Course Name : PreCAT Crash Course – OC07
Subject Name : Operating System Concepts

Section-B => 9 questions are reserved for this subject
- 90%
- all questions are concept oriented – (no programming /no numericals)

# OS DAY-01:
Q. Why there is a need of an OS?

What is a Computer?
- Computer is a machine/hardware/digital device used to do diff tasks efficiently and accurately for user.

- Computer H/W mainly contains: Processor/CPU, Memory Devices & IO Devices etc...

- basic 4 functions of computer:
1. data storage
2. data processing
3. data movement
4. control

As any user cannot directly interacts with computer hardware, so there is a need of some interface between user and hardware, to provide this interface is the job an OS.

What is a software?
- collection of programs

Program:
- finite set of instructions written in any programming language given to the machine to do specific task.
- there are 3 types pf programs:
1. system programs: programs which are integral part of an OS (in built programs of an OS).
e.g. kernel, cpu scheduler, loader, dispatcher, memory manager etc....

2. application programs:
e.g. compiler, notepad, ms office, google chrome, calculator, games, IDE etc...

3. user programs: programs which are defined by programmer user
e.g. main.c, addition.cpp, program.java etc...

- program a passive entity, and process is an active entity
- if any program wants becomes active it must be loaded into the RAM
- program which is in the RAM is called as a process.
- program in execution is called a process
- running program is called as a process

- set of instructions written => program => HDD

- as any user cannot diretcly interacts with an OS, and hence an OS provides two types of interface for the user in the form of programs:
1. CUI/CLI: Command User Interface/Command Line Interface
- in this type of interface user can interacts with an OS by means of entering commands in a text format throgh command line.
Example:
In Windows: **Command Prompt** => **cmd.exe**
In Linux: terminal/shell

```
$gcc program.c => command to compile a program
$./program.out OR $.\a.exe => to execute a program
cp, mv, ls, clear, cls etc.....
```

2. GUI: Graphical User Interface
- in this type of interface user can interacts with an OS by means of making an events like single click, double click, click on buttons, exit, min, max, menu bar, menu list etc.....

In Windows : **explorer.exe**

task manager => search for a program explorer.exe => end task

In Linux: GNOME/KDE

#include<stdio.h>

- **header files** contains only declarations of library functions, and definitions of library functions are exists inside a lib folder in precompiled object module format.

- **stdio.h** file contains declarations of standard input output functions like printf( ), scanf( ) etc...

- RAM is also called as Main Memory?
- for an execution of any program RAM memory is must, and hence RAM is also called as main memory.
- loader – it is a system program (i.e. inbuilt program of an OS) which loads an executable file from HDD into the main memory.

- dispatcher – it is a system program (i.e. inbuilt program of an OS) which loads data & instructions of program which is in the main memory onto the CPU.

Sachin => SunBeam ==> SunBeam
Loader => OS ==> OS

Scenario-1:
Machine-1 : Linux   : program.c
Machine-2 : Windows : program.c => compile + execute => YES

portability: program written in C on one machine/platform can be compile and execute on any other machine/platform.

Scenario-2:
Machine-1 : Linux   : program.c => compile => program.out (executable code)

Machine-2 : Windows : program.out (executable code) => execute NO

- file format of an executable file in Linux is ELF (Executable & Linkable Format), whereas file format of an executable file in Windows is PE(Portable Executable).
File format => OS specific

file format is a specific way of an OS to keep/store data & instructions inside an executable file in an organized manner.

- for example: elf file format divides an executable file logically into sections:
1. elf header/primary header/exe header:

2. bss section(block started by symbol):
int g_num;//global var
static int num;

3. data section:
int g_num=99;//global var
static int num=999;

4. rodata section:

100 => int constant
1000L => long int constant
012 => octal constant
‘A’ => char constant
0x12 => hex constant
“SunBeam” - string literal

5. code section/text section
6. symbol table

- magic number – it is a constant number generated by the compiler which is file format specific (e.g. In Linux: ELF => magic number in Linux starts with ELF in its hexa decimal eq) => OS specific.

- when we execute a porgram, loader first verifies file format, if file format matches then only it checks magic number, and if file format as well as magic number both matches then only it loads an executable file from HDD into the main memory.

Q. What is an OS?
- An OS is a systsem software (i.e. collection of system programs) which acts as an interface between user nad hardware.
- An OS also acts as an interface between programs (application & user programs) & hardware.
- An OS allocates required resources like main memory, CPU time & IO devices access to all running programs, it is also called as resource allocator.
- to control an execution of all programs => OS
- to control hardware devices which are connected to the computer system => OS, and OS is also called as control program.
- an OS manages limited available resources among all running programs, it is also called as a resource manager.
- an OS is a software (i.e. collection of system programs & application programs which are in a binary format) comes with either in a CD/DVD/PD mainly has 3 components:
1. Kernel :

OS is Kernel OR Kernel is OS => basic minimal functionalities of an OS.

breathing =>

teaching =>

Kernel – crash => reinstall an OS
MS Office – crash => repair

speaking => basic minimal functionality

extra utility functionalities:
teaching
singing
to give speech

2. Utilitiy softwares
3. Application softwares:

=> installation of an OS: to install an OS onto the machine is nothing but to store OS software (i.e. collection of thousands of system programs and application programs which are in a binary format) onto the HDD.

=> if any OS want to becomes active, atleast its core program i.e. kernel must be loaded initially from HDD into the main memory,
process to load kernel from HDD into the main memory is called as booting, and this job is done by bootstrap program (it is exists into the HDD in first sector i.e. in a boot sector in first 512 bytes).
- while booting, bootstrap program locates the kernel and load it into the main memory, and kernel remains present into the main memory till we do not shutdown the system.

UNIX: basically designed for the developer by the developers

Windows => Desktop/Server => commercial OS => buisness
Linux => Desktop/Server: Open Source OS : source code kernel of that OS is freely available

vmlinuz – source code is freely available onto internet

Ubuntu – Linux distro
Redhat – Linux distro
Fedora Ubutu
Centos Linux

MAC OSX => MAC Machine by Apple
iOS => iPhone by Apple
UNIX
Android => mobile phones
Solaris => Server
etc...
thousands of OS's are available in market

Ken Thompson => B.E. Electricals
Denies Ritchie => M. Sc. Physics & Ind Maths

UNIX = B + BCPL + Assembly Language

C was invented while developement UNIX, in 1972, and in 1973 UNIX was rewritten in C => Portable => and hence UNIX can run from nailtop to super computer.

- Linux OS was invented by Linus Torvalds as his academic project in the University of Helsinki in the decade of 1990's by getting inspired from UNIX.

- Linux & UNIX these are 2 difefrent OS
- Linux is a UNIX like OS.
- Windows => Microsoft

Google =>

GK Question based on OS ==> Google/Wikipedia

Human Body System:
- Nervous System
- Excretory System
- Digestive System
- Respiratory System
etc...

OS:
- File Subsystem
- Process Control Subsystem: IPC, Memory Management & Scheduling
- Hardware abstraction system
- IO Devices Management Subsystem
- System Call Interface Block

- there are 2 major subsystems:
1. file subsystem
2. process control subsystem
- file & process are two very very imp concept in any OS
- In UNIX : file has space and process has life

- In UNIX whatever that can be stored is considered as a file, and whatever is in active state is considered as a process.

- In UNIX all devices are considered as a file/UNIX treats all devices as file:
- devices can be catgorised into 2 catagories:

Keyboard => hardware/input device
UNIX point of view => KBD => character special device file
Monitor => hardware/output device
UNIX point of view => Monitor => character special device file

HDD => memory device
UNIX point of view => HDD => block special device file

usually size of 1 sector = 512 bytes

HDD (file) ==> PD (file)

Human Body – Soul => Dead Body => passive => program/files
Human Body + Soul => living being => active

=> file subsystem
=> process control subsystem

calculator: program
main( ): client

services:
addition( )
division( )
substraction( )
multiplication( )

Kernel: Program
main( )
functions defined in it: system calls =>

- system calls are the functions in kernel program defined in c, c++ & assembly language which provides interface of services made available by the kernel for user.

In other words:
if any programmer user want to use services made available by the kernel in his/her program (i.e. in a user program), then it can be directly used by means of giving call directly to system calls or indirectly system calls can be called from inside library functions.

system developers: PGDESD

fopen( ) => open( ) : to open a file/to create a new file
fclose( ) => close( ) : to close a file
fprintf( )/printf( )/fwrite( )/fputs( )/fputc( ) => write( ) : to write data into the file (to print => stdout file which is associated with standard o/p device: monitor).
fscanf()/scanf()/fread()/fgets()/fgetc() => read() : read data from file

- In UNIX total 64 system calls are there
- In Linux more than 300 system calls are there
- In Windows more than 3000 system calls are there

fork( ) => to create a new process/child process
In UNIX => fork( )
In Linux => fork( ) / clone( )
In Windows => CreateProcess( )

- irrespective of any OS there 6 catagories of system calls:
1. file operations system calls: e.g. open(), close(), read(), write() etc...
2. device control system calls: e.g. open(), close(), read(), write(), ioctl() etc...
3. process control system calls: e.g. fork( ), _exit(), wait() etc...
4. inter process communication system calls: e.g. pipe( ), signal( ) etc...
5. accounting information system calls: e.g. getpid( ), getppid( ) etc...
6. protection & security system calls: e.g. chmod( ), chown( ) etc....

getpid( ) sys call returns pid of calling process
pid: process id – unique identifier of a process

getppid( ) sys call returns pid of parent of calling process