Operating Systems

Gurmukh Singh

B.Tech. CSE

Instructor:
Mr. Amit Chauhan

Contents

1 Operating systems

Def^n :
Operating system is an interface between user and the hardware. Without operating system the user cannot access any hardware.

Def^n :
When we have many programs running at one given time, we need to allocate CPU to one of them and memory with osem other application. This is known as resource sharing. Sometimes resources are not sharable so we allocate some process or program and later after taking it back we can allocate it to some other process.

1. Operating system acts like a manager. It is a kind of book keeping like which resource is currently allocated to which process.
2. memory can be managed like :
   - how much memory do we have?
   - how much memory we can allocate?
   - how much free memocy is available?

1.1 Goals of operating systems:

1. Primary Goal : Convenience (personal computers)
2. Secondary Goal: Efficiency (Supercomputers)

1.2 Types of operating systems:

1. Batch OS
2. Multiprogramming OS
3. Multitasking OS
4. Multiprocessing OS
5. Real time OS

1.2.1 Batch OS

There used to be a single computer to which everyone was given access. This computer is called mainframe and everyone should give a program to this mainframe computer in a queue. Then the comuter will pick them one after the other and execute them and later at some point the user will pick the program.

If any of the jobs require less amount of CPU time and more I/O time since here CPU time is greater than I/O time still CPU is not allocated to any other job until the current process is completed. It is not interactive as well which means thatresponse is not given immediately

1.2.2 Multiprogramming OS

It is basically an extension to Batch processing using multiprogramming CPU will be used efficiently. e.g. suppose the first job requires I/O then it will go to I/O device then CPU will be allocated to second job at the same time in such a way that CPU remains busy as longer as the are jobs.

Advantages of Multiprogramming OS:

1. We need not to keep CPU ideal.
2. CPU will be busy all the time with maximum utilization

1.2.3 Multitasking OS

It is an extension to multiprogramming OS. CPU will be multiplexing among all the jobs without completing any first job. Like for some time CPU is allocated to job1 then to job2 then to job3 and so on. And repeat this cycle again and again.

e.g. round robin approach.

1.2.4 Multiprocessing OS

In this instead of having one CPU we will be having lots of CPU's in the same computer.

Advantages:

1. Many jobs can be run simultaneously and parallelism can be achieved.
2. Throughput can be improved. (number of jobs completed per unit time.)
3. if one CPU fails, jobs can be rescheduled between other CPU's

1.2.5 Real time OS

In this scenario, we are given some jobs and jobs will be having some deadlines. then we are supposed to finish those jobs in the assigned deadline.
In this scenario timing is everything.

2 Process management

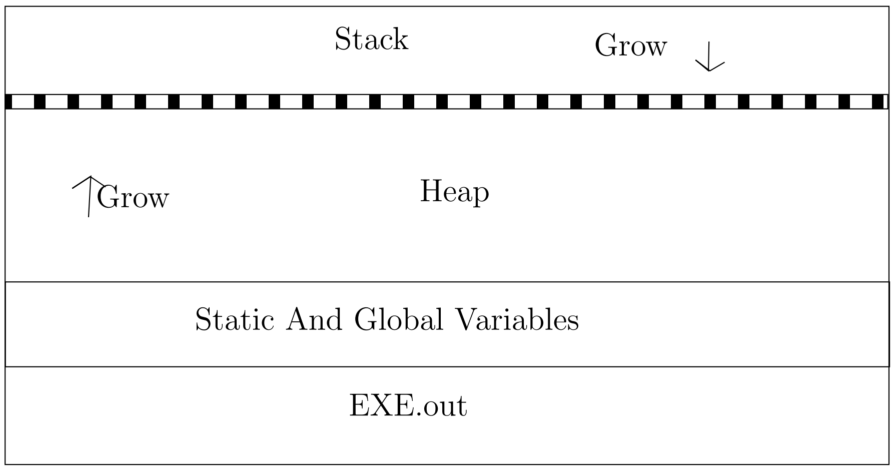
suppose we write a file "exa.c" in C language. now the program has to be first converted from high level language to low level language or machine language. This process is done by a compiler.

Program generally resides in the secondary memory. Operating system will take it from secondary memory and place it in main memory. Then start the execution. When the OS puts the program in main memory, it creates a datastructure called a process.

2.1 Process

It is something which is created by the operating sysetem in order to execute a program. So before we execute a program we need to create a process. The proces is real and the program is virtual.

For example: Making of the food is the process and the recipie is the program


Stack
Grow
Grow
Heap
Static And Global Variables
EXE.out

Once the Datastructure is created, the OS will start executing the executable code line by line.

2.1.1 Attributes of a process

1. Process ID
2. Program counter
3. Process state
4. Priority
5. General purpose register
6. List of open files
7. List of open devices
8. Protection

Process ID:
Each process is given a unique number called the process ID. the number of bits depends on the OS.

Program counter:
While we are executing a process, we suddenly start a process and then we restart it at the end of execution.

Process state:
This is the current state of the process.

Priority:
Whenever a process is created by an OS, it is assigned some priority. So a process with high priority will be executed first by the CPU.

GPR:
When a process is prempted and some other process comes for execution then the contents of the registers related to the prempted process should also be preserved so that when it resumes it's execution consistency will be maintained.

List of open files:
During the execution of a process some files need to be opened , some of the files are opened for reading purpose and others for writing purpose.

List of oped devices:
Devices which are currently used by the process

Protection:
One process should not go into other process workspace and vice versa. For every process OS is going to create a process control block (PCB) which stores all the above information about a process.

2.2 States of a process

A process from it's creation to completion. Depending upon the requirements a process will transition from one state to another. When we have a lot of processes that are ready to rum is known as multiprogramming.

States of a process

1. New: Whenever a process is created It is in the new state.
2. Ready: whenever processes are created they are ready to run in the main memory. There are various processes which are ready to run In multiprogramming there are two types of processes:
   (a) With premption
   (AKA multitasking or time sharing) With premption the process is forcefully stopped
   (b) Without premption
   Without premption the process is allowed to complete it's execution, it is not forced to stop.
3. Run
   simultaneously only process will be running if we have one CPU and if we have n CPU's then n processes will be running simultaneously.
4. Block/wait
   Sometimes a process which is running might need some resources like a file or access to a device then we pull it out and put it in block state or wait state until it finishes with I/O. Once it finishes the I/O we again bring it to previous state i.e. ready state.
5. Termination/completion
   Once a process finishes or completes it's execution it is known to be in completion stage or termination stage. Once a process completes it's execution it's PCB(Process control block) is also deleted.
6. Suspend ready
   Whenever space in main memory is not sufficient and whenever we want to accomodate more important processes(high priority processes) we just move the low priority processes out of the main memory to the secondary memory. These are the set of processes which work initially in the ready state and now because of lack of resources we are throwing them outside into the secondary memory.
7. Suspend wait/ suspend block
   performing processes are performing some I/O and are waiting for initialization so we let them wait in secondary memory or move to wait or block state of secondary memory.

Generation of OS:

1. First generation: (1940-1948)
   large and expensive.
2. Second Generation: (1956-1963)
   transistors replaced vaccuum tubes
   generated a lot of heat.
3. Third Generation(1964-1971)
   Introduced IC.
   transistors were placed on silicon chips
   keyboards and monitors became widespread.
4. Fourth Generation(1971-present)
   Microprocessors, GUI's, mouse and handheld device.
5. Fifth Generation(Present and beyond)
   Based on AI, Early stage
   voice recognition
   parallel processing and superconductors are aiding this transition.
   quantum computers and nanotechnology.

Services of OS:

1. provide programs an environment to execute.
2. provide users means to run programs.

Program execution

1. leads a program into memory.
2. execution the program.
3. handles program's execuion.
4. Provides a mechanism for process syncronization.
5. Provides a mechanism for process communication.
6. Provides a mechanism for deadlock handling.

I/O