Programming Fundamentals and Operating System Concepts

- OS Concepts
- Linux commands
- Shell scripts
- Linux System call Programming

Evaluations

- Theory - 40 marks (CCEE)
- Lab - 40 marks (Linux commands and shell script)
- Internal - 20 marks

Learning OS

- step 1: End user
  - Linux commands
- step 2: Administrator
  - Install OS (Linux)
  - Configuration - Users, Networking, Storage, ...
  - Shell scripts
- step 3: Programmer
  - Linux System call programming
- step 4: Designer/Internals
  - UNIX & Linux internals

What is OS?

- Interface between end user and computer hardware.
- Interface between Programs and computer hardware.
- Control program that controls execution of all other programs.
- Resource manager/allocator that manage all hardware resources.
- Bootable CD/DVD = Core OS + Applications + Utilities
- Core OS = Kernel -- Performs all basic functions of OS.

OS Functions

- CPU scheduling
- Process Management
- Memory Management
- File & IO Management
- Hardware abstraction
- User interfacing
- Security & Protection
- Networking

Process Management

Program

- Set of instructions given to the computer --> Executable file.
- Program --> Sectioned binary --> "objdump" & "readelf".
    - Exe header --> Magic number, Address of entry-point function, Information about all sections. (objdump -h program.out)
    - Text --> Machine level code (objdump -S program.out)
    - Data --> Global and Static variables (Initialized)
    - BSS --> Global and Static variables (Uninitialized)
    - RoData --> String constants
    - Symbol Table --> Information about the symbols (Name, Size, section, Flags, Address) (objdump -t program.out)
- Program (Executable File) Format
    - Windows -- PE
    - Linux -- ELF
- Program are stored on disk (storage).

Process

- Progam under execution
- Process execute in RAM.
- Process control block contains information about the process (required for the execution of process).
    - Process id
    - Exit status
        - 0 - Indicate successful execution
        - Non-zero - Indicate failure
    - Scheduling information (State, Priority, Sched algorithm, Time, ...)
    - Memory information (Base & Limit, Segment table, or Page table)
    - File information (Open files, Current directory, ...)
    - IPC information (Signals, ...)
    - Execution context (Values of CPU registers)
    - Kernel stack
- PCB is also called as process descriptor (PD), uarea (UNIX), or task_struct (Linux).
- In Linux size of task_struct is approx 4KB

Memory Management

- Compiler convert code from high level language to low level langugae.
    - objdump -h 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