GPCET
Pioneering Innovative Education

G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC with 'A' Grade of UGC, Approved by AICTE, New Delhi

Permanently Affiliated to JNTUA, Ananthapuramu

(Recognized by UGC under 2(f) and 12(B) & ISO 9001:2008 Certified Institution)

Nandikotkur Road, Venkayapalli, Kurnool – 518452

Department of Computer Science and Engineering

Bridge Course
On
Operating Systems

By

P.Suman Prakash.

GPCET
Pioneering Innovative Education

Table of Contents

Sno	Topic	Page number
1	• Operating system definition • Functions of OS • Interrupt • Storage structure	1
2	PROCESS SYNCHRONIZATION	3
3	CRITICAL SECTION PROBLEM	4
4	SEMAPHORES:	5
5	DEADLOCK:	6
6	STORAGE MANAGEMENT:	9
7	FILE SYSTEMS AND ORGANIZATION:	7
8	UNIX:	9
9	Security	10

Aims and Objectives:

Computer software can be divided into two main categories: application software and system software. Application software consists of the programs for performing tasks particular to the machine's utilization. Examples of application software include spreadsheets, database systems, desktop publishing systems, program development software, and games."

The most important type of system software is the operating system. An operating system has three main responsibilities:

i) Perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

ii) Ensure that different programs and users running at the same time do not interfere with each other.

iii)Provide a software platform on top of which other programs (i.e., application software) can run.

The basic functions of an operating system are as follows:

- File management--analogous to the file cabinet and its use
- Working with the Files--analogous to picking up and preparing to use a calculator or some other tool
- Configuration of your working environment--analogous to shifting your desk around to suit you better.

An OS is a program that acts as an intermediary between a user of a computer and the computer hardware

Goals: Execute user programs, make the comp. system easy to use, utilize hardware efficiently

Computer system: Hardware ↔ OS ↔ Applications ↔ Users (↔ = 'uses')

OS is:

Resource allocator: decides between conflicting requests for efficient and fair resource use

Control program: controls execution of programs to prevent errors and improper use of computer

Kernel: the one program running at all times on the computer

• **Bootstrap program:** loaded at power-up or reboot Stored in ROM or EPROM (known as firmware), Initializes all aspects of system, loads OS kernel and starts execution

• I/O and CPU can execute concurrently

• Device controllers inform CPU that it is finished w/ operation by causing an interrupt

◦ Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines ,Incoming interrupts are disabled

while another interrupt is being processed.Trap is a software generated interrupt caused by error or user request

OS determines which type of interrupt has occurred by polling or the vectored interrupt system

• **System call:** request to the operating system to allow user to wait for I/O completion

• **Device-status table:** contains entry for each I/O device indicating its type, address, and state

◦ OS indexes into the I/O device table to determine device status and to modify the table entry to include

INTERRUPT • STORAGE STRUCTURE:

Main memory – random access, volatile ◦ Secondary storage – extension of main memory That provides large non-volatile storage

Disk – divided into tracks which are subdivided into sectors. Disk controller determines logical interaction between the device and the computer.

Caching – copying information into faster storage system

Multiprocessor Systems: Increased throughput, economy of scale, increased reliability, can be asymmetric or symmetric,c lustered systems – Linked multiprocessor systems

Multiprogramming – Provides efficiency via job scheduling when OS has to wait (ex: for I/O), switches to another job

Timesharing – CPU switches jobs so frequently that each user can interact with each job while it is running (interactive computing)

• Dual-mode operation allows OS to protect itself and other system components – User mode and kernel mode ,some instructions are only executable in kernel mode, these are privileged • Single-threaded processes have one program counter, multi-threaded processes have one PC per thread

• **Protection** – mechanism for controlling access of processes or users to resources defined by the OS

Security – defense of a system against attacks , User IDs (UID), one per user, and Group IDs, determine which users and groups of users have which privileges

Process:

Process is defined as a program in execution and is the unit of work in a modern timesharing system. Such a system consists of a collection of processes: Operating-system processes executing system code and user processes executing user code. All these processes can potentially execute concurrently, with the CPU (or CPUs) multiplexed among them. By switching the CPU between processes, the operating system can make the computer more productive.

A process is more than the program code, it includes the program counter, the process stack, and the contents of process register etc. The purpose of process stack is to store temporary data, such as subroutine parameters, return address and temporary variables. All these information

will be stored in Process Control Block (PCB). The Process control block is a record containing many pieces of information associated with a process including process state, program counter, cpu registers, memory management information, accounting information, I/O status information, cpu scheduling information, memory limits, and list of open files.

PROCESS SYNCHRONIZATION:

Introduction to process synchronization

Process synchronization will be clear with the following example. Consider the code for consumer and producer as follows

Producer

```
while(1)
{
while(counter = = buffersize);
buffer[in]=nextproduced;
in = (in+1) % buffersize;
counter ++;
}
```

Consumer

```
while(1)
{
while(counter = = 0);
nextconsumed = buffer[out];
out = (out+1) % buffersize;
counter --;
}
```

Both the codes are correct separately but will not function correctly when executed concurrently. This is because the counter++ may be executed in machine language as three separate statements as

```
register1 = counter
register1 = register1 + 1
counter = register1
```

and the counter- - as

```
register2 = counter
register2 = register2 - 1
counter = register2
```

The execution of these statements for the two processes may lead to the following condition for

example.

a) producer execute register1 = counter (register1 =5)

b) producer execute register1 = register1 + 1 (register1 =6)

c) consumer execute register2= counter (register2 =5)

d) consumer execute register2 = register2 - 1 (register2 =4)

e) producer execute counter = register1 (counter = 6)

f) consumer execute counter = register2 (counter = 4)

You can see that the answer counter =4 is wrong as there are 5 full buffers.

A situation like this, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access take place, is called a race condition. For this we have to make sure that only one process at a time should manipulate the counter. Such situation occurs frequently in OS and we require some form of synchronization of processes.

CRITICAL SECTION PROBLEM

Each process will be having a segment of code called a critical section, in which the process may be changing a common variable, updating a table, writing a file, and so on.

```
do
{
   entry section
    critical section
    exit section
     reminder section
}  while(1);
```

A solution should satisfy the following three requirements

a) Mutual exclusion: If a process is executing in its critical section, then no other processes can be executing in their critical sections

b) Progress: If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

c) Bounded waiting: There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

SEMAPHORES:

The solution described in the above section cannot be generalized most of the times. To over come this, we have a synchronization tool called a semaphore proposed by Dijkstra. Semaphores are a pair composed of an integer variable that apart from initialization is accessed only through two standard atomic operations: wait and signal.

***wait*:** decrease the counter by one; if it gets negative, block the process and enter its id in the queue.
signal: increase the semaphore by one; if it's still negative, unblock the first process of the queue, removing its id from the queue itself.

```
wait(S)
  {
            while(S<=0);
              S - -;
   }

signal(S)
             {
                 S++;

               }
```

DEADLOCK:

state of deadlock (or is deadlocked) if the process or thread is waiting for a particular event that will not occur. In a system deadlock, one or more processes are deadlocked. Most deadlocks develop because of the normal contention for dedicated resources (i.e., resources that may be used by only one user at a time). Circular wait is characteristic of deadlocked systems.

One example of a system that is prone to deadlock is a spooling system. A common solution is to restrain the input spoolers so that, when the spooling files begin to reach some saturation threshold, they do not read in more print jobs. Today's systems allow printing to begin before the job is completed so that a full, or nearly full, spooling file can be emptied or partially cleared even while a job is still executing. This concept has been applied to streaming audio and video clips, where the audio and video begin to play before the clips are fully downloaded.
In any system that keeps processes waiting while it makes resource-allocation and process scheduling decisions, it is possible to delay indefinitely the scheduling of a process while other processes receive the system's attention. This situation, variously called indefinite postponement may occur because of biases in a system's resource scheduling policies. Some systems prevent

indefinite postponement by increasing a process's priority as it waits for a resource—this technique is called aging.

Resources can be preemptable (e.g., processors and main memory), meaning that they can be removed from a process without loss of work, or nonpreemptible meaning that they (e.g., tape drives and optical scanners), cannot be removed from the processes to which they are assigned. Data and programs certainly are resources that the operating system must control and allocate. Code that cannot be changed while in use is said to be reentrant. Code that may be changed but is reinitialized each time it is used is said to be serially reusable.

Reentrant code may be shared by several processes simultaneously, whereas serially reusable code may be used by only one process at a time. When we call particular resources shared, we must be careful to state whether they may be used by several processes simultaneously or by only one of several processes at a time. The latter kind—serially reusable resources—are the ones that tend to become involved in deadlocks.

CHARACTERISTICS OF DEADLOCK

The four necessary conditions for deadlock are:

a) A resource may be acquired exclusively by only one process at a time (mutual exclusion condition);

b) A process that has acquired an exclusive resource may hold it while waiting to obtain other resources (wait-for condition, also called the hold-and-wait condition);

c) Once a process has obtained a resource, the system cannot remove the resource from the process's control until the process has finished using the resource (nopreemption condition);

d) And two or more processes are locked in a "circular chain" in which each process in the chain is waiting for one or more resources that the next process in the chain is holding (circular-wait condition).

STORAGE MANAGEMENT:

The organization and management of the main memory or primary memory or real memory of a computer system has been one of the most important factors influencing operating systems design. Regardless of what storage organization scheme we adopt for a particular system, we must decide what strategies to use to obtain optimal performance.

Storage Management Strategies are of four types as described below

a) Fetch strategies – concerned with when to obtain the next piece of program or data for transfer to main storage from secondary storage

a. Demand fetch – in which the next piece of program or data is brought into the main storage when it is referenced by a running program Anticipatory fetch strategies – where we make guesses about the future program control which will yield improved system performance

b) Placement strategies – concerned with determining where in main storage to place and incoming program. Examples are first fit, best fit and worst fit

c) Replacement strategies – concerned with determining which piece of program or data to replace to make place for incoming programs

FILE SYSTEMS AND ORGANIZATION:

In computing, a file system (often also written as file system) is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. File systems may use a data storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files, they might provide access to data on a file server by acting as clients for a network protocol (e.g., NFS, SMB, or 9P clients), or they may be virtual and exist only as an access method for virtual data.

More formally, a file system is a set of abstract data types that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data. File systems share much in common with database technology, but it is debatable whether a file system can be classified as a special-purpose database (DBMS).

UNIX:

The Unix operating system was created more than 30 years ago by a group of researchers at AT&T's Bell Laboratories. During the three decades of constant development that have followed, Unix has found a home in many places, from the ubiquitous mainframe to home computers to the smallest of embedded devices. This lesson provides a brief overview of the history of Unix, discusses some of the differences among the many Unix systems in use today, and covers the fundamental concepts of the basic Unix operating system.

UNIX is a computer operating system, a control program that works with users to run programs, manage resources, and communicate with other computer systems. Several people can use a UNIX computer at the same time; hence UNIX is called a *multiuser* system. Any of these users can also run multiple programs at the same time; hence UNIX is called *multitasking*. Because UNIX is such a pastiche—a patchwork of development—it's a lot more than just an operating system. UNIX has more than 250 individual commands. These range from simple commands—for copying a file, for example—to the quite complex: those used in high-speed networking, file revision management, and software development. Most notably, UNIX is a multichoice system. As an example, UNIX has three different primary command-line-based user interfaces (in UNIX, the command-line user

interface is called a *shell* The three choices are the Bourne shell, C shell, and Korn shell. Often, soon after you learn to accomplish a task with a particular command, you discover there's a second or third way to do that task. This is simultaneously the greatest strength of UNIX and a source of frustration for both new and current users.

Security

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. We're going to discuss following topics in this chapter.

- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

System Threats

System threats refer to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats create such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.

- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.