Memory Management

- Compiler convert code from high level language to low level langugae.
  - objdump -h main.out
  - readelf -h main.out
  - file main.out
- Compiler assumes a low config machine, while converting high level code to low level code called as "Virtual Machine".
  - e.g. gcc -m32 -o main.out main.c
    - Compiler assumes 80386 processor with 4 GB RAM.
- Compiler and Linker assign addresses to each variable/instruction assuming that program will execute in VM RAM. These addressed are called as "virtual addr" or "logical addr". The set of virtual addresses used by the process is referred "Virtual address space".
- However while execution these addresses might be occupied by other processes. Loader relocates all instructions/variables to the address avaible in RAM. The actual addresses given to the process at runtime are called as "physical addr" or "real addr". The set of physical addresses used by the process is referred "Physical address space".
- CPU always executes a process in its virtual addr space i.e. CPU always request virtual addresses (on addr bus).
- These virtual addresses are verified and then converted into corresponding physical addresses by a special hardware unit called as "Memory Management Unit (MMU)".
- Simple MMU holds physical base address and limit (length) of the process. The base & limit of each process is stored in its PCB and then loaded into MMU during context switch.
- The memory management of OS depends on MMU hardware.
  - Simple MMU --> Contiguous memory allocation
  - Segmentation MMU --> Segmentation
  - Paging MMU --> Paging
- If memory(RAM) is full, some area of disk can be used as extension of main memory called as "Swap area"/"Virtual memory". Inactive processes are swapped out to the swap area and RAM space is made available for new/active processes.
- OS may use swap area in form of swap partition or swap file.
  - Linux: Swap partition (given during installation)
  - Windows: Swap file (C:\pagefile.sys)
- Advantages of virtual memory
  - Can execute bigger programs (than main memory).
  - Can execute more number of programs.

File Management

File

- File is collection of data/information on storage device.
  - File = Contents (Data) + Information (Metadata)
  - The data is stored in zero or more Data blocks (in FS), while metadata is stored in the FCB (in filesystem).
- FCB is called as "inode" on UNIX/Linux. It contains

    - type: UNIX/Linux has 7 types of files
        - -: regular, d: directory, l: symbolic link, p: pipe, s: socket, c: char device, b: block device
    - size: number of bytes
    - links: number of hard links
    - mode (permissions): (u) rwx, (g) rwx, (o) rwx
    - user & group
    - time-stamps: modification, creation, access.
    - info about data blocks
- terminal> ls -l
    - type, mode, links, user, group, size, timestamp, name.
- terminal> stat filepath

File System

- Files are stored on storage device. Arrangement of files in storage device is called as "File System".
- e.g. FAT, NTFS, EXT2/3/4, ReiserFS, XFS, HFS, etc.
- File System logically divide partition into 4 sections.
    - Boot block/Boot sector
        - Contains programs/info required for booting of OS
        - Typically contains bootstrap program and bootloader program
    - Super block/Volume control block
        - Contains information of whole partition.
        - Capacity, Label.
        - terminal> df -h
            - Total number of data blocks/inodes.
            - Number of used/free data blocks/inodes.
            - Information of free data blocks/inodes.
    - Inode List/Master file table
        - Inodes (FCB) for each file
    - Data blocks
        - Stores data of the file.
        - Each file have zero or more data blocks.
        - Size of data blocks can be configured while creating file system

Disk/partition naming conventions

```
* Windows:
    * Disks are named as disk0, disk1, ...
    * partitions are named as drives i.e. C:, D:, E:, ...
* Linux:
    * Disks are named as /dev/sda, /dev/sdb, /dev/sdc, etc.
    * Partitions per disk are named as
        * sda partitions: sda1, sda2, sda3, ...
        * sdb partitions: sdb1, ...
```

User interfacing

- UI of OS is a program (Shell) that interface between End user and Kernel.
- Shell -- Commmand interpreter
  - End user --> Command --> Shell --> Kernel
- User interfacing (Shell)
  - Graphical User Interface (GUI)
  - Command Line Interface (CLI)

Example shells

- Windows
  - GUI shell: explorer.exe
  - CLI shell: cmd.exe, powershell.exe
- DOS
  - CLI shell: command.com
- Unix/Linux
  - CLI shell: bsh, "bash", ksh, csh, zsh, ...
    - ls /bin/*sh
    - echo $SHELL
  - shell of current user can be changed using "chsh" command.
- GUI shell/standards
  - GNOME: GNU Network Object Model Environment (e.g. Ubuntu, Redhat, CentOS, ...)
  - KDE: Kommon Desktop Environment (e.g. Kubuntu, SuSE, ...)
  - XFCE: XForms Common Environment (e.g. Raspberry Pi, ...

Hardware abstraction layer

- Most important feature of OS.
- It hides hardware details from end-user as well as from user programs.
- HAL Provides reusable code to interact with hardware.
- HAL layer of any OS is always arch depedent. Most of the code is written in assembly language.

Networking

- Networking feature enable computers (processes) to communicate with each other.
- Even though important (nowadays), networking is optional feature of OS.
- For networking computers are connected to each other in LAN, MAN or WAN.
- Computers are connected with different topologies e.g. bus, star, ring, mesh, ...
- Networking feature internally use "sockets" IPC mechanism.
- Socket is communication end-point.

Security and Protection

- Security is securing system from "external" threats e.g. virus, trojans, worms, hacking, etc.
  - Security is optional feature and is not implemented in many OS.
  - Security is usually provided by "Anti-virus" application.
  - Windows 10+ comes with Windows Defender, which is handling security aspects.
- Protection is protecting system (programs & files) from internal threats/elements.
  - Dual mode protection: CPU can differentiate whether code belongs to OS or user application.

    - IO protection: Only OS should be able to perform IO operations. User programs should use system calls to perform IO. This is feasible when IO instructions are privileged instructions (they can only be executed in kernel mode).
    - Memory protection: One process should not access memory of another process directly, so that one process cannot disturb execution of another process. This is implemented using MMU.'
    - CPU protection: If a process goes in infinite loop, the whole system should not hang. This is done using Timer hardware.

Types of commands

- Internal commands
    - commands are part of shell program only.
    - No seperate executable is present
    - eg. alias, unalias, cd
- External commands
    - commands are not part of shell program
    - Seperate executable is available in any one of the bin directory
    - eg. ls, cp, mv
- which
    - which command
        - display the location of command executable.
- whereis
    - whereis command
        - display the location of command executable and also manual paage location.