PURBANCHAL UNIVERSITY

SCHOOL OF ENGINEERING (PUSOE)

Operating System

Assignment 1

Submitted by: **Barsha Karn**

Roll no : 13 'A'

Faculty: Computer Engineering

Batch Year : 2020

Year/ Semester: 3rd/5th

College: PUSOE, Brt

Submitted to:

Er. Deeoranjan Dongol sir

Assignment 1

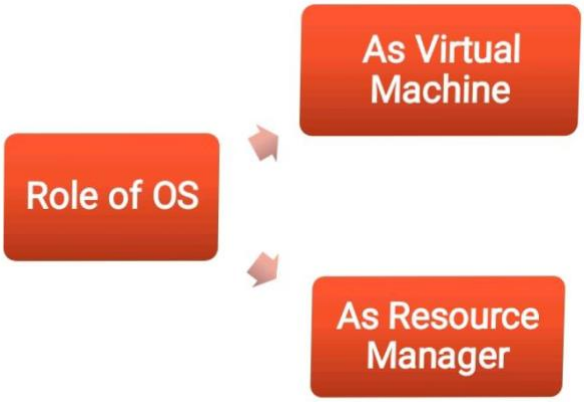
Queation 1. What is an operating system? Explain the roles of operating system.

Ans: An operating system (OS) is software that acts as an intermediary between computer hardware and user applications. It manages and controls the computer's resources, providing an environment in which applications can run efficiently.

In other words, OS is the interface between the application program and hardware level which controls the hardware and makes the hardware usable.

Examples of OS are: Windows, MacOS, Linux, Android, iOS, Unix, and Chrome OS, etc.

The roles of an operating system can be categorised in two ways-



As Virtual Machine, OS plays the following roles:

1. Resource Allocation: The OS allocates hardware resources, such as CPU, memory, and storage, to each virtual machine, ensuring fair distribution and preventing one VM from monopolizing resources.

2. Isolation: Each virtual machine is isolated from others, meaning that software running in one VM cannot directly interfere with the operation of software in another VM.
3. Emulation of Hardware: The OS provides virtualized hardware interfaces to each VM, making each VM believe it has access to dedicated hardware resources.
4. Management: The OS as a virtual machine manager enables the creation, deletion, and management of virtual machines which includes tasks like starting, stopping, suspending, and migrating VMs across physical servers.
5. Snapshotting and Cloning: The OS can allow users to take snapshots of a VM's state at a particular moment, allowing easy rollback to that state if needed.

As Resource Manager, OS plays the following roles:

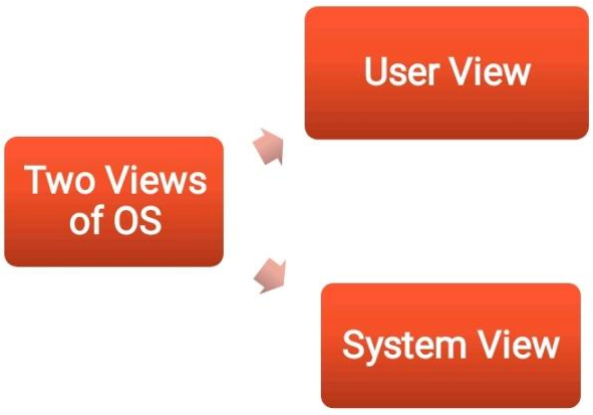
1. CPU Management: The OS schedules and allocates CPU time to different processes and threads. It uses scheduling algorithms to determine which process gets to use the CPU and for how long.
2. Memory Management: The OS manages system memory (RAM), allocating portions of memory to running processes. It uses techniques like virtual memory to provide the illusion of more memory than physically available by using disk space as an extension of RAM.
3. File System Management: The OS manages storage resources and provides a file system for organizing and storing data on disks. It ensures that files are stored, retrieved, and managed

efficiently, and it handles tasks like file creation, deletion, and access permissions.

4. Device Management: The OS manages input and output devices such as printers, disk drives, and network interfaces. It provides a consistent interface for applications to interact with these devices, handling device-specific details and ensuring proper communication.
5. Power Management: The OS implements power-saving features to manage energy consumption. It can control the behavior of hardware components to save power when resources are not in heavy use.

Question 2. Highlight on views of an operating system.

Ans: The operating system may be observed from the viewpoint of the user or the system. It is known as the user view and the system view. There are mainly two types of views of the operating system.



- User View : The user view depends on the system interface that is used by the users. Some systems are designed for a

single user to monopolize the resources to maximize the user's task. In these cases, the OS is designed primarily for ease of use, with little emphasis on quality and none on resource utilization. The user viewpoint focuses on how the user interacts with the operating system through the usage of various application programs. In contrast, the system viewpoint focuses on how the hardware interacts with the operating system to complete various tasks.

- System View : The OS may also be viewed as just a resource allocator. A computer system comprises various sources, such as hardware and software, which must be managed effectively. The operating system manages the resources, decides between competing demands, controls the program execution, etc. According to this point of view, the operating system's purpose is to maximize performance. The operating system is responsible for managing hardware resources and allocating them to programs and users to ensure maximum performance.

Hence, The user viewpoint is all about how the user has to interact with the operating system with the help of various application programs and from the system point of view we see how the hardware has to interact with the operating system for accomplishing the various tasks.

Question 3. Differentiate between time sharing and real time operating system.

Ans:

BASIS FOR COMPARISON	TIME SHARING OPERATING SYSTEM	REAL-TIME OPERATING SYSTEM
Basic	Emphasis on providing a quick response to a request.	It focuses on accomplishing a computational task before its specified deadline.
Computer resources	Shared between the user.	No sharing takes place and events are external to the system.
Process deals with	More than one application simultaneously.	Single application at a time.
Modification of the program	The programs can be modified and written by the users.	No modification is possible.
Response	The response is generated within the second, but there is no compulsion.	User must get the response within the defined time constraint.
Switching	Takes place among the processes.	Does not present

Question 4. Explain virtualization and it's importance.

Ans: Virtualization is a technology that enables the creation of virtual versions of computer hardware, software, storage devices, or network resources. It allows multiple instances of these resources to run independently on a single physical machine or be distributed across multiple machines. Virtualization provides several benefits, including efficient resource utilization, isolation, flexibility, and ease of management.

Virtualization in operating systems (OS) offers several important benefits that contribute to efficient resource utilization, flexibility, and ease of management. Here are some key reasons why virtualization is significant:

1. Optimal Resource Utilization: Virtualization maximizes hardware efficiency by running multiple virtual instances on one physical machine.

2. Isolation and Security: Virtualization provides strong separation between virtual instances, preventing conflicts and security breaches.

3. Server Consolidation : Virtualization reduces the number of physical servers needed, saving space, power, and hardware costs.

4. Flexibility and Scaling: Virtual instances can be easily created, resized, and moved to adapt to changing demands.

5. Disaster Recovery: Virtualization simplifies disaster recovery with easily backed up and restored virtual instances.

6. Testing and Development: Developers use isolated environments to test software changes without affecting production.

7. Legacy Software Support: Virtualization allows running older software on modern hardware and software environments.

8. Efficient Development Environments: Developers work in consistent setups using virtualization, reducing configuration issues.

9. Cloud Computing and Virtual Data Centers: Cloud providers offer scalable resources through virtualization for customers.

10. Quick Provisioning: Virtual machines can be provisioned faster than physical hardware.

11. Green Computing: Virtualization reduces hardware needs, contributing to energy savings and eco-friendly practices.

12.Resource Partitioning: Virtualization divides physical servers into independent parts for better resource allocation.

Question 5. What is system call? Explain the system call flow with the help of diagram.

Ans: A system call in an operating system (OS) is a mechanism that allows user-level programs to request services or resources from the kernel, which is the core part of the OS. System calls provide an interface between applications and the underlying hardware and services of the computer system.

System calls provide a controlled and secure way for user programs to interact with the lower-level functionality of the OS and the hardware. The OS maintains control over these privileged operations to prevent unauthorized or malicious actions by user

programs while allowing legitimate applications to perform necessary tasks.

The system call flow can be described as below:

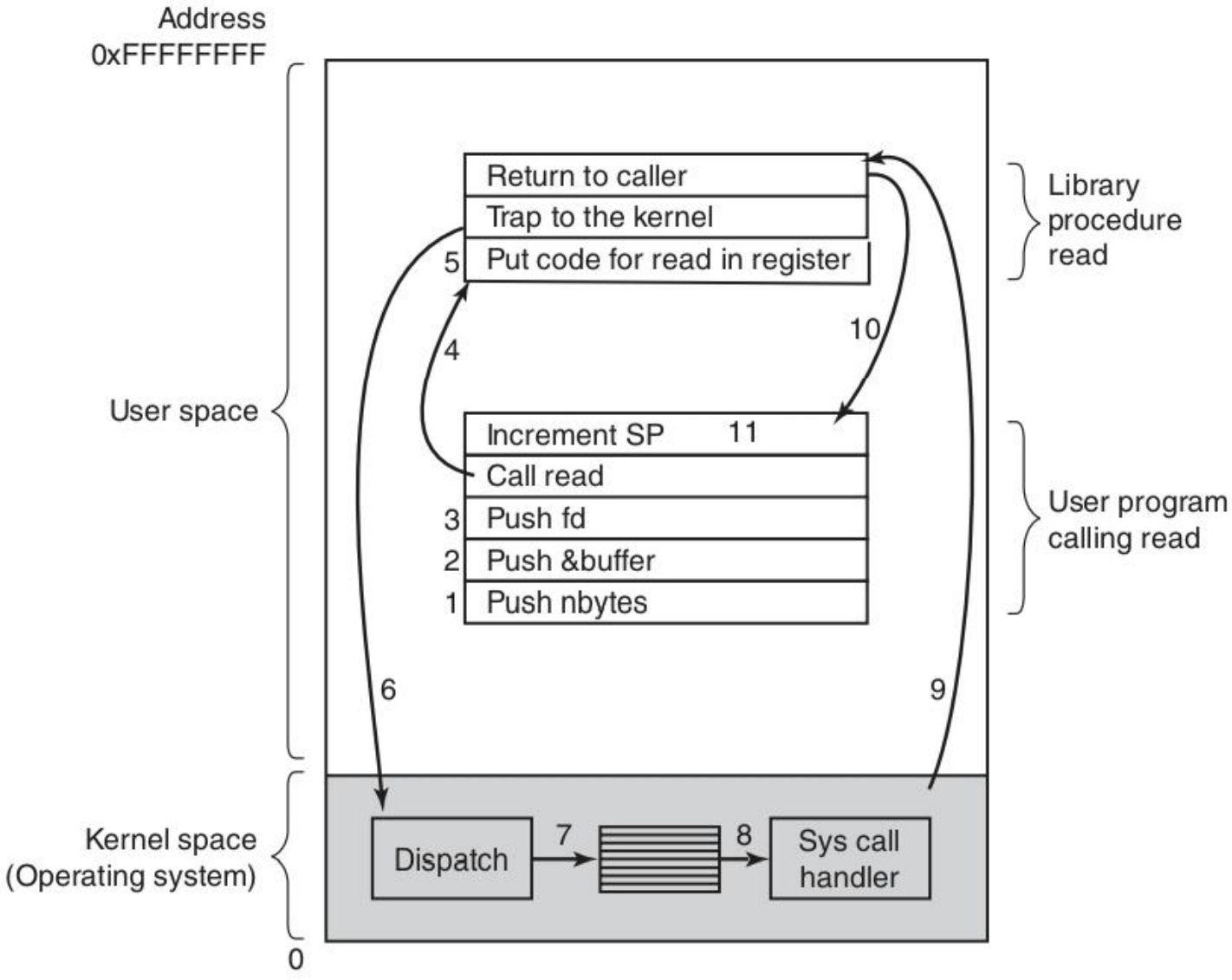


Figure 1-17. The 11 steps in making the system call read(fd, buffer, nbytes).

System calls are performed in a series of steps. To make this concept clearer, Let us examine the read call .

Steps 1-3 : In preparation for calling the read library procedure, which actually makes the read system call, the calling program , First pushes the parameters onto the stack, as shown in steps 1–3 in Fig. 1-17. C and C++ compilers push the parameters onto the stack in reverse order for Historical reasons (having to do with making the first parameter to printf, the format string, appear on top of the stack). The first and third parameters are called by Value, but the second parameter is passed by reference, meaning that the address of the buffer (indicated by &) is passed, not the contents of the buffer.

Step 4: Then comes the Actual call to the library procedure (step 4). This instruction is the normal procedure-call instruction used to call all procedures.

Step 5: The library procedure, possibly written in assembly language, typically puts the system-call number in a place where the operating system expects it, such as a Register (step 5).

Step 6: Then it executes a TRAP instruction to switch from user mode to Kernel mode and start execution at a fixed address within the kernel (step 6). The TRAP instruction is actually fairly similar to the procedure-call instruction in the sense that the instruction following it is taken from a distant location and the return Address is saved on the stack for use later. Depending on the architecture, either it Jumps to a single fixed location or there is an 8-bit field in

the instruction giving the index into a table in memory containing jump addresses, or equivalent.

Step 7: The kernel code that starts following the TRAP examines the system-call number and then dispatches to the correct system-call handler, usually via a table of Pointers to system-call handlers indexed on system-call number (step 7).

Step 8: At that Point the system-call handler runs (step 8).

Step 9: Once it has completed its work, control May be returned to the user-space library procedure at the instruction following the TRAP instruction (step 9).

Step 10: This procedure then returns to the user program in the usual way procedure calls return (step 10).

Step 11: To finish the job, the user program has to clean up the stack, as it does after Any procedure call (step 11). Assuming the stack grows downward, as it often does, the compiled code increments the stack pointer exactly enough to remove the parameters pushed before the call to read. The program is now free to do whatever It wants to do next.

Question 6. What do you mean by trap and interrupt? What is the use of each function?

Ans:

In the context of operating systems, both traps and interrupts are mechanisms used to handle events that require the attention of the

operating system or the CPU. They serve different purposes and have distinct use cases.

Trap: A trap, also known as a software interrupt or exception, is a synchronous event that is triggered by the execution of a specific instruction in a user-level program. It's a deliberate way for a program to request a service or action from the operating system.

Use of Trap:

1. System Calls: User-level programs use traps to request services from the operating system, such as reading from or writing to files, allocating memory, or creating new processes.

2. Error Handling: Traps can also be used for error handling. For example, a divide-by-zero operation triggers a trap, and the operating system can handle the error gracefully.

Interrupt: An interrupt is an asynchronous event that occurs external to the currently executing program. It is typically triggered by hardware devices (such as a timer, keyboard, or disk controller) to request attention from the operating system.

Use of Interrupt:

1. I/O Operations: Interrupts are used to signal that an I/O operation (like data transfer from a disk) has completed or that data is ready to be processed.

2. Timer Interrupts: A timer interrupt is used for preemptive multitasking, allowing the operating system to regain control after a predefined time slice to ensure fair CPU scheduling.

3. Device Notification: Hardware devices can interrupt the CPU when they require immediate attention, like a key press on a keyboard or data arrival on a network interface.

Both traps and interrupts are crucial for managing the interactions between user programs, the operating system, and hardware devices. They ensure that the OS can respond to requests and events efficiently while maintaining control and security over the system's operations.