What is an operating system? Explain the function of operating system, Types of OS, Features and Examples

An **Operating System (OS) is a software that acts as a**n interface between computer hardware components and the user. Every computer **system must have at** least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks .

The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible for the user to use any computer or mobile device without having an operating system.

HARDWARE

•CPU, Memory, Hard Drive

OPERATING SYSTEM

•Windows, Apple OS X Linux

END USER

a

History Of OS

• Operating systems were first developed in the late 1950s to manage tape storage

The General Motors Research Lab implemented the first OS in the early 1950s for their IBM 701 . In the mid-1960s, operating sy**stems start**ed to use disks . In the late 1960s, the first version of the Unix OS was developed . The first OS built by Microsoft was DOS. It was built in 1981 by purchasing the 86-DOS software

from a Seattle company The present-day popular OS Windows first came to existence in

1985 when a GUI was created

and paired with MS-DOS. **Types of Operating System (OS)** Following are the popular types of Operating System:

Batch Operating System Multitasking/Time Sharing OS Multiprocessing OS Real Time OS Distributed OS

Network OS . Mobile OS **Batch Operating System** Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group. The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer **operator.**

Multi-Tasking/Time-sharing Operating systems

Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time. The processor time (CPU) which is shared among multiple users is termed as time sharing. **Real time OS** A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems are the real time OS example. **Distributed Operating System** **Distributed systems use many processors located in different machines to provide very fast computation** to its users. **Network Operating System** Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions. **Mobile OS** *M*obile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices. Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watchOS. **Functions of Operating System** Below are the main functions of Operating System:

Memory *M*anagement

Processor **Management**

File **Management**

Device Management

i/o **management**

Secondary

Storage management

Security

Command Interpretation

Networking

Communication Management

Job accounting

Functions of Operating System In an operating system software performs each of the function:

1. **Process management:- Process management helps OS to create an**d delete processes. It also

provides mechanisms for synchronization and communication among processes. 2. **Memory management:-** Memory management module performs the task of allocation and de allocation of memory space to programs in need of this resources. 3. **File management:-** It manages all the file-related activities such as organization storage,

retrieval, naming, sharing, and protection of files.

4. **Device Management**: Device management keeps tracks of all devices. User This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices. **5.*1/O* System Management: One** of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

6. **Secondary-Storage Management:** Systems have several levels of Application storage which includes primary storage, secondary storage, and cache **storage. Instructions and data must be stored in primary storage or cache** so that a running program can reference it.

7. Security:- Security module protects the data and information of a Operating System

computer system against malware threat and authorized access. **8. Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

9. Networking: A distributed system is a group of processors which do not Hardware share memory, hardware devices, or a clock. The processors communicate

with one another through the network. **10. Job accounting**: Keeping track of time & resource used by various job and users. **11. Communication management: C**oordination and assignment of compilers, interpreters, and

another software resource of the various users of the computer systems. **Features of Operating System (OS)** Here is a list important features of OS:

• Protected and supervisor mode

Allows disk access and file systems Device drivers Networking Security

Program Execution

- **Memory m**anagement Virtual Memory Multitasking

Handling I/O operations Manipulation of the file system Error Detection and handling Resource allocation Information and Resource Protection

information and Resource Protection

D

a

•

Advantage of using Operating System

. Allows you to hide details of hardware by creating an abstraction

Easy to use with a GUI

Offers an environment in which a user may execute programs/applications

• The operating system must make sure that the computer system convenient to use

• Operating System acts as an intermediary among applications and the h**ardware components**

- It provides the computer system resources with easy to use format
- **Acts as an intermediator between all hardware's and softwa**re's of the system

Disadvantages of using Operating System

. If any issue occurs in OS, you may lose all the contents which have been stored in your system

• Operating system's software is quite expensive for small size organization which adds burden on

them. Example Windows It is never entirely secure as a threat can occur at any time

What is a Kernel? The kernel is the central component of a computer operating systems. The only job performed by the kernel is to the manage the communication between the software and the hardware. A kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.

Kernel

Hardware

Operating System

Terminal

User

Shell

Introduction to Kernel **Features of Kennel**

Low-level scheduling of processes

I

A

C

Inter-process communication . Process synchronization

. Context switching **Types of Kernels** There are many types of kernels that exists, but among them, the two most popular kernels are: **1. Monolithic** A monolithic kernel is a single code or block of the program. It provides all the required services offered by the operating system. It is a simplistic design which creates a distinct communication **layer between the hardware and software. 2. Microkernels** Microkernel manages all system resources. In this type of kernel, services are implemented in different address space. The user services are stored in user address space, and kernel services are stored under kernel address space. So, it helps to reduce the size of both the kernel and operating system.

Difference between Firmware and Operating System

Firmware

Operating System

Firmware is one kind of programming that is embedded on a chip in the device which controls that specific device.
OS provides functionality over and above that which is provided by the firmware.

Firmware is programs that been encoded by the manufacture of the IC or something and cannot be changed.
OS is a program that can be installed by the user and can be changed.

It is stored on non-volatile memory.
OS is stored on the hard drive.

Difference between 32-Bit vs. 64 Bit Operating System

Parameters
32. Bit
64. Bit

Allow 32 bit of data processing simultaneously
Architecture and **Software**
Allow 64 bit of data processing simultaneously

Compatibility
32-bit applications require 32-bit OS and CPUs.
64-bit applications require a 64-bit OS and CPU.

Systems Available
All versions of Windows 8, Windows 7, Windows Vista, and Windows XP, Linux, etc.
Windows XP Professional, Vista, 7, Mac OS X and Linux.

Memory Limits
32-bit systems are limited to 3.2 GB of RAM.
64-bit systems allow a maximum 17
Billion GB of RAM.

Define the term semaphore. How does semaphore help in dining philosopher problem.

A system has two process and three resources. Each process needs a maximum of two resources, IS deadlock possible? Explain with answer.

No, deadlock is not possible. **Because in the worst case each process gets one resource and we still have one more reso**urce left, which can be assigned to one of the processes. After some time that process will release the resources and can be assigned to another process.

Suppose a new process in a system arrives at an average of six process per minute and each such process requires an average of 8 seconds of service time, Estimate the fraction of time the CU is busy in a system with a single processor,

Given that there are on an average 6 processes per minute.

So the arrival rate = 6 process/min. i.e. every 10 seconds a new process arrives on **an average. Or we can say that every process stays f**or 10 seconds with the CPU Service time = 8 sec. Hence the fraction of time CPU is busy = service time / staying time = 8/10

=0.8 So the CPU is busy for 80% of the time.

Given Refferences to the following pages by a program,

0,9,0,1,8,1,8,7,8,8,1,2,3,2,7,8,2,3,8,3

How many page faults will occur if the program has three page frames for each of the following algorithms?

A) FIFO B) LRU

GOOD MORNING

long

I given References to the following pages by a progoam: 09.0,

118, 1, 8, 7, 8, 8, 272, 8, 2, 7, 8,
2, 3, 813 How many page faults will occur if the program has 3 pages foume for each of the following a) sifo Jalgorithms ger forumet

Б) LED

fifol * hit hit hit hit hit hit & hit hit hit hit hit & 7100018 8 8 8 8 8 8
18 18 18 818
1999 1998 7777 f 1 1 1 1 1 1 1 1 4 2/2/2/2/21
& s hit a t hit hit & hit hit hit hit hit & hit hit of hit fobo 10 000
774 2 2 2 2 2 2 2 2 1199 9 8 8 8 8 8 8 8 8 8 8 8 818
2171313

- page fault : -8 A page hit 8-12

What is file? Explain how access control matrix provides resource protection that may access process,

What do you mean by one-time password in authentication? How worms are differing from virus.

Assignment 4

1. What is race condition?

A race condition is an undesirable situation that occurs when a device or **system attempts or perform two** or more operation at the same time but because of the nature of the device or system the operation must be done in the proper sequence to de done correctly.

Example

A Simple example of a race condition is a light switch In some homes there are multiple light switches connected to a common celling light when these types of circuits are used, the switch position becomes **irrelev**ant, If the light is on moving either switch for its current position turns the light off similarly, it the light is off then moving ither switch from its current position turns the light on with that in mind imagine what might happen if two people tired to turn on the light using two different switches at exactly the some time one instruction might cancel the other or the two actions might trip the circuit **breakers.**

2.When a computer is being developed, it is usually first simulated by a program that runs one instruction at a time. Even multiprocessors are simulated strictly sequentially like this. Is it possible for a race condition to occur when there are no simultaneous events like this?

Yes. The simulated multi programmed computer could have race condition.

For example, while process A is running, it reads out some shared variable. Then a simulated clock tick happens and process B runs. It also reads out the same variable. Then it adds 1 to the variable. When process A runs, if it also adds 1 to the variable, we have a race condition.

3.Define the term Semaphore. How does Semaphore help in Dining Philosopher problem.

The Dining Philosophers Problem says that there are five philosophers sitting around a circular table and they eat and think alternatively. There is a bowl of noodles for each of them and there are five chopsticks **placed between e**ach philosopher. However, a philosopher can only eat when they can acquire both left and right chopsticks. Otherwise, the philosopher keeps the chopstick down and start thinking. Each chopstick can be held by only one philosopher at a time. After one philosopher finishes eating he needs to put down both chopsticks and those chopsticks become available for others.

P5

P2

P4

4.Compare the use of monitor and semaphore operations

Monitor

A variable used to control access to a common resource by multiple process in a concurrent system such as a multi

tasking operating system.

• An Integer Variable

Semaphore

A synchronization construct that allows threads to have both mutual exclusion and the ability to wait (block) for certain condition to become true **An Abstract data types**

There is no condition variable concept *W*hen a process requires to access the semaphore, It performs wait () and Performs Signal () when releasing the **resource**

Has condition variables

A Process uses procedures to access the **shared v**ariable in the monitor

5.What is the meaning of busy waiting? what others kinds of waiting are in OS? Compare each types on their applicability and relative merits.

• **The Repeated execution of a loo of code while for an event to o**ccur is called busy-waiting.

- The other kind of waiting in OS:
  - Example: 1/0 semaphore & it does so without having the cpu assign to it. The alternative to busy waiting is blocked waiting also known as sleeping waiting, **Where a task sleep**s until an event occurs. For blocked waiting to work the must be some central **agent that can wake up the task**

• When the event has car may have occurred.

• Process also wait when they are ready to run (not waiting for any event) But the

processor is

bussy executing other tasks. **6. Show the Peterson's algorithm preserve mutual exclusion**

There can be multiple ways to solve this problem, but most of them require additional hardware support. The simplest & the most popular way to do this is by using peterson's, Algorithm for mutual exclusion It was developed by Peterson in 1981 thought the initial work in this direction was done by the dorus jozef dekkr who come up with Dekker's algorithm is 1960, which was later **refined by Peterson & come to be known as Peters**on's Algorithm

// Filename: peterson_spinlock.c // Use below command to compile: // *gcc*-pthread peterson_spinlock.c-o peterson_spinlock

#include <stdio.h> #include <pthread.h> #include"mythreads.h"

int flag[2]; int turn; const int MAX = 1e9; int ans = 0;

void lock_init()

// Initialize lock by reseting the desire of

// both the threads to acquire the locks. // And, giving turn to one of them. flag[0] = flag[1] = 0; turn = 0;

// Executed before entering critical section void lock(int self)

// Set flag[self] = 1 saying you want to acquire lock flag[self] = 1;

// But, first give the other thre**ad the chance to** // acquire lock turn = 1-self;

//Wait until the other thread looses the desire // to acquire lock or it is your turn to get the lock. while (flag[1-self]==1 && turn=-1-self);

// *E*xecuted after leaving critical section void unlock(int self)

{

```
// You do not desire to acquire lock in future. // This will allow the other thread to acquire // the lock. flag[self] = 0;


// A Sample function run by two threads created // in main void* func(void *s)

int i = 0; int self = (int *)s; printf("Thread Entered: %d\n", self);

lock(self);

// Critical section (Only one thread // can enter here at a time) for (i=0; i<MAX; i++)
ans++;

unlock(self);


// Driver code int main()

// Initialized the lock then fork 2 threads pthread_t p1, p2; lock_init();

// Create two threads (both run func) pthread_create(&p1, NULL, func, (void*)0); pthread_create(&p2, NULL, func, (void*)1);

//Wait for the threads to end. pthread_join(p1, NULL); pthread_join(p2, NULL);

printf("Actual Count: %d | Expected Count: %d\n",
ans, MAX*2);

return 0;
```

7.Does the busy waiting solution using the turn variable (Fig. 2-20- see ref slide at last) work when the two processes are running on a shared-memory multiprocessor, that is, two CPUs sharing a common memory?

```
while (TRUE) {
while (turn = 1); critical_region ();
/= loop
while (TRUE)
while (tuen != 0) = loop =/ critical_region(); turn = 11 noncritical_region();
/

noncritical_reqion()
```

Figure 2-20. A proposed solution to the critical region problem. (a) Process O. (b) Process 1. In both cases, be sure to note the semicolons terminating the while statements.

•

Yes, the busy-waiting solution using **the turn variable will wo**rk in a multiprocessor environment with a shared memory. In the given example, the 'turn' variable is used to control access to a critical region by two processes. In a multiprocessor, the difference is that the two processes are running on different CPU'S. The only condition is that 'turn' itself should be a variable in shared memory.

•

8. Consider a computer that does not have a TSL instruction but does have an instruction to swap the contents of a register and a memory word in a single indivisible action. Can that be used to write a routine enter region such as the one found in Fig. 2-22 (see ref slide)?

```
enter region:
TSL REGISTER, LOCK I copy lock to register and set lock to 1 CMP
REGISTER, #0 | was lock zero? JNE enter region | if it was non zero,
lock was set, so loop RET | return to caller; critical region entered

leave region:
MOVE LOCK, #0
| store a 0 in lock RET | return to caller
```

Figure 2-22. Entering and leaving a critical region using the TSL instruction.

"indivisible no other processor can access the memory word until the instruction is finished

TSL operation reads the contents of the memory word lock into register RX and then stores a nonzero value at the memory address lock. The operations of reading the word and storing into it are guaranteed to be indivisible no other processor can access the memory word until the instruction is finished. The CPU executing the TSL instruction locks the memory bus to prohibit other CPUs from accessing memory until it is done.

Yes, it is possible with swap instruction to solve mutual exclusion..

Swap

another type of hardware assistance for process synchronization

a special hardware instruction that does a complete swap (ordinarily three operations) atomically i.e., the complete swap is executed or no part of it is **given pass-by-reference parameter**s A and B, it swaps their values

•

Swap Solution to the Critical Section Problem

• . .

uses two variables called lock and key

intuition: if lock is false, then a process can enter the critical section, and otherwise it can't initially, lock is false

repeat

var = true while (var == true) swap(lock, var); critical section **lock = false;** remainder section

until false

9. Give a sketch of how an operating system that can disable interrupts could implement semaphores.

To do a semaphore operation, the operating system first disables interrupts. Then it reads the value of the semaphore. If it is doing a down and the semaphore is equal to zero, it puts the calling process on a list of blocked processes associated with the semaphore. If it is doing an up, it must check to see if any processes are blocked on the semaphore.

If one or more processes are blocked, one of them is removed from the list of blocked processes and made runnable. When all these operations have been completed, interrupts can be enabled again.

10. In the solution to the dining philosophers problem (Fig. 2-33 see ref slide at last), why is the state variable set to HUNGRY in the procedure take_forks?

•

In Dining-Philosopher's problem, the variable state is used to denote the intentions and current status of every philosopher. A philosopher can be in either in Hungry, Thinking, or in Eating **state**.

The state Thinking signifies that the philosopher is idle and does not require entering critical region. The state Eating signifies that the philosopher has entered the critical region and is executing. The state Hungry states that the philosopher is ready to enter the critical region. In order to be able to enter critical region, the philosopher should be in Hungry state and be able to possess the two forks. The take_fork() method checks which of the philosopher's is hungry. A hungry philosopher is then put to a test to see if its neighbours are using the fork or not. To check for the fork, the method checks if the left or right neighbours is executing in critical region or not. If neither of the neighbours is in critical region (!Eating), then the hungry philosopher is given the forks.

11. Consider the procedure put_forks in Fig. 2-33 (see ref slide at last). Suppose that the variable state[i] was set to THINKING after the two calls to test, rather than before. How would this change affect the solution?

The change would mean that after a philosopher stopped eating, neither of his neighbors could be chosen next. In fact, they would never be chosen. Suppose that philosopher 2 finished eating. He would run test for philosophers 1 and 3, and neither would be started, even though both were hungry and both forks were available. Similarly, if philosopher 4 finished eating, philosopher 3 would not be started. Nothing would start him.

12. The readers and writers problem can be formulated in several ways with regard to which category of processes can be started when. Carefully describe three different variations of the problem, each one favoring

(or not favoring) some category of processes. For each variation, specify what happens when a reader or a writer becomes ready to access the database, and what happens when a process is finished using the database.

Variation 1: readers have priority.

No writer may start when a reader is active. When a new reader appears, it may start immediately unless a writer is currently active. When a writer finishes, if readers are waiting, they are all started, regardless of the presence of waiting writers.

Variation 2: Writers have priority.

No reader may start when a writer is waiting. When the last active process finishes, a writer is started, if there is one; otherwise, all the readers (if any) are started.

Variation 3: symmetric version

When a reader is active, new readers may start immediately. When a writer finishes, a new writer has priority, if one is waiting. In other words, once we have started reading, we keep reading until there are no readers left. Similarly, once we have started writing, all pending writers are allowed to run..

Nepal