# Embedded Operating System

# Agenda

- OS can be categorized based on the target system (computers).
  - Mainframe systems
  - Desktop systems
  - Multi-processor (Parallel) systems
  - Distributed systems
  - Hand-held systems
  - Real-time systems

- OS Structure / Kernel Types
  - Simple structure
  - Layered structure
  - Monolithic kernel
  - Micro-kernel
  - Modular kernel
  - Hybrid kernel

# Classification of OS

- **OS can be categorized based on the target system (computers).**
  - Mainframe systems
  - Desktop systems
  - Multi-processor (Parallel) systems
  - Distributed systems
  - Hand-held systems
  - Real-time systems

# Mainframe systems

- Examples: UNIX and its flavours, IBM-360, etc.
- **Resident Monitor**
  - Early (oldest) OS resides in memory and monitor execution of the programs. If it fails, error is reported.
  - OS provides hardware interfacing that can be reused by all the programs.
- **Batch Systems**
  - The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it.
  - The programs are executed one after another.
  - In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.

- **Multi-Programming**
  - In multi-programming systems, multiple program can be loaded in the memory.
  - The number of program that can be loaded in the memory at the same time, is called as "degree of multi-programming".
  - In these systems, if one of the process is performing IO, CPU can continue execution of another program.
  - This will increase CPU utilization.
  - Each process will spend some time for CPU computation (CPU burst) and some time for IO (IO burst).
  - If CPU burst > IO burst, then process is called as "CPU bound".
  - If IO burst > CPU burst, then process is called as "IO bound".
  - To efficiently utilize CPU, a good mix of CPU bound and IO bound processes should be loaded into memory.
  - This task is performed by an unit of OS called as "Job scheduler" OR "Long term scheduler".
  - If multiple programs are loaded into the RAM by job scheduler, then one of process need to be executed (dispatched) on the CPU.
  - This selection is done by another unit of OS called as "CPU scheduler" OR "Short term scheduler".

- **Multi-tasking OR time-sharing**
  - CPU time is shared among multiple processes in the main memory is called as "multi-tasking".
  - In such system, a small amount of CPU time is given to each process repeatedly, so that response time for any process < 1 sec.
  - With this mechanism, multiple tasks (ready for execution) can execute concurrently.
  - There are two types of multi-tasking:
    - **Process based multitasking**:
      - Multiple independent processes are executing concurrently.
      - Processes running on multiple processors called as "multi-processing".
    - **Thread based multi-tasking OR multi-threading:**
      - Multiple parts/functions in a process are executing concurrently.

- **Multi-user**
  - Multiple users can execute multiple tasks concurrently on the same systems.
  - e.g. IBM 360, UNIX, Windows Servers, etc.
  - Each user can access system via different terminal.
  - There are many UNIX commands to track users and terminals.
    - tty (teletype) : It prints the name of the current terminal
    - who : Information about currently logged in users, system boot time, run level,processes,…
    - who am i : Gives you the name of the current user, the terminal they are logged in at, the date and time when they logged in.
    - whoami : It gives username of the current user
    - w : Displays the users.

# Desktop systems

- Personal computers -- desktop and laptops
- User convenience and Responsiveness
- Examples: Windows, Mac, Linux, few UNIX, ...

# Multiprocessor systems

- The systems in which multiple processors are connected in a close circuit is called as "multiprocessor computer".
- The programs/OS take advantage of multiple processors in the computer are called as "Multi-procssing" programs/OS.
  - Windows Vista: First Windows OS designed for multi-processing.
  - Linux 2.5+ : Linux started supporting multi-processing.
  - terminal> uname -a
- Since multiple tasks can be exeuted on these processors simultaneously, such systems are also called as "parallel systems".
- Parallel systems have more throughput (Number of tasks done in unit time).
- There are two types of multiprocessor systems:
  - Asymmetric Multi-processing
  - Symmetric Multi-processing

- **Asymmetric Multi-processing**
  - OS treats one of the processor as master processor and schedule task for it.
  - The task is in turn divided into smaller tasks and get them done from other processors.

- **Symmetric Multi-processing**
  - OS considers all processors at same level and schedule tasks on each processor individually.
  - All modern desktop systems are SMP.

- **Distributed systems**
  - Multiple computers connected together in a close network is called as "distributed system".
  - Its advantages are high availability (24x7), high scalability (many clients, huge data), fault tolerance (any computer may fail).
  - The requests are redirected to the computer having less load using "load balancing" techniques.
  - The set of computers connected together for a certain task is called as "cluster".
  - Examples: Linux

- **Handheld systems**
  - OS installed on handheld devices like mobiles, PDAs, iPODs, etc.
  - Challenges:
    - Small screen size
    - Low end processors
    - Less RAM size
    - Battery powered
  - Examples: Symbian, iOS, Linux, PalmOS, WindowsCE, etc.

- **Realtime systems**
  - The OS in which accuracy of results depends on accuracy of the computation as well as time duration in which results are produced, is called as "RTOS".
  - If results are not produced within certain time (deadline), catastrophic effects may occur.
  - These OS ensure that tasks will be completed in a definite time duration.
  - Time from the arrival of interrupt till begin handling of the interrupt is called as "Interrupt Latency".
  - RTOS have very small and fixed interrupt latencies.
  - RTOS Examples: uC-OS, VxWorks, pSOS, RTLinux, FreeRTOS, etc

- **Simple OS structure**
  - Small Operating systems like MS-DOS or few embedded OS follow a very simple structure.
  - DOS operating system is made up of three files only.
    - COMMAND.COM <- command interpreter
    - MSDOS.SYS <- kernel
    - IO.SYS <- device drivers

- **Layered structure (architecture)**
  - OS is divided into multiple layers, so that each layer depends on the lower layer and provide functionality to the upper layer.
  - Example: Windows, UNIX, Linux, etc.
  - Windows OS have following layers
    - applications system call APIs
    - system call implmenetation
    - Kernel Executive : File Mgr, Memory Mgr, Process Mgr, Scheduler, Thread Mgr, etc.
    - IO Subsystem
    - Device Drivers
    - Hardware Abstraction Layer

- **Monolithic Kernel**
  - Multiple kernel source files are compiled into single kernel binary image.
  - Such kernels are "mono-lithic" 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 are 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
  - Static components (like monolithic kernel*)
    - Scheduler, Memory manager, IO subsystem, System calls, Process/thread management, etc.
    - All of these are compiled into single binary kernel image -- "vmlinuz".
      - /boot/vmlinuz
      - terminal> ls /boot
    - If any of these components need to modify, whole kernel should be recompiled.
  - Dynamic components (like modular kernel) File system managers, Device drivers, etc.
    - All of these are compiled into .ko (kernel object) files
      - /lib/modules//*
    - terminal> ls /lib/modules/
    - terminal> ls /lib/modules/uname -r
    - terminal> find /lib/modules/uname -r -iname "*.ko"
    - If any of these components need to modify, only that component needs to be recompiled and added into kernel again.

# Linux kernel compilation Steps

1. Download Linux kernel from www.kernel.org.
2. Extract Linux kernel using "tar" command.
3. Configure the Linux kernel.
    1. Which components to be compiled as static and which one to be compiled as dynamic.
    2. Configure params like buffer size, timer interval, etc.
    3. terminal> make menuconfig
4. Compile all static components and build kernel image.
    1. terminal> make bzImage
5. Compile all dynamic components.
    1. terminal> make modules
6. Copy dynamic components into /lib/modules.
    1. terminal> sudo make modules_install
7. Copy kernel image into /boot.
    1. terminal> sudo make install
8. Reboot the system and select new kernel to boot

# Booting

- Process from computer power on to OS startup
- Terminologies :
- Bootstrap Program
  - It loads OS kernel into main memory
  - Each OS has Its own Bootstrap program.
  - Located in first sector of bootable storage device
- Bootable Device
  - Storage device whose first sector (512 bytes) contains a special program “Bootstap Program “.
  - Usually it is device that stores OS installation setup;
  - E.g. Bootable CD/DVD ,Bootable Pendrive.
- Bootloader
  - When multiple OS are installed on single computer, at the beginning one program asks end user about which OS boot.
  - This task perform by bootloader program.
  - Bootloader program bootstrap program of selected OS.
  - Located in second sector of bootable storage device.

- BIOS/Firmwaer
  - Firmware is a program/set of a program loaded into base ROM of motherboard.
  - It is developed by Manufacturer of motherboard.
  - The firmware developed for PC (by IBM) is named as BIOS : Basic Input Output System
  - BIOS Contain
    - • POST/BIST
    - Bootstrap loader
    - Information utility
    - Bootable Device preference/settings
    - Basic/Minimal device driver.

- POST/BIST
  - POST - Power ON Self Test
  - BIST - Built In Self Test
  - Send signal to all peripheral(e.g. keyboard , mouse, monitor…)and test if they are functioning well.
  - Located in Base ROM(Part of Firmware)

- Bootstrap Loader :
  - Finds the bootable device in the computer and start it’s Bootloader.
  - It check all devices in a order given in BIOS and start the first found bootable device.
  - Located in Base ROM (Part of Firmware).

# Operating Systems Concepts

**# Booting:**

- There are two steps of booting:

**1. Machine Boot:**

**Step-1:** when we switched on the power supply current gets passed to the motherboard on which from ROM memory one micro-program gets executes first called as **BIOS(Basic Input Output System).**

**Step-2:** first step of BIOS is **POST(Power On Self Test),** under POST it checks wheather all peripheral devices are connected properly or not and their working status.

**Step-3:** After POST it invokes **Bootstrap Loader** programs, which searches for available **bootable devices** presents in the system, and it selects only one bootable device at a time as per the priority decided in BIOS settings.

**2. System Boot:**

**Step-4:** After selection of a bootable device (budefault HDD), **Bootloader Program** in it gets invokes which displays list of names operating systems installed on the disk, from which user need to select any one OS.

**Step-5:** Upon selection of an OS, **Bootstrap Program** of that OS gets invokes, which locates the kernel and load into the main memory


# Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com