way
b) Upstart
c) SystemD -> new standard

★ System V
- -> starts & stops processes sequentially
- advantage -> easier to resolve dependency issues
- disadvantage -> not fast as it doesn't ~~p~~ do multitask

System V Runlevels (0-6)

0 ->	shutdown
1 ->	single user
2 ->	multiuser connected locally
3 ->	multiuser connected via networking
4 ->	unused
5 ->	multiusers through networking & GUI
6 ->	reebooting

There are many services in system V
-> use them via-> sudo service 'name' start

ANKUR

(b) Upstart

↳ uses events & job model

events → messages recieved from other processes that trigers the job

↓ helps to respond to an event as soon as it happens

job → Actions performed

○ How upstart works

↓

first it loads up all the configs from /etc/init

↓

then when startup event occurs

↓

it runs all the jobs that were meant to be trigered by that event

c) System d Implementations

↳ uses goals to get system up & running

goals ↳ like targets

this imp. is flexible & don't do ~~s~~ sequentially to get all processes started

~~○ How it works~~

~~↓~~

~~loads up config files~~

Targets in Systemd

1) poweroff.target -> shutdown
2) rescue.target -> single user
3) multi-user.targe -> multiuser with network-ing
4) graphical.target -> '' '' '' & GUI
5) reboot.target -> for a reboot

-> also known as default.target