Question #1

1. What is an operating system? What are the main purposes of an operating system?

Operating system is a kind of all time running software in computer system to control other programs and allocate resources and it can organize and manage hardware and software to user reasonably and effectively, which also help user use computers and programmer coding friendly. Sometimes, it is also well module.

The purpose of operating system is to execute user programs and use hardware effectively to solve programs easier for user and developer.

1. Define the essential properties of the following:

Batch: scheduling jobs in memory and execute continuously and after user sending jobs then it has to wait the responds. CPU may idle long time for low speed of I/O.

Time sharing: time-sharing system has time slice to loop serving. It can improve the interaction between users and computers by proper slice to let users don’t realize the system is also serving others.

Dedicated: it is an un-sharable resource.

Real time: in a rigid range of time to response requests.

Virtualization: real physical objects can map to some logical objects from time or space.

DMA: direct access memory, large data can be packed as blocks passing from buffer storage to main memory block by block per time without CPU.

Interrupt Timeline:

Daisy chain: some devices linked by plugs in sequence ending to an port on computer.

1. Under which circumstances it is better using a time-sharing system rather than a PC or single-user workstation?

When user do not need so much good UI and APPs, using time-sharing system can use less operations.

Question #2

1. What is the performance advantage in having devices synchronization by means of device interrupts, rather than by polling (i. e. , device driver keeps on polling the device to see if a specific event has occurred)? Under which circumstances can polling be advantageous over interrupts?

Synchronization is like A and B may use some same resource and only one can be executed at same time. For interrupts, the waiting process can be executed. Polling is like CPU is loop searching the process, if I/O storage the process into main memory then it can be executed but may waste of lot of time in CPU cycling. So if a situation where resources have already been loaded in main memory and the amount of program is a lot then the polling is more efficient than interrupts because in this case, there will have a tons of interrupts and every time, system need to store messages which cost more expansive. Plus, if the devices are super fast or their have a bunch of them we can use polling because interrupts cost more.

1. Is it possible to use a DMA controller if the system does not support interrupts? Explain why.

No. DMA also needs to use interrupts to implement synchronization. DMA was suit to transfer big data to main memory not a lot of bytes for hardware structure.

Question #3

1. If a user program needs to perform I/O, it needs to trap the OS via a system call that transfers control to the kernel. The kernel performs I/O on behalf of the user program. However, system calls have added overheads, which can slow down the entire system. In that case, why not let user processes perform I/O directly, without going through the kernel?

Firstly, kernel can improve protection and security. If there is no kernel, user may make some mistake changing time unconsciously, which result in big problem of computer. Secondly, kernel can release the pressure of users to use without complex coding. Thirdly, if a kernel have already been designed and implemented, the kernel can be ported and help others.

1. Consider a computer running in user mode. It will switch to kernel mode whenever an interrupt or trap occurs, jumping to the address determined from the interrupt vector.
2. A smart, but malicious, user took advantage of a certain serious loophole in the computer’s protection mechanism, by which he could make run his own user program in the kernel mode! This can cause disastrous effects. What could have he possibly done to achieve this? What disastrous effects could it cause?

Enter Kernel

He can change the interrupt vector in kernel pointing to his malicious program like looping without ending. He should get the privilege to control interrupt vector pointing address and modify it to his own program, which means he actually controls what the computer executing to some extend.

1. Suggest a remedy for the loophole.

Use privileged instructions in hardware that only can be executed in kernel mode, if someone in user mode wants to change interrupt vector, the hardware should deal with this problem because cracker can’t change hardware.

Question #4

Assume you are given the responsibility to design two OS systems, a Network Operating System and a Distributed Operating System. Indicate the primary differences between these two systems. Additionally, you need to indicate if there are any possible common routines between these systems? If yes, indicate some of these routines. If no, explain why common routines between these two particular systems do not make sense.

The primary differences are Distributed Operating System is on different computers to deal with something like auto scheduling, dispatching jobs which can improve the computing ability and increase security (like block chain is based on this system, every operation can be traced and is invisible to every computer in the system), while Network Operating System is just based on network and obeys OSI/RM and easy to share resources and communicate to others.

Yes, operating systems need to handle some basic common things. Interrupt service routine, CPU scheduling routine, security routine, memory-management routine…

Question #5

How can we make MS-DOS operating system a layered structure with minimal effort?

Based on the hardware, we divide the system to User Mode and Kernel Mode. Because user want to get content of files so The upper layer of Kernel is file system, for file system, we need to provide communication layer for files so communication layer should be put into the next of file system. The memory management user will not control it even for protection. So memory management should be designed at second nearest position to hardware. The nearest position to hardware should be process management because we need it to start our operating system. Then the operating system should have device management between memory management and communication layer. This is the kernel mode. For user mode, applications can be accessed.

Question #6

Suppose that you have a costumed-designed board that has memory mapped I/O CPU. What is the procedure to add a new I/O hardware to this board and access it?

Device send CPU a interrupt signal and CPU execute what command it is running and make sure where exactly new device existing and send a signal back. CPU sets to kernel mode and save important information on stack for later executing. CPU checked the map and get the address and execute from that address. Then load it to memory.