Lecture 1 - Operating System Overview

Instructor : Bibhas Ghoshal (bibhas.ghoshal@iiita.ac.in)

Autumn Semester, 2018

Lecture Outline

- Defining an Operating System
- Computer System Structure
- Illustration of the role of Operating System
- Goals of Operating System
- Tasks performed by OS
- Types of Operating system
- OS isolation - user mode and kernel mode
- Operating System structure

Some slides have been taken from:

- lecture slides of the book - Operating System Concepts by Silberschatz, Galvin and Gagne, 2005
- lecture slides of CS140 – (Instructors: Adam Belay et al.), Stanford University

What is an Operating system?

A layer between hardware and software that allows the user of a computer system to use the system in a fast and efficient manner.

Abstract views of OS by four different users

Student - That allows to browse the internet on a smart phone

Salesman - allows me to use the sales package on the computer

Programmer - allows me to develop programs efficiently

Technician - component of the computer system that controls the plant

All views are correct, but none complete. All consider utilizing the computer system for specific purpose

What is a Computer System that OS manages?

- Computer system can be divided into :
    - Hardware – provides basic computing resources
        - CPU - runs instructions corresponding to user code or OS
        - Memory - holds the user code and data and OS
        - I/O devices- Secondary storage devices, network card, keyboard, mouse, printer
    - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
        - Word processors, compilers, web browsers, database systems, video games

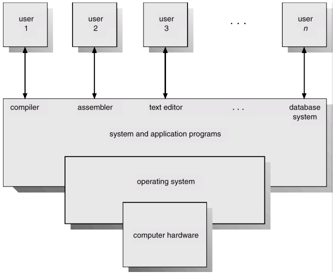
Operating system controls and coordinates use of hardware among various applications and users

- User view : Concerned with ease of use
- System view : Resource utilization and to control user programs and I/O

Goals of an Operating System

- Provides abstraction for applications
  - Manages and hides details of hardware from application
  - Accesses hardware through low level applications
- Resource Management - controls and coordinates use of hardware among various applications
- Provides protection - Ensures that multiple applications do not clobber each other
  - Prevents one user from clobbering another

OS provides Hardware Abstraction



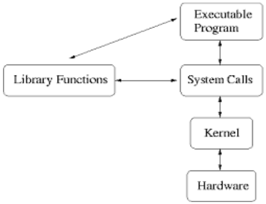
- No more details required for the programmers
- Applications can reuse OS functionality
- OS interfaces are consistent - application remains same when hardware changes

How does OS provide Hardware Abstraction?

The common utilities to access hardware are provided as services by the OS and user programs access these services through interfaces.

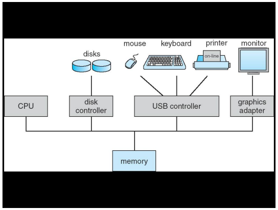
- Kernel : OS core that holds the important functionalities (services required by the user programs to access hardware)
- System Programs : utilities to access the services provided by kernel

User programs request for kernel services through interfaces called **System calls**



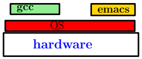
Resource Management of Operating System

- Hardware cost forces use of limited resources
    - Need to share resources - One or more CPUs, device controllers connect through common bus providing access to shared memory
    - Concurrent execution of CPUs and devices competing for memory cycles

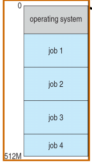


- Challenge is to manage resources for competing users
- OS is a resource allocator : Manages all resources (CPU, memory, disks etc.) and decides between conflicting requests for efficient and fair resource use

Multiprogramming - needed for efficiency



- Idea: Run more than one process at once; When one process blocks (waiting for disk, network, user input, etc.) run another process
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (e.g, for I/O), OS switches to another job



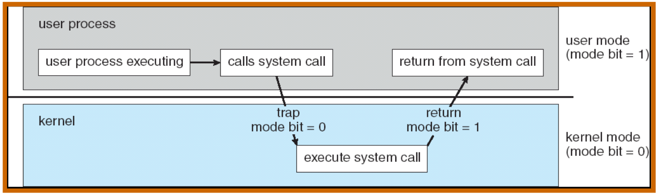
Memory Layout for Multiprogramming

OS Isolation and Protection

- OS should protect the users/processes from each other as well as protect itself from users
- Dual-mode operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
    - Set interrupt after specific period
    - Operating system decrements counter
    - When counter zero generate an interrupt
    - Set up before scheduling process to regain control or terminate program that exceeds allotted time



OS usage

Algorithm 1 Pseudocode to show the use of OS

```
1: int $main(intargc, char * argv[], char * envp[])$ {
2: char$buf[100]$;
3: char $ptr = malloc(100)$;
4: $printf(''Helloworld'')$;
5: int $fd = open(''/dir1/file'', OWRONLY, 0666)$;
6: $(ptr + 1000) =' 0'$;
7: $write(fd, ptr, 100)$;
8: $close(fd)$;
9: $exit(0)$; }
```

- Compile the program ($gcc file.c) - OS makes sure the file (residing in HDD) is provided and takes care of the H/W on which it runs
- array call (line 2) - OS allocates a space for array in the memory according to the required format and it ensures that when your friend runs the same program, it does not overwrite contents of your program.
- dynamic memory allocation (line 3) - provides more memory than alloted.
- library call (e.g printf in line 4)- OS makes sure to write the result on the screen when the library call is made.
- system call (e.g. open in line 5) - the OS interprets the string as a file/directory residing in hard disk and transfers it from hard disk to main memory

- segmentation fault (line 6) - you write 1000 bytes to the alloted 100 bytes. OS informs that some condition of the code has been violated.
  Two ways you mess up the code
  - you harm yourself - OS informs you
  - you harm others - OS ensures you do not do that; prevents writing to memory code alloted to others (open file in read only mode)
- write call (line 7) - OS system call transfering something from memory to disk
- exit code (line 10) - return to some other program (say console)

What does the OS do?

- Process Management
- Memory Management
- Storage and I/O Management

Process Management

- A process is a program in execution. It is a unit of work within the system.
  Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task- CPU, memory, I/O, files and Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
- Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Concurrency by multiplexing the CPUs among the processes / threads

Process Management

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
- Optimizing CPU utilization and computer response to users
- Memory management activities
- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- Protection – any mechanism for controlling access of processes or users to resources defined by the OS
- Security – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do whatever
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file
  - Privilege escalation allows user to change to effective ID with more rights

Examples of Operating Systems

- Desktop and server OS: Microsoft Windows, UNIX and variants - Linux, Solaris, FreeBSD, Mac OSX
  - Resource utilization is important
- Realtime OS: QNX, Vxworks, RT linux
  - Meeting timing constraints is a major concern
- Embedded OS : Contiki OS
  - used in memory constrained environment
- Mobile OS : Android, iOS, Windows Touch