(1) *What is operating System? Explain the abstract view of the components of a computer system.*

- An operating system (OS) is a collection of software that manages computer hardware resources and provides various services for computer programs. It acts as an intermediary between the user of a computer and the computer hardware.

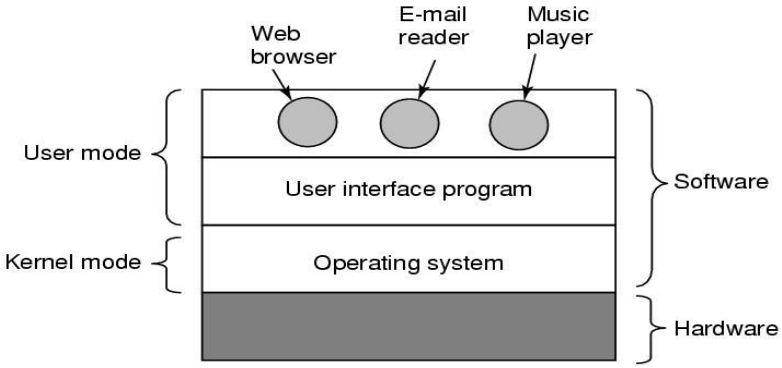

Figure 1-1. A computer system consists of hardware, system programs, and application programs.

- The placement of the operating system is shown in Fig. 1-1. At the bottom is the hardware, which, consists of integrated circuit chips, wires, disks, a key board, a monitor and similar physical devices.
- On the top of the hardware is the software. Operating system runs on the bare hardware and it provides base for the rest of the software.
- Most computers have two modes of operation: **kernel mode and user mode**.
- The operating system is the most fundamental piece of software and runs in kernel mode.
- In this mode it has complete access to all the hardware and can execute any instruction that the machine is capable of executing.
- The rest of the software runs in user mode, in which only a subset of the machine instructions is available. Here we find the command interpreter (shell), compilers, editors, and other system programs.
- Finally, above the system programs are the application programs. These programs are purchased or written by the users to solve their particular problems, such as word processing, spreadsheets, web browser or music player.
- To hide complexity of hardware, an operating system is provided. It consists of a layer of software that (partially) hides the hardware and gives the programmer a more convenient set of instructions to work with.

(2) Give the view of OS as an extended machine.

Operating systems perform two basically unrelated functions: providing a clean abstract set of resources instead of the messy hardware to application programmers and managing these hardware resources.

Operating System as an Extended Machine

- The architecture (instruction set, memory, I/O, and bus structure) of most computers at the machine level language is primitive and awkward to program, especially for input / output operations.
- Users do not want to be involved in programming of storage devices.
- Operating System provides a simple, high level abstraction such that these devices contain a collection of named files.
- Such files consist of useful piece of information like a digital photo, e-mail messages, or web page.
- Operating System provides a set of basic commands or instructions to perform various operations such as read, write, modify, save or close.
- Dealing with them is easier than directly dealing with hardware.
- Thus, Operating System hides the complexity of hardware and presents a beautiful interface to the users.

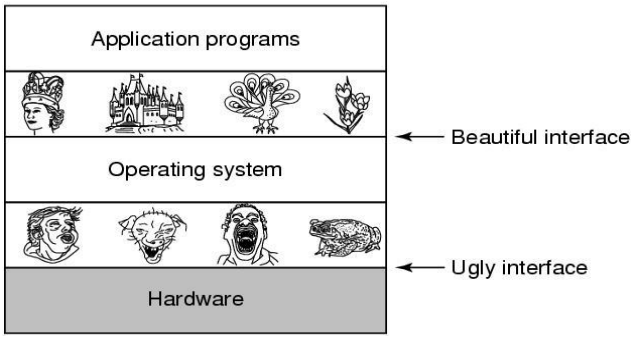

Figure 1-2. Operating Systems turn ugly hardware into beautiful abstractions.

- Just as the operating system shields (protect from an unpleasant experience) the programmer from the disk hardware and presents a simple file-oriented interface, it also conceals a lot of unpleasant business concerning interrupts, timers, memory management, and other low level features.
- In each case, the abstraction offered by the operating system is simpler and easier to

use than that offered by the underlying hardware.

- In this view, the function of the operating system is to present the user with the equivalent of an **extended machine** or **virtual machine** that is easier to work with than the underlying hardware.
- The operating system provides a variety of services that programs can obtain using special instructions called system calls.

(3) Give the view of OS as a Resource Manager.

- The concept of an operating system as providing abstractions to application programs is a top down view.
- Alternatively, bottom up view holds that the OS is there to manage all pieces of a complex system.
- A computer consists of a set of resources such as processors, memories, timers, disks, printers and many others.
- The Operating System manages these resources and allocates them to specific programs.
- As a resource manager, Operating system provides controlled allocation of the processors, memories, I/O devices among various programs.
- Multiple user programs are running at the same time.
- The processor itself is a resource and the Operating System decides how much processor time should be given for the execution of a particular user program.
- Operating system also manages memory and I/O devices when multiple users are working.
- The primary task of OS is to keep the track of which programs are using which resources, to grant resource requests, to account for usage, and to resolve conflicting requests from different programs and users.
- An Operating System is a control program. A control program controls the execution of user programs to prevent errors and improper use of computer.
- Resource management includes multiplexing (sharing) resources in two ways: in time and in space.
- When a resource is time multiplexed, different programs or users take turns using it. First one of them gets to use the resource, then another, and so on.
- For example, CPU and printer are time multiplexed resources. OS decides who will use it

and for how long.

- The other kind of multiplexing is space multiplexing, instead of the customers taking turns, each one gets part of the resource.
- For example, both primary and secondary memories are space multiplexed. OS allocates them to user programs and keeps the track of it.

(4) Explain different types of tasks done by OS. OR Write different services provided by operating system.

- Operating system services and facilities can be grouped into following areas:

Program development

- Operating system provides editors and debuggers to assist (help) the programmer in creating programs.
- Usually these services are in the form of utility programs and not strictly part of core operating system. They are supplied with operating system and referred as application program development tools.

Program execution

- A number of tasks need to be performed to execute a program, such as instructions and data must be loaded into main memory. I/O devices and files must be initialized.
- The operating system handles these scheduling duties for the user.

Access to I/O devices

- Each I/O devices requires its own set of instruction for operations.
- Operating system provides a uniform interface that hides these details, so the programmer can access such devices using simple reads and writes.

Memory Management

- Operating System manages memory hierarchy.
- It keeps the track of which parts of memory are in use and free memory.
- It allocates the memory to programs when they need it.
- It de-allocates the memory when programs finish execution.

Controlled access to file

- In the case of file access, operating system provides a directory hierarchy for easy access and management of files.

- OS provides various file handling commands using which users can easily read, write, and modify files.
- In case of system with multiple users, the operating system may provide protection mechanism to control access to file.

System access

- In case of public systems, the operating system controls access to the system as a whole.
- The access function must provide protection of resources and data from unauthorized users.

Error detection and response

- Various types of errors can occur while a computer system is running, which includes internal and external hardware errors. For example, memory error, device failure error and software errors as arithmetic overflow.
- In case, operating system must provide a response that clears error condition with least impact on running applications.

Accounting

- A good operating system collects usage for various resources and monitor performance parameters.
- On any system, this information is useful in anticipating need for future enhancements.

Protection & Security

- Operating systems provides various options for protection and security purpose.
- It allows the users to secure files from unwanted usage.
- It protects restricted memory area from unauthorized access.
- Protection involves ensuring that all access to system resources is controlled.

(5) Give the features of Batch Operating System.

- Batch operating system is one that processes routine jobs without any interactive user presents. Such as claim processing in insurance and sales reporting etc.
- To improve utilization, the concept of batch operating system was developed.
- Jobs with similar needs were batched together and were run through the computer as a group.
- Thus, the programmer would leave their program with operator, who in turn would

sort program into batches with similar requirements.

- The operator then loaded a special program (the ancestor of today's operating system), which read the first job from magnetic tape and run it.
- The output was written onto a second magnetic tape, instead of being printed.
- After each job finished, the operating system automatically read the next job from the tape and began running it.
- When the whole batch was done, the operator removed the input and output tapes, replaced the input tape with the next batch, and brought the output tape for offline printing.
- With the use of this type of operating system, the user no longer has direct access to machine.
- Advantages:
  - Move much of the work of the operator to the computer.
  - Increase performance since it was possible for job to start as soon as the previous job finished.
- Disadvantages:
  - Large Turnaround time (the amount of time taken to complete a request).
  - More difficult to debug program.
  - Due to lack of protection scheme one batch job can affect pending jobs.

(6) Explain the features of Time Sharing System.

- Time Sharing is a logical extension of multiprogramming.
- Multiple jobs are executed simultaneously by switching the CPU back and forth among them.
- The switching occurs so frequently (speedy) that the users cannot identify the presence of other users or programs.
- Users can interact with his program while it is running in timesharing mode.
- Processor's time is shared among multiple users. An interactive or hands on computer system provides online communication between the user and the system.
- A time shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time shared computer. Each user has at least one separate program in memory.
- A time shared operating system allows many users to share computer simultaneously. Since each action or command in a time shared system tends to be short, only a little CPU time is needed for each user.

- Advantages :-
  - Easy to use
  - User friendly
  - Quick response time
- Disadvantages:-
  - If any problem affects the OS, you may lose all the contents which have stored already.
  - Unwanted user can use your own system in case if proper security options are not available.

(7) Explain the features of Real Time Operating System.

- A real time operating system is used, when there are rigid (strict) time requirements on the operation of a processor or the flow of data.
- It is often used as a control device in a dedicated application. Systems that control scientific experiments, medical imaging systems, and industrial control system are real time systems. These applications also include some home appliance system, weapon systems, and automobile engine fuel injection systems.
- Real time Operating System has well defined, fixed time constraints. Processing must be done within defined constraints or the system will fail.
- Since meeting strict deadlines is crucial in real time systems, sometimes an operating is simply a library linked in with the application programs.
- There are two types of real time operating system,
  - Hard real system:
    - ✓ This system guarantees that critical tasks complete on time.
    - ✓ Many of these are found in industrial process control, avionics, and military and similar application areas.
    - ✓ This goal says that all delays in the system must be restricted.
  - Soft real system:
    - ✓ In soft real-time system, missing an occasional deadline, while not desirable, is acceptable and does not cause any permanent damage.
    - ✓ Digital audio or multimedia systems fall in this category.
- An example of real time system is **e-Cos**.

(8) Explain different types of OS.

Mainframe Operating Systems

- The operating system found in those room sized computers which are still found in major corporate data centers. These computers differ from personal computers in terms of their I/O capacity.
- They typically offer three kinds of services: batch, transaction processing, and timesharing.
- **Batch operating system** is one that processes routine jobs without any interactive user presents, such as claim processing in an insurance and sales reporting etc.
- **Transaction processing** system handles large numbers of small requests, for example check processing at a bank and airline reservation.
- **Time sharing** allows multiple remote users to run jobs on the computer at once, such as querying a database.
- An example mainframe operating system is **OS/390** and a descendant of **OS/360**.

Server Operating Systems

- They run on servers, which are very large personal computers, workstations, or even mainframes.
- They serve multiple users at once over a network and allow the users to share hardware and software resources.
- Servers can provide print service, file service or web service.
- Typically server operating systems are **Solaris**, **FreeBSD**, and **Linux** and **Windows Server 200x**.

Multiprocessor Operating Systems

- An increasingly common way to get major group computing power is to connect multiple CPUs into a single system. Depending on precisely how they are connected and what is shared, these systems are called parallel computers, multicomputers, or multiprocessors. The operating systems used in this system are multiprocessor operating system.
- They need special operating systems, but often these are variations on the server OS with special features for communication, connectivity and consistency.
- Multiprocessor operating systems includes **Windows** and **Linux**, run on

multiprocessors.

Personal Computer Operating Systems

- The next category is the personal computer operating system. All Modern computers support multiprogramming, often with more than one programs started up at boot time. Their job is to provide good support to a single user.
- They are widely used for word processing, spreadsheets and Internet access.
- Common examples of personal computer operating system are **Linux**, **FreeBSD**, **Windows Vista**, and **Macintosh** operating system.

Handhelds Computer Operating Systems

- Continuing on down to smaller and smaller systems, we come to handheld computers. A handheld computer or PDA (personal digital assistant) is a small computer that fits in a pocket and performs a small number of functions, such as electronics address book and memo pad.
- The OS that runs on handhelds are increasingly sophisticated with the ability to handle telephony, photography and other functions.
- One major difference between handhelds and personal computer OS is that the former do not have multi gigabyte hard disks.
- Two of the most popular operating systems for handhelds are **Symbian** OS and **Palm** OS.

Embedded Operating Systems

- Embedded systems run on the computers that control devices that are not generally thought of as computers and which do not accept user installed software.
- The main property which distinguishes embedded systems from handhelds is the certainty that no untrusted software will ever run on it.
- So, there is no need for protections between applications, leading to some simplifications.
- Systems such as **QNX** and **VxWorks** are popular embedded operating system.

Sensor Node Operating Systems

- Networks of tiny sensor nodes are being deployed (developed) for numerous purposes. These nodes are tiny computers that communicate with each other and with a base station using wireless communication.
- These sensor networks are used to protect the perimeters of buildings, guard

national borders, detect fires in forests, measure temperature and precipitation for weather forecasting, glean information about enemy movements on battlefields, and much more.

- Each sensor node is a real computer, with a CPU, RAM, ROM and one or more environmental sensors.
- It runs a small but real operating system, usually one that is event driven, responding to external events or making measurements periodically based on internal clock.
- All the programs are loaded in advance which makes the design much simpler.
- **TinyOS** is a well-known operating system for a sensor node.

Real Time Operating Systems

- These systems are characterized by having time as a key parameter.
- Real time operating system has well defined, fixed time constraints. Processing must be done within define constraints or the system will fail.
- Types of Real Time Operating System:
  - ✓ Hard real time system
    - Many of these are found in industrial process control, avionics, and military and similar application areas.
    - These systems must provide absolute guarantees that a certain action will occur be a certain time.
  - ✓ Soft real time system
    - Missing an occasional deadline, while not desirable is acceptable and does not cause any permanent damage.
    - Digital audio, digital telephone and multimedia systems fall into this category.
- An example of real time system is **e-Cos**.

Smart Card Operating Systems

- The smallest operating systems run on smart cards, which are credit card sized devices containing a CPU chip. They have very severe processing power and memory constraints.
- Some of them can handle only a single function such as electronic payments but others can handle multiple functions on the same card.

(9) Explain different types of operating system structure. OR
Explain architectures of different operating system structure.

Monolithic system

- In this approach the entire operating system runs as a single program in kernel mode.
- The operating system is written as a collection of procedures, linked together into a single large executable binary program.
- When this technique is used, each procedure in the system has a well-defined interface in terms of parameters and results, and each one is free to call any other one, if the latter provides some useful computation that the former needs.
- To construct the actual object program of the operating system, when this approach is used, one first compiles all the individual procedure and then binds (group) them all together into a single executable file using the system linker.
- The services (system calls) provided by the operating system are requested by putting the parameters in a well-defined place (e.g., on the stack) and then executing a trap instruction.
- This instruction switches the machine from user mode to kernel mode and transfers control to the operating system.
- The operating system then fetches the parameters and determines which system call is to be carried out.
- This organization suggests a basic structure for the operating system.
  - ✓ A main program that invoke (call up) the requested service procedure.
  - ✓ A set of service procedures that carry out the system calls.
  - ✓ A set of utility procedures that help the service procedure.
- In this model, for each system call there is one service procedure that takes care of it and executes it.
- The utility procedures do things that are needed by several services procedure, such as fetching data from user programs.
- This division of the procedure into three layers is shown in figure 1-3

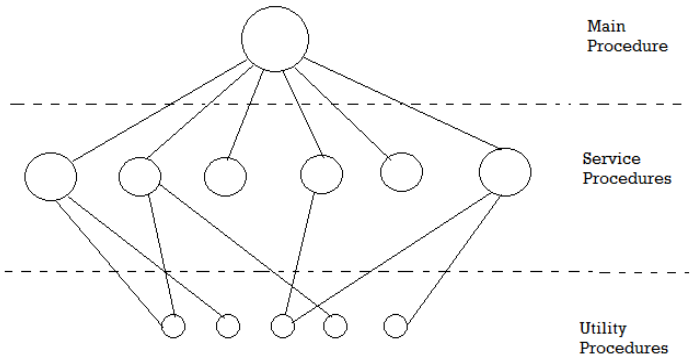

Figure 1-3. A simple structuring model for a monolithic system.

Layered system

- In this system, operating system is organized as a hierarchy of layers, each one constructed upon the one below it as shown below in figure 1-4.

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Figure 1-4. Structure of THE operating system.

- The first system constructed in this way was the THE system.
- The system had six layers.
- Layer 0 dealt with allocation of the processor, switching between processes when interrupts occurred or timers expired.
- Layer 0 provided the basic multiprogramming of the CPU.
- Layer 1 did the memory management. It allocated space for process in main memory and on a 512K word drum used for holding parts of processes for which there was no room in main memory.
- Layer 2 handled communication between each process and the operator console (i.e. user).
- Layer 3 takes care of managing the I/O devices and buffering the information

streams to and from them.

- Layer 4 was where the user programs were found.
- The system operator process was located in layer 5.
- A further generalization of the layering concept was present in the MULTICS system.
- Instead of layers, MULTICS was described as having a series of concentric rings, with the inner ones being more privileged than the outer ones.
- When a procedure in an outer ring wanted to call a procedure in an inner ring, it had to make the equivalent of a system call, that is, a TRAP instruction whose parameters were carefully checked for validity before allowing the call to proceed.
- Although the entire operating system was part of the address space of each user process in MULTICS, the hardware made it possible to designate individual procedures (memory segments, actually) as protected against reading, writing, or executing.

Microkernel

- With the layered approach, the designers have a choice where to draw the kernel user boundary.
- Traditionally, all the layers went in the kernel, but that is not necessary.
- In fact, a strong case can be made for putting as little as possible in kernel mode because bugs in the kernel can bring down the system instantly.
- In contrast, user processes can be set up to have less power so that a bug may not be fatal.
- The basic idea behind the microkernel design is to achieve high reliability by splitting the operating system up to into small, well defined modules, only one of which the microkernel runs in kernels mode and the rest of all are powerless user processes which would run in user mode.
- By running each device driver and file system as separate user processes, a bug in one of these can crash that component but cannot crash the entire system.
- Examples of microkernel are Integrity, K42, L4, PikeOS, QNX, Symbian, and MINIX 3.
- MINIX 3 microkernel is only 3200 lines of C code and 800 lines of assembler for low level functions such as catching interrupts and switching processes.
- The C code manages and schedules processes, handles inter-process communication and offer a set of about 35 systems calls to the rest of OS to do its work.

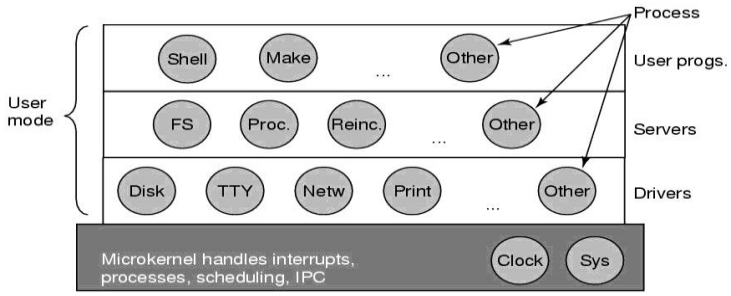

Figure 1-5. Structure of MINIX 3 system.

- The process structure of MINIX 3 is shown in figure 1-5, with kernel call handler labeled as *Sys*.
- The device driver for the clock is also in the kernel because the scheduler interacts closely with it. All the other device drivers run as separate user processes.
- Outside the kernel, the system is structured as three layers of processes all running in user mode.
- The lowest layer contains the device driver. Since they run in user mode they do not have access to the I/O port space and cannot issue I/O commands directly.
- Above driver is another user mode layer containing servers, which do most of the work of an operating system.
- One interesting server is the **reincarnation server**, whose job is to check if the other servers and drivers are functioning correctly. In the event that a faulty one is detected, it is automatically replaced without any user intervention.
- All the user programs lie on the top layer.

Client Server Model

- A slight variation of the microkernel idea is to distinguish classes of processes in two categories.
- First one is the servers, each of which provides some services, and the second one is clients, which use these services.
- This model is known as the Client Server model.
- Communication between clients and servers is done by message passing.
- To obtain a service, a client process constructs a message saying what it wants and sends it to the appropriate services.
- The service then does the work and sends back the answer.
- The generalization of this idea is to have the clients and servers run on different

computers, connected by a local or wide area network.

- Since a client communicates with a server by sending messages, the client need not know whether the message is handled locally in its own machine, or whether it was sent across a network to a server on a remote machine.
- A PC sends a request for a Web page to the server and the Web page comes back. This is a typical use of client server model in a network.

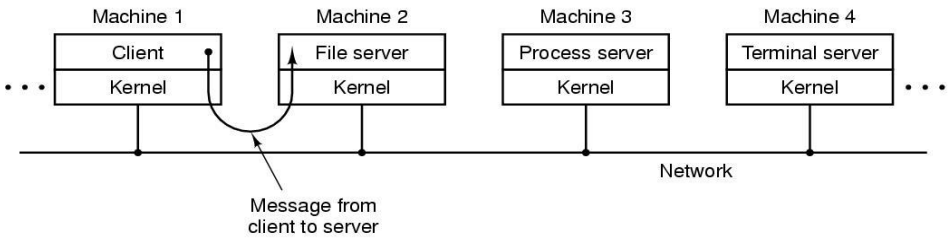

Figure 1-6. The client server model over a network.

Virtual Machine

- The initial releases of OS/360 were strictly batch systems. But many users wanted to be able to work interactively at a terminal, so OS designers decided to write timesharing systems for it.

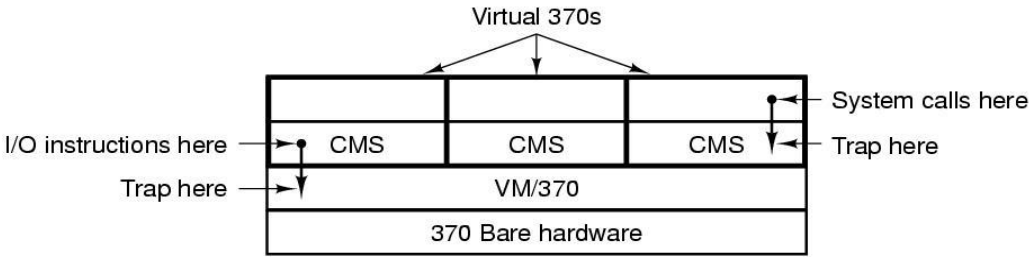

Figure 1-7. The structure of VM/370 with CMS.

- The heart of the system, known as the virtual machine monitor, run on the bare hardware and does the multiprogramming, providing not just one but several virtual machines to the next layer up.
- Each virtual machine is identical to the true hardware; each one can run any OS that will run directly on the bare hardware.
- Different virtual machines can run different operating systems.
- On VM/370, some run OS/360 while the others run single user interactive system called CMS (Conversational Monitor System) for interactive time sharing users.

- When CMS program executed a system call, a call was trapped to the operating system in its own virtual machine, not on VM/370. CMS then issued the normal hardware I/O instruction for reading its virtual disk or whatever was needed to carry out the call.
- These I/O instructions were trapped by VM/370 which then performs them.
- The idea of a virtual machine is heavily used nowadays in a different context.
- An area where virtual machines are used, but in a somewhat different way, is for running Java programs.
- When Sun Microsystems invented the Java programming language, it also invented a virtual machine (i.e., a computer architecture) called the **JVM (Java Virtual Machine).**
- The Java compiler produces code for JVM, which then typically is executed by a software JVM interpreter.
- The advantage of this approach is that the JVM code can be shipped over the Internet to any computer that has a JVM interpreter and run there.

Virtual Machines Rediscovered

- Many huge companies have considered virtualization as a way to run their mail servers, Web servers, FTP servers and other servers on the same machine without having a crash of one server bring down the rest.
- Virtualization is also popular in Web hosting world.
- Web hosting company offers virtual machines for rent, where a single physical machine can run many virtual machines; each one appears to be a complete machine.
- Customers who rent a virtual machine can run any OS or software they want to but at a fraction of the cost of dedicated server.
- Another use of virtualization is for end users who want to be able to run two or more operating systems at the same time, say Windows and Linux, because some of their favorite application packages run on one and some am on the other.
- This situation is illustrated in Fig. 1-8(a), where the term "virtual machine monitor" has been renamed type 1 **hypervisor** in recent years.
- VMware Workstation is a type 2 hypervisor, which is shown in Fig. 1-8(b). In contrast to type 1 hypervisors, which run on the bare metal, type 2 hypervisors run as application programs on top of Windows, Linux, or some other operating system,

known as the **host operating system.**

- After a type 2 hypervisor is started, it reads the installation CD-ROM for the chosen **guest operating system** and installs on a virtual disk, which is just a big file in the host operating system's file system.
- When the guest operating system is booted, it does the same thing it does on the actual hardware, typically starting up some background processes and then a GUI.
- Some hypervisors translate the binary programs of the guest operating system block by block, replacing certain control instructions with hypervisor calls.
- The translated blocks are then executed and cached for subsequent use.
- A different approach to handling control instructions is to modify the operating system to remove them. This approach is not true virtualization, but **para-virtualization.**

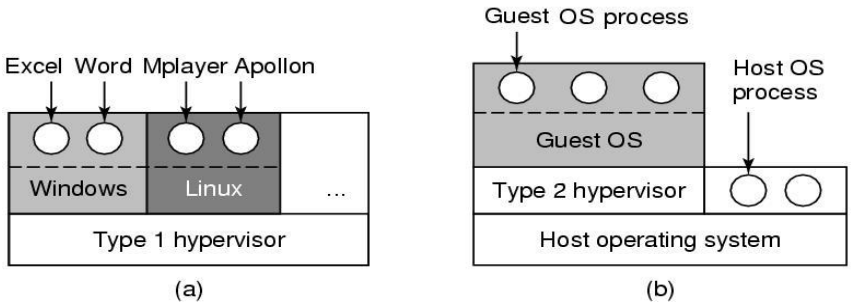

Figure 1-8. Type 1 & 2 Hypervisor.

Exokernels

- Rather than cloning (copying) the actual machine, as is done with virtual machines, another strategy is partitioning it.
- In other words, giving each user a subset of the resource.
- For example, one virtual machine might get disk blocks 0 to 1023, the next one might get block 1024 to 2047, and so on.
- Program running at the bottom layer (kernel mode) called the exokernel. Its job is to allocate resources to virtual machines and then check attempt to use them to make sure no machine is trying to use somebody else's resources.
- The advantage of the exokernel scheme is that it saves a layer of mapping.
- In the other designs, each virtual machine thinks it has its own disk, with blocks running from 0 to some maximum, so the virtual machine monitor must maintain tables to remap disk addresses.
- In exokernel remapping is not needed. The exokernel need only keep track of which

virtual machine has been assigned which resource.

(10) What is a system call? How it is handled by an OS? OR Write a short note on system calls.

- The interface between the operating system and the user programs is defined by the set of system calls that the operating system provides.
- The system calls available in the interface vary from operating system to operating system.
- Any single-CPU computer can execute only one instruction at a time.
- If a process is running a user program in user mode and needs a system service, such as reading data from a file, it has to execute a trap or system call instruction to transfer control to the operating system.
- The operating system then figures out what the calling process wants by inspecting the parameters.
- Then it carries out the system call and returns control to the instruction following the system call.

Following steps describe how a system call is handled by an operating system.

- To understand how OS handles system calls, let us take an example of read system call.
- Read system call has three parameters: the first one specifying the file, the second one pointing to the buffer, and the third one giving the number of bytes to read.
- Like nearly all system calls, it is invoked from C programs by calling a library procedure with the same name as the system call: read.
- A call from a C program might look like this:

  count = read(fd, buffer, nbytes);

- The system call return the number of bytes actually read in count.
- This value is normally the same as nbytes, but may be smaller, if, for example, end-of-file is encountered while reading.
- If the system call cannot be carried out, either due to an invalid parameter or a disk error, count is set to -1, and the error number is put in a global variable, errno.
- Programs should always check the results of a system call to see if an error occurred.
- System calls are performed in a series of steps.
- To make this concept clearer, let us examine the read call discussed above.
- 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. 1-9.

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

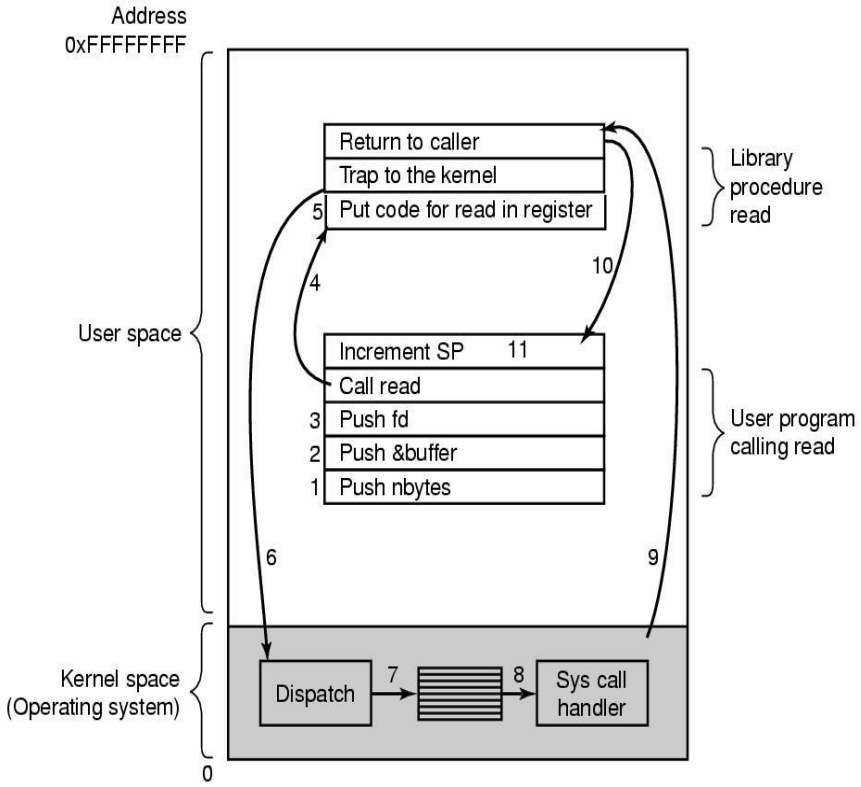

Figure 1-9. The 11 steps in making the system call read(fd, buffer, nbytes).

- 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 kernel code that starts 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).
- At that point the system call handler runs (step 8).

Process management	
Call	**Description**
pid = for k( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status
File management	
Call	**Description**
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information
Director and file system management	
Call	**Description**
s = mkdir(name,mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system
Miscellaneous	
Call	**Description**
s = chdir(dir name)	Change the working directory
s = chmod(name,mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Table 1-1. Some of the major POSIX system calls.

- Once the system call handler has completed its work, control may be returned to the user-space library procedure at the instruction following the TRAP instruction (step 9).
- This procedure then returns to the user program in the usual way procedure calls return (step 10).
- To finish the job, the user program has to clean up the stack, as it does after any procedure call (step 11).