Q1.1: What is an Operating System & What are the three main purposes of an operating system?
An operating system is software that manages computer hardware and communication between applications, programs, and hardware components. The three main purposes of using the operating system are the following.

- To **provide an interface** for a user to communicate with hardware components in a convenient and efficient manner and on which the user does not need to know the details of components.
- To act as a **resource allocator** so that different computer resources are allocated to specific programs and users as fairly and efficiently as possible.
- To act as a **control program** that manages the execution of the user programs to prevent errors and improper use of the computer.

Q1.2: We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to "waste" resources? Why is such a system not really wasteful?
It is appropriate to "waste" resources in single-user systems if the user gets better interaction --- that is, if we improve ease of use.
Such system is not really wasteful because, as we said, the user gets better interaction. The other important fact is that single-user systems generally require less computer resources than multi-user systems.

Q1.3: What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?
Recall that real-time operating system has fixed time constraints on its operations of a processor and flow of data. This means that the system will fail if some process takes to long to complete. Therefore, programmers have to pay a lot of attention to optimization and resource allocation.
Result: Programmers have to pay a lot of attention to optimization and resource allocation. Additionally, it is of utmost importance that every process runs according to schedule and that it runs precisely as long as schedule says.

Q1.4: Keeping in mind the various definitions of operating system, consider whether the operating system should include applications such as web browsers and mail programs. Argue both that it should and that it should not, and support your answers.
Reasons for:
There are users which would like for OS to come with programs such as web browser or e-mail client out-of-the-box so that they do not have to install them themselves. This is simply convenient for user who just want out-of-the-box system and do not care about details of those programs.
Reason against:
Tight coupling of applications to OS is not something every user wants. Some user want functionality of OS but choose their own applications for high-level tasks which are not in itself part of OS.

Q1.5: How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security)?
There are some instructions which can cause harm to the system. We designate such instructions as **privileged instructions** which can be executed only in kernel mode.
When we try to execute them in the user mode, the hardware will not execute them --- they are considered illegal and hardware will trap them to the operating system.
Result: It helps because potentially dangerous instructions can be forbidden by OS in user-mode

Q1.6: Which of the following instructions should be privileged?
a. Set value of timer.
b. Read the clock.
c. Clear memory.
d. Issue a trap instruction.
e. Turn off interrupts.
f. Modify entries in device-status table.
g. Switch from user to kernel mode.
h. Access I/O device.
Privileged instructions are the instructions that are only executed in kernel mode. If a privileged instruction is attempted to get executed in user mode, that instruction will get ignored and treated as an illegal instruction

a. Timer is an important part of the system, so it must be protected. This means that this should be a **privileged** instruction.
b. Reading the clock is not a dangerous operation, so there is no need to set this is a privileged instruction.
c. Clearing memory may prove fatal, so this operating must be protected. So, it should be a **privileged** instruction
d. Everyone can issue a trap instruction, so there is no need to set this as a privileged instruction.
e. Since interrupts are an important part of the computer system, we must stop the unwanted attempts to turn them off. This means that this must be **privileged** instruction.
f. This should obviously be a privileged instruction.
g. If we do not designate this as a privileged instruction, everyone could just go into kernel mode and then execute any privileged instruction. This means that this must be privileged instruction.
h. Accessing I/O device must be controlled, so this should be a **privileged** instruction.

Q1.7: Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

1. Such operating system cannot be modified or updated. This can prove fatal if we notice some critical bug, since there is not way to correct it.
2. All passwords and other credentials of users must be kept in an unprotected memory, since we cannot store it in the protected memory.

Q1.8: Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?

1. CPUs that support virtualization have one additional mode which indicates when the virtual machine manager is in control. It has more privileges than in the user mode, but less than in the kernel mode.
2. We can also use so called privileged levels — when the system has a higher privileged level it can execute more instructions.

Result: Virtualization and privileged levels

Q1.9: Timers could be used to compute the current time. Provide a short description of how this could be accomplished.

Timers can be used to calculate current time in the following way:
A process can save a local state and set a timer for some time. It can then be suspended by OS.
After timer has passed, an interrupt will wake up the program. Based on current time which was saved by process .before being suspended and current time when it was awakened process can measure time.
This makes an assumption that process can obtain current time with a system call for example.
If process can not obtain current time, it can count amount of times it was suspended and measure time like that. Obviously, second way of time measuring is not ideal.
Result: Using combination of local states, current time (if available) and interrupts process can either count interrupts or measure time passed.

Q1.10: Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

Two reasons why they are useful:

1. Cache is a relatively fast memory, so getting some data from cache is faster than getting it from main or secondary memory.
2. Caches are particularly useful when two or more components need to exchange data - transfer speeds are much faster when using caches, and the transfer can be done without using the main memory.

Therefore, we see that caches solve the transfer problem transfer speeds are much faster, so the component receiving data from some other component does not have to wait as long as it would have to if we didn't use caches. Notice that this means that means that caches improve the efficiency of the whole system.
The main problems caused by caches are inconsistency (when data is changed in the main memory or in the memory of some component, the data in caches must also be updated) and cache misses (if the data is not in the cache we lost the time for checking if the data is located in the cache, and we lose some additional time to transfer the data to the cache).
The main problem with the last question (why don't we use bigger caches) is that faster memories are more expensive. Therefore, such enormous cache wouldn't be affordable.
Result:

Useful: caches are fast, and reduce transfer times.
Problems: inconsistency and cache misses.
Caches are expensive.

Q1.11: Distinguish between the client–server and peer-to-peer models of distributed systems.

Client-server model:

In this model, clients send requests to servers to execute some action. Some information is then sent by the server to the client which requested it. This can be the result of the action (for example, if the client wishes to log-in, the server will check whether the received password is correct and send the information whether the log-in was successful to the client), or some file (for example, your browser received a HTML file when you opened this solution).
The main advantage of this model is that it is more secure than the peer-to-peer model (all authentication is done on the server).

Peer-to-peer model:

In this model, all connected systems are called peers, and they may act as either a client or a server, depending on whether they are providing or requesting some service. Since there are no servers, which are so called "bottlenecks"' in the client-server model, this model is generally faster. Also, it is more simple and less expensive to implement for fixed number of peers. However, note that it is not as scalable as client-server model --- as the number of users keeps increasing, this model becomes much more difficult to maintain.
Result: Client-server model is more secure and it is scalable, while peer-to-peer model is generally faster and less expensive.

Q1.12: How do clustered systems differ from multiprocessor systems? What is required for two machines belonging to a cluster to cooperate to provide a highly available service?

The difference is that clustered systems contain two or more independent systems (in multiprocessor systems, processors are not so independent of each other).
For clustered systems to be efficient, we need to be sure that the system will not halt if one or more systems in the cluster fail. This is done by adding a level of redundancy in the system. For example, if some system fails, some other system can take control of its storage and rerun the required applications.

Result:

Difference: clustered systems contain two or more independent systems.
Level of redundancy.

Q1.13: Consider a computing cluster consisting of two nodes running a database. Describe two ways in which the cluster software can manage access to the data on the disk. Discuss the benefits and disadvantages of each.

We can use two types of clustering:

1. Asymmetric clustering here one node will do all work, while the other will be in standby and will activate in the case that the active node fails. The benefit is that the system will clearly not halt if the active system fails. The disadvantage is that it is not really efficient; we have two nodes, but we actively only use one of them.

2. Symmetric clustering to be more precise, we will observe so called parallel clustering. In this case multiple nodes (here, 2 of them) can read the data from the disk. The benefit is that a system with parallel clustering is far more efficient than a system with asymmetric clustering. The disadvantage is that we need special software to implement the parallel clustering.

Result: HINT: asymmetric and symmetric (parallel) clustering.

Q1.14: What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?
The **interrupt** is used to inform the computer about terminating the current process and initiating a new one for a circuit. For this process, it sends a **computer signal**. When we compare an interrupt with a **trap**, we can see that an interrupt is created by **hardware**. But a tap is created by the **software**.
The main reason to generate a trap is to let a program or task be executed continuously. That's why program continuity will never be lost.
Yes, traps can also be generated intentionally by a user program.

- Because sometimes arithmetic failures occur in the system. If these errors are not gone, solving these problems will be more complicated in the future. That's why the user intervenes with traps.
- Also, **OS procedures** are called by the trap.

Q1.18: Many SMP systems have different levels of caches; one level is local to each processing core, and another level is shared among all processing cores. Why are caching systems designed this way?
Local caches are much faster than shared caches. The data which is expected to be used by some processor is put into its local cache, but it is also put into the shared cache (if it is not already there --- notice that it is plausible that it was in the shared cache before it was transferred to the local cache). We put the data into the shared cache so that it can be quickly transferred to a local cache of another processor if it starts some process which needs this data.

Q1.21: Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following processing environments:
a. Single-processor systems
b. Multiprocessor systems
c. Distributed systems

1. **Single:** In a single-processor system, there is only one cache and only one process being executed sequentially by the CPU. Since there are no other users to simultaneously modify or access data, the cache is reliably up-to-date.
2. **Multi:** In a multiprocessor system, each CPU has its own cache and/or a shared cache and the main memory. Without cache coherence, each CPU might store a local copy of some data, say an integer I that was stored to the cache from main memory, and changes to I will not persist between CPUs. Thus, accessed data may be out of date or inconsistent between processors.

3. **Distributed:** In a distributed system, data is often shared between computers by keeping separate copies of files on each computer. However, changes to one file will not be visible to the rest of the system unless each copy is constantly kept up-to-date.

Q1.22: Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.
One such mechanism is called virtual memory. Every process which is running on computer will think it has access to all of the RAM and only it has access to the RAM. In reality, it has access to a small portion and when it addresses some location in RAM, OS will use a table to calculate a physical address in RAM from address requested by that process.
Another way can be bound checking. If a process tries to access some addresses in RAM which are needed by OS, request can be ignored and interrupt can be raised.
Result: Paging algorithm and virtual memory.

Q1.23: Which network configuration—LAN or WAN—would best suit the following environments?
a. A campus student union
b. Several campus locations across a statewide university system
c. A neighborhood
LAN (local area network) is a group of computers and network devices connected together, usually within the same building. \par WAN (ide area network) cover larger areas and even allow computers in different nations to connect. As a result we can conclude that

- A campus student union consists of computers connected together which are in a specific area inside the campus probably connecting offices in several buildings. Thus is is a LAN.
- Several campus locations across a statewide university system are part of a larger network which is distributed across a city so it is a WAN
- A neighborhood is made of several houses where each house or building or business in it is a LAN. For example each family might have several PC's connected in a LAN. Since several LAN connected together make a WAN the neighborhood belongs to a WAN

Result:
LAN
WAN
LAN

Q1.24: Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.
Some challenges include: memory limitation, size of the device, inability for user to upgrade hardware or components in order to speed up the device.
Result: Memory limitation, closed platform, energy consumption, size of the device

Q1.25: What are some advantages of peer-to-peer systems over client–server systems?
In a client-server system, the speed of the server is a limiting factor on the provision of services and data, while in a peer-to-peer system, services and data can be provided by many peers at

once. Decentralized services are cheaper to provide since the creator of the service or files does not have to run a server. Additionally, peer-to-peer networks can provide anonymity and are more robust.

Q1.26: Describe some distributed applications that would be appropriate for a peer-to-peer system.
Common distributed applications include file-sharing networks, multimedia streams such as a video chat, cryptocurrency, and anonymous web browsers.

Q1.27: Identify several advantages and several disadvantages of open-source operating systems. Identify the types of people who would find each aspect to be an advantage or a disadvantage.
Advantages:

- The user enjoys open-source software free of charge, along with an active community of volunteer developers who help maintain it.
- Open source software is considered safer because of the availability of the source code for anyone to analyse and debug.
- By making their software open source, companies can benefit from increased usage and feedback, as well as an improved public image.

Disadvantages:

- Companies are likely to lose money from the lack of sales of software licenses.
- Some open source communities are underserved, leaving the software dilapidated.

Computer system can be divided into four components:

- Hardware – provides basic computing resources
    - CPU, memory, I/O devices
- Operating system
    - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
- Users
    - People, machines, other computers

What Operating System do?
User's view of the computer:

- The operating system is designed mostly for **ease of use**, with some attention paid to performance and security and none paid to resource utilization
- Increasingly, many users interact with mobile devices such as smartphones and tablets—devices that are replacing desktop and laptop computer systems for some users.

- The user interface for mobile computers generally features a **touch screen**, where the user interacts with the system by pressing and swiping fingers across the screen rather than using a physical keyboard and mouse. Many mobile devices also allow users to interact through a **voice recognition**
- Some computers have little or no user interface, such as **embedded computers** in devices and automobiles

System View:

- From the computer's point of view, the operating system is the program most intimately involved with the hardware. we can view an operating system as a **resource allocator.**
- A computer system has many resources that may be required to solve a problem: CPU time, memory space, storage space, I/O devices, and so on. The operating system acts as the manager of these resources.
- A slightly different view of an operating system emphasizes the need to control the various I/O devices and user programs. An operating system is a control program. A **control program** manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.

Operating System Definition:

The operating system includes the always running **kernel**, **middleware frameworks** that ease application development and provide features, and **system programs** that aid in managing the system while it is running.

Computer-System Organization:

One or more CPUs, device controllers connect through common **bus** providing access to shared memory

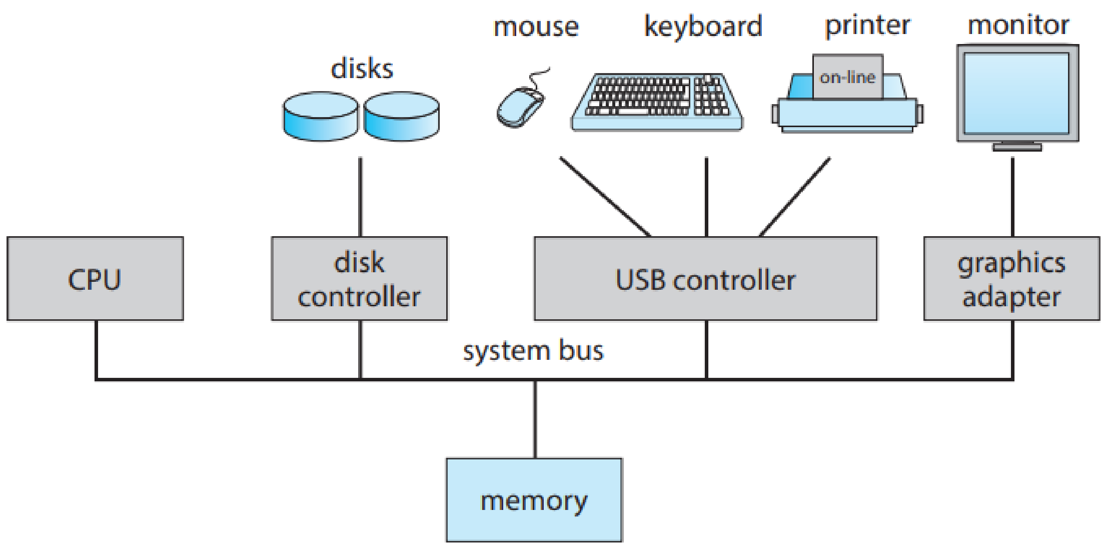

Figure 1.2 A typical PC computer system.