Operating System Services

- An operating system provides an environment for the execution of programs.
- Different operating systems are organized along different lines.
- As such, specific services may differ, but common classes exist:
    - Those that provides convenience for user/programmer
    - Those that ensure efficient operations of the system

Services that provides convenience to user/programmer EPICFU:

- User interface
    - Batch interface, CLI, GUI
- Program execution
    - Able to load, run and terminate a program
- Input/ Output Operations
    - File, CD/DVD drive, Display device, Smart Card
- File-system manipulation
    - Create/Read/Write/Delete files and directories
    - Permission management
- Communications
    - Local/remote inter-process communication
- Error detection
    - Able to detect error and take proper action to ensure consistent computing

Services that ensures efficient operation of the system (RAP):

- Resource allocation
    - Resources to be allocated among multiple users fairly
    - Different algorithms for different resources
- Accounting
    - Keep track of usage statistics. e.g. CPU, Printer, hard disk quota
    - Reconfigure system to improve computing services
- Protection and security
    - Security of system from outsiders
    - Ensure access to all system resources is controlled
    - Audit trail of access

Functions of an Operating System

- 4 Major group of basic functions common to all OS
    - Device management
    - Process, thread and resource management
    - Memory management
    - File management
- Close interactions between the 4 managers.

Device management

- Refers to the way generic devices are handled. Includes disk, tapes, terminals, printers, etc
- Special management approaches for processor and memory
- Partitioning design simplifies adding and upgrading of devices

Process, thread and resource management

- Creates abstractions of processes, threads, resources
- Allocates processor resource equitably (in a fair manner)
- Allocates and tracks abstract resource such as queues, semaphores (access control), messages
- Cooperates with memory manager to administer the primary memory

Memory Management

- Administer and allocate primary memory
- Enforces resource isolation
- Enables sharing between processes
- Provides virtual memory extensions
    - Abstract machine's memory appear larger than physical memory

File Management

- Creates abstraction of storage devices. i.e. I/O operations
- Range from byte stream files to indexed records
- Local and Remote file systems

OS Requirements

- Manage resource sharing
    - Time/space-multiplexing where appropriate.
- Exclusive use of a resource
    - Allow processes to use a resource exclusively as required.
- Isolation
    - Allow a resource to save information without fear of it being modified or tampered with.
- Managed sharing
    - Sharing must be done in an orderly fashion according to the properties of the resource.
    - E.g. printer vs disk drive

Implementation mechanisms

- 3 basic mechanisms to address isolation and sharing:
    - Processor modes (User vs Supervisor)
    - Kernel
    - Method of invoking system service

Processor Modes

- Distinguish between trusted and un-trusted software
- Determine exception capability and accessible memory areas
- Modern processors provides 2 modes.
- Mode bit: Supervisor mode or User mode

- Trusted OS software executes in supervisor mode.
- All other software (including some parts of the OS) executes in user mode.
- Concept allow for OS to be able to control access to resources.
    - User programs have to ask the OS to execute privileged instructions on their behalf.
    - Any particular configuration can isolate or permit sharing or resources according to the administrator's policy

Supervisor mode (for OS)

- Can execute all machine instructions, including privileged instructions.
- e.g. of privileged instructions:
    - I/O instructions
    - Memory-related instructions
    - Processors mode-change instructions
- Can reference all memory locations:
    - System (or supervisor, kernel or protected) space
        - Refers to memory area used by the OS.
    - User space
        - Refers to memory area used by application processes.

User mode (for user programs)

- Can only execute a subset of instructions.
- Can only reference a subset of memory locations.

Kernels

- The part of the OS critical to correct operation (trusted software).
- Implements the basic mechanisms that assure secure operation of entire OS.
- Executes in supervisor mode.
- The `trap` instruction is used to switch from user to supervisor mode, entering the OS.

Requesting Service from OS

- In order to execute privileged instructions, user programs have to activate routines in the kernel, which can then execute on the user programs' behalf.
- Two techniques:
    - System call
    - Message passing

System Call

A system call is a call to the OS through the Supervisor mode to do service for the application program.

- In system call, the relevant function is activated via a `trap` instruction.
- OS provides a sub function which the user program calls.
- Stub function will switch the processor to supervisor mode.
- It will execute the `trap` instruction by branching to a trap table to the entry point of the system function to be invoked.
- On completion, processor is switched back to user mode and control returns to user process.

- Appears as ordinary function call to the application programmer.

Message Passing

- In the message passing method, the user program constructs a message that requests the desired service.
- Uses OS `send()` system call.
- OS kernel implements target function.
- Kernel process must be started or active i.e. must be in supervisor mode. to receive message.
- User process waits for result with `receive()` operation.
- Kernel sends message back to user process on completion.

System Call vs Message Passing

- In the system call, the user process/threads gains ability to execute privileged instructions.
- In the message passing, the system function is executed by the kernel process/thread.
- System calls are more efficient than message passing.
    - Message passing has the cost of message formatting/copying and process multiplexing.
    - System calls just requires a `trap` command.
- Most modern systems use system calls.

Modern OS architecture

- A modern OS architecture implements each manager in its own software module.
- Interaction among various managers via abstract data type.
- Frequent calls incurs performances penalty.
- Sacrifice modularity for performance
    - Monolithic kernel implementation
        - Four basic modules are combined into a single software module.
    - Microkernel approach
        - Employs a small kernel that implements only the essential and critical functions.
        - Reminder function are implemented outside the kernel, possibly in separated modules.