Operating System Concepts

Agenda

- VIM editor
  - ~/.vimrc file
- Shell scripts
  - float calculations
  - if-else statement
  - Relational and logical operators
  - File operators
  - case statement
  - loop statements
  - array
  - string operations
  - functions
  - positional params
- Booting
- Dual mode protection
- System calls
- Kernel types

Shell scripts

- Refer codes

Computer structure

- CPU: Genral purpose processor for program/OS execution
- Memory: RAM
- Storage: Disk

- IO: Keyboard, Monitor
- Connected by "bus".
- Each IO device has a "dedicated" "internal" processing unit -- IO device controller.

Computer IO (Input Output)

- Synchronous IO: CPU is waiting for IO to complete.
    - Hw technique: Polling
- Asynchronous IO: CPU is not waiting for IO to complete (doing some other task)
    - Hw technique: Interrupts
    - OS maintains a device status table to keep track of IO devices (busy/idle) and processes waiting for those IO devices.

Interrupt Processing

- IO event is sensed by IO device controllers.
- It will be conveyed to CPU as a special signal - Interrupt.
- CPU pause current execution and execute interrupt handler.
- "Interrupt handler" will get address of "ISR" (from IVT) and execute ISR.
- When ISR is completed, execution resumes where it was paused.

Hardware vs Software interrupt

- Hardware -- interrupts from hardware peripherals.
- Software interrupt
    - Special instructions (Assembly/Machine level) when executed, current execution is paused, interrupt handler is executed and then the paused execution resumes.
    - Arch specific:
        - 8085/86: INT
        - ARM 7: SWI
        - ARM Cortex: SVC
    - Also called as "Trap" in few architecture.

Interrupt Controller

- Convey the interrupts from various peripherals to the CPU.
- Also manage priority of the interrupt (when multiple interrupts arrives at same time).
- e.g. 8085/86 <-- 8259, Modern x86 processors (apic), ARM-7 (VIC), ARM-CM3 (NVIC), ...

System Calls

- Software interrupt is used to implement OS/Kernel services.
- Functions exposed by the kernel so that user programs can access kernel functionalities, are called as "System calls".
  - e.g. Process Mgmt: create process, exit process, communication, synchronization, etc.
  - e.g. File Mgmt: create file, write file, read file, close file, etc.
  - e.g. Memory Mgmt: alloc memory, release memory, etc.
  - e.g. CPU Scheduling: Change process priroty, change process CPU affinity, etc.
- System calls are specific to the OS:
  - UNIX: 64 syscalls e.g. fork(), ..
  - Linux: 300+ syscalls e.g. fork(), clone(), ...
  - Windows: 3000+ syscalls e.g. CreateProcess(), ...

Types of kernel

Monolithic Kernel

- Multiple kernel source files are compiled into single kernel binary image. Such kernels are "monolithic" kernels.
- Since all functionalities present in single binary image, execution is faster.
- If any functionality fails at runtime, entire kernel may crash.
- Any modification in any component of OS, needs recompilation of the entire OS.
- Examples: BSD Unix, Windows (ntoskrnl.exe), Linux (vmlinuz), etc.

Micro-kernel

- Kernel is having minimal functionalities and remaining functionalities are implemented as independent processes called as "servers".
  - e.g. File management is done by a program called as "file server".
- These servers communicate with each other using IPC mechanism (message passing) and hence execution is little slower.
- If any component fails at runtime, only that process is terminated and rest kernel may keep functioning.

- Any modification in any component need to recompile only that component.
- Examples: Symbian, MACH, etc.

Modular Kernel

- Dynamically loadable modules (e.g. .dll / .so files) are loaded into calling process at runtime.
- In modular systems, kernel has minimal functionalities and rest of the functionalities are implemented as dynamically loadable modules.
- These modules get loaded into the kernel whenever they are called.
- As single kernel process is running, no need of IPC for the execution and thus improves performance of the system.
- Examples: Windows, Linux, etc.

Hybrid Kernel

- Mac OS X kernel is made by combination of two different kernels.
- BSD UNIX + MACH = Darwin

Linux - OS Structure

Linux components

- Linux kernel has static and dynamic components.
- Static components are
  - Scheduler
  - Process management
  - Memory management
  - IO subsystem (core)
  - System calls
- Dynamic components are
  - File systems (like ext3, ext4, FAT)
  - Device drivers
- Static components are compiled into the kernel binary image.
  - They are kernel components.
  - The kernel image is /boot/vmlinuz.

- Dynamic components are compiled into kernel objects (*.ko files).
  - They are non-kernel components.
  - They are located in /lib/modules/kernel-version.

Booting

- Process from computer power on to OS startup.

Bootable device

- Stoarage device (disk, pen drive, CD/DVD, etc.) whose boot block contains bootstrap program, is said to be Bootable device.

Bootstrap program

```
* Bootstrap program can load OS kernel into RAM and start its execution.
* Bootstrap program is different for each OS each version.
* Bootstrap is located in first sector (512 bytes) of a disk/partition.
```

Bootloader program

- Bootloader program can be configured to show multiple boot options to end user.
- Depending on user selection, it runs Bootstrap program of corresponding OS.
- There are many important bootloader programs.
  - ntldr (Before Windows Vista) --> boot.ini
  - "bootmgr" (Windows Vista Onwards) --> bcd (boot config data)
    - (admin command prompt) cmd> bcdedit
  - LiLo --> Linux Loader
  - "GrUB" (Grand Unified Boot Loader) --> menu.lst or grub.cfg
  - BTX (BooT eXtended) --> BSD Unix
  - SILo (Sparc Interactive Loader) --> Solaris
  - Bootcamp --> Mac OS X

    - "uBoot" --> Linux bootloader for Embedded
- The Bootloader has some config file associated with it.

Bootstrap loader

- The set of programs fixed in Base ROM (Motherboard) is called as "Firmware".
    - BIOS (Basic Input Output System) -- Firmware developed for PC by IBM + MS.
    - EFI (Extensible Firmware Interface) -- Firmware developed for PC by Intel + HP.
- Bootstrap loader is a program from Base ROM.
- It find the bootable device as per "boot device priority" in the BIOS setup.
- Once bootable device is found, it starts its bootloader.

Booting steps

- Computer power on.
- Load firmware (BIOS or EFI) programs from base ROM into RAM.
- Run POST/BIST (from firmware).
- Run Bootstrap loader (from firmware) to find bootable device.
- Bootstrap loader loads bootloader program from bootable device.
- Bootloader program shows multiple options to end user and user select one of them.
- Bootloader program run corresponding bootstrap program.
- Bootstrap program load OS kernel into main memory and OS boot.