Introduction

1. What are the two main functions of an operating system?

Ans; An operating system must provide the users with an extended machine, and it must manage the I/O devices and other system resources. To some extent, these are different functions.

1. In Section 1.4, nine different types of operating systems are described. Give a list of applications for each of these systems (one per operating systems type).

Ans: Obviously, there are a lot of possible answers. Here are some. Mainframe operating system: Claims processing in an insurance company. Server operating system: Speech-to-text conversion service for Siri. Multiprocessor operating system: Video editing and rendering. Personal computer operating system: Word processing application. Handheld computer operating system: Context-aware recommendation system. Embedded operating system: Programming a DVD recorder for recording TV. Sensor-node operating system: Monitoring temperature in a wilderness area. Real-time operating system: Air traffic control system. Smart-card operating system: Electronic payment.

5. On early computers, every byte of data read or written was handled by the CPU (i.e., there was no DMA). What implications does this have for multiprogramming?

Ans: The prime reason for multiprogramming is to give the CPU something to do while waiting for I/O to complete. If there is no DMA, the CPU is fully occupied doing I/O, so there is nothing to be gained (at least in terms of CPU utilization) by multiprogramming. No matter how much I/O a program does, the CPU will be 100% busy. This of course assumes the major delay is the wait while data are copied. A CPU could do other work if the I/O were slow for other reasons (arriving on a serial line, for instance).

6. Instructions related to accessing I/O devices are typically privileged instructions, that is, they can be executed in kernel mode but not in user mode. Give a reason why these instructions are privileged.

Ans: Access to I/O devices (e.g., a printer) is typically restricted for different users. Some users may be allowed to print as many pages as they like, some users may not be allowed to print at all, while some users may be limited to printing only a certain number of pages. These restrictions are set by system administrators based on some policies. Such policies need to be enforced so that userlevel programs cannot interfere with them.

9. There are several design goals in building an operating system, for example, resource utilization, timeliness, robustness, and so on. Give an example of two design goals that may contradict one another.

Ans: Consider fairness and real time. Fairness requires that each process be allocated its resources in a fair way, with no process getting more than its fair share. On the other hand, real time requires that resources be allocated based on the times when different processes must complete their execution. A realtime process may get a disproportionate share of the resources.

10. What is the difference between kernel and user mode? Explain how having two distinct modes aids in designing an operating system.

Ans: Most modern CPUs provide two modes of execution: kernel mode and user mode. The CPU can execute every instruction in its instruction set and use ev ery feature of the hardware when executing in kernel mode. However, it can execute only a subset of instructions and use only subset of features when executing in the user mode. Having two modes allows designers to run user programs in user mode and thus deny them access to critical instructions.

16. When a user program makes a system call to read or write a disk file, it provides an indication of which file it wants, a pointer to the data buffer, and the count. Control is then transferred to the operating system, which calls the appropriate driver. Suppose that the driver starts the disk and terminates until an interrupt occurs. In the case of reading from the disk, obviously the caller will have to be blocked (because there are no data for it). What about the case of writing to the disk? Need the caller be blocked awaiting completion of the disk transfer?

Ans: Maybe. If the caller gets control back and immediately overwrites the data, when the write finally occurs, the wrong data will be written. However, if the driver first copies the data to a private buffer before returning, then the caller can be allowed to continue immediately. Another possibility is to allow the caller to continue and give it a signal when the buffer may be reused, but this is tricky and error prone.

17. What is a trap instruction? Explain its use in operating systems.

Ans: A trap instruction switches the execution mode of a CPU from the user mode to the kernel mode. This instruction allows a user program to invoke functions in the operating system kernel.

18. Why is the process table needed in a timesharing system? Is it also needed in personal computer systems running UNIX or Windows with a single user?

Ans: The process table is needed to store the state of a process that is currently suspended, either ready or blocked. Modern personal computer systems have dozens of processes running even when the user is doing nothing and no programs are open. They are checking for updates, loading email, and many other things, On a UNIX system, use the ps -a command to see them. On a Windows system, use the task manager.

19. Is there any reason why you might want to mount a file system on a nonempty directory? If so, what is it?

Ans: Mounting a file system makes any files already in the mount-point directory inaccessible, so mount points are normally empty. Howev er, a system administrator might want to copy some of the most important files normally located in the mounted directory to the mount point so they could be found in their normal path in an emergency when the mounted device was being repaired.

27. Modern operating systems decouple a process address space from the machine’s physical memory. List two advantages of this design.

Ans: This allows an executable program to be loaded in different parts of the machine’s memory in different runs. Also, it enables program size to exceed the size of the machine’s memory

28. To a programmer, a system call looks like any other call to a library procedure. Is it important that a programmer know which library procedures result in system calls? Under what circumstances and why?

Ans: As far as program logic is concerned, it does not matter whether a call to a library procedure results in a system call. But if performance is an issue, if a task can be accomplished without a system call the program will run faster. Every system call involves overhead time in switching from the user context to the kernel context. Furthermore, on a multiuser system the operating system may schedule another process to run when a system call completes, further slowing the progress in real time of a calling process.

31. Explain how separation of policy and mechanism aids in building microkernel-based operating systems.

Ans: Separation of policy and mechanism allows OS designers to implement a small number of basic primitives in the kernel. These primitives are simplified, because they are not dependent of any specific policy. They can then be used to implement more complex mechanisms and policies at the user level.

32. Virtual machines have become very popular for a variety of reasons. Nevertheless, they have some downsides. Name one.

Ans: The virtualization layer introduces increased memory usage and processor overhead as well as increased performance overhead.

Processes and Threads

18. What is the biggest advantage of implementing threads in user space? What is the biggest disadvantage?

Ans: . The biggest advantage is the efficiency. No traps to the kernel are needed to switch threads. The biggest disadvantage is that if one thread blocks, the entire process blocks.

26. In Sec. 2.3.4, a situation with a high-priority process, H, and a low-priority process, L, was described, which led to H looping forever. Does the same problem occur if roundrobin scheduling is used instead of priority scheduling? Discuss.

Ans: With round-robin scheduling it works. Sooner or later L will run, and eventually it will leave its critical region. The point is, with priority scheduling, L never gets to run at all; with round robin, it gets a normal time slice periodically, so it has the chance to leave its critical region.

29. The producer-consumer problem can be extended to a system with multiple producers and consumers that write (or read) to (from) one shared buffer. Assume that each producer and consumer runs in its own thread. Will the solution presented in Fig. 2-28, using semaphores, work for this system?

ans: Yes, it will work as is. At a given time instant, only one producer (consumer) can add (remove) an item to (from) the buffer.

36. A fast-food restaurant has four kinds of employees: (1) order takers, who take customers’ orders; (2) cooks, who prepare the food; (3) packaging specialists, who stuff the food into bags; and (4) cashiers, who give the bags to customers and take their money. Each employee can be regarded as a communicating sequential process. What form of interprocess communication do they use? Relate this model to processes in UNIX.

Ans: The employees communicate by passing messages: orders, food, and bags in this case. In UNIX terms, the four processes are connected by pipes.

File Systems

8. A simple operating system supports only a single directory but allows it to have arbitrarily many files with arbitrarily long file names. Can something approximating a hierarchical file system be simulated? How?

Ans: Use file names such as /usr/ast/file. While it looks like a hierarchical path name, it is really just a single name containing embedded slashes.

11. Contiguous allocation of files leads to disk fragmentation, as mentioned in the text, because some space in the last disk block will be wasted in files whose length is not an integral number of blocks. Is this internal fragmentation or external fragmentation? Make an analogy with something discussed in the previous chapter.

Ans: Since the wasted storage is between the allocation units (files), not inside them, this is external fragmentation. It is precisely analogous to the external fragmentation of main memory that occurs with a swapping system or a system using pure segmentation.

17. For a given class, the student records are stored in a file. The records are randomly accessed and updated. Assume that each student’s record is of fixed size. Which of the three allocation schemes (contiguous, linked and table/indexed) will be most appropriate?

Ans: For random access, table/indexed and contiguous will be both appropriate, while linked allocation is not as it typically requires multiple disk reads for a given record.

27. Oliver Owl’s night job at the university computing center is to change the tapes used for overnight data backups. While waiting for each tape to complete, he works on writing his thesis that proves Shakespeare’s plays were written by extraterrestrial visitors. His text processor runs on the system being backed up since that is the only one they have. Is there a problem with this arrangement?

Ans: Ollie’s thesis may not be backed up as reliably as he might wish. A backup program may pass over a file that is currently open for writing, as the state of the data in such a file may be indeterminate

Memory Management

5. What is the difference between a physical address and a virtual address?

Ans: Real memory uses physical addresses. These are the numbers that the memory chips react to on the bus. Virtual addresses are the logical addresses that refer to a process’ address space. Thus a machine with a 32-bit word can generate virtual addresses up to 4 GB regardless of whether the machine has more or less memory than 4 GB

9. What kind of hardware support is needed for a paged virtual memory to work?

Ans: There needs to be an MMU that can remap virtual pages to physical pages. Also, when a page not currently mapped is referenced, there needs to be a trap to the operating system so it can fetch the page.

17. Suppose that a machine has 38-bit virtual addresses and 32-bit physical addresses. (a) What is the main advantage of a multilevel page table over a single-level one? (b) With a two-level page table, 16-KB pages, and 4-byte entries, how many bits should be allocated for the top-level page table field and how many for the next level page table field? Explain.

Ans: Consider,

(a) A multilevel page table reduces the number of actual pages of the page table that need to be in memory because of its hierarchic structure. In fact, in a program with lots of instruction and data locality, we only need the toplevel page table (one page), one instruction page, and one data page.

(b) Allocate 12 bits for each of the three page fields. The offset field requires 14 bits to address 16 KB. That leaves 24 bits for the page fields. Since each entry is 4 bytes, one page can hold 212 page table entries and therefore requires 12 bits to index one page. So allocating 12 bits for each of the page fields will address all 238 bytes

45. Explain the difference between internal fragmentation and external fragmentation. Which one occurs in paging systems? Which one occurs in systems using pure segmentation?

Ans: Internal fragmentation occurs when the last allocation unit is not full. External fragmentation occurs when space is wasted between two allocation units. In a paging system, the wasted space in the last page is lost to internal fragmentation. In a pure segmentation system, some space is invariably lost between the segments. This is due to external fragmentation.

48. Can you think of any situations where supporting virtual memory would be a bad idea, and what would be gained by not having to support virtual memory? Explain.

Ans: General virtual memory support is not needed when the memory requirements of all applications are well known and controlled. Some examples are smart cards, special-purpose processors (e.g., network processors), and embedded processors. In these situations, we should always consider the possibility of using more real memory. If the operating system did not have to support virtual memory, the code would be much simpler and smaller. On the other hand, some ideas from virtual memory may still be profitably exploited, although with different design requirements. For example, program/thread isolation might be paging to flash memory.

“Describe the process in figure 3-29 in detail”

Diagram

Description automatically generated

# Ch.5 Input/Output

## Problem 3

Figure 5-3(b) shows one way of having memory-mapped I/O even in the presence of separate buses for memory and I/O devices, namely, to first try the memory bus and if that fails try the I/O bus. A clever computer science student has thought of an improvement on this idea: try both in parallel, to speed up the process of accessing I/O devices. What do you think of this idea?

Diagram

Description automatically generated

Ans: It is not a good idea. The memory bus is surely faster than the I/O bus, otherwise why bother with it? Consider what happens with a normal memory request. The memory bus finishes first, but the I/O bus is still busy. If the CPU waits until the I/O bus finishes, it has reduced memory performance to that of the I/O bus. If it just tries the memory bus for the second reference, it will fail if this one is an I/O device reference. If there were some ways to instantaneously abort the previous I/O bus reference to try the second one, the improvement might work, but there is never such an option. All in all, it is a bad idea

## Problem 7

One mode that some DMA controllers use is to have the device controller send the word to the DMA controller, which then issues a second bus request to write to memory. How can this mode be used to perform memory to memory copy? Discuss any advantage or disadvantage of using this method instead of using the CPU to perform memory to memory copy

Ans: Memory to memory copy can be performed by first issuing a read command that will transfer the word from memory to DMA controller and then issuing a write to memory to transfer the word from the DMA controller to a different address in memory. This method has the advantage that the CPU can do other useful work in parallel. The disadvantage is that this memory to memory copy is likely to be slow since DMA controller is much slower than CPU and the data transfer takes place over system bus as opposed to the dedicated CPU-memory bus.

## Problem 10

In Fig. 5-9(b), the interrupt is not acknowledged until after the next character has been output to the printer. Could it have equally well been acknowledged right at the start of the interrupt service procedure? If so, give one reason for doing it at the end, as in the text. If not, why not?

Graphical user interface, text

Description automatically generated

Ans: It could have been done at the start. A reason for doing it at the end is that the code of the interrupt service procedure is very short. By first outputting another character and then acknowledging the interrupt, if another interrupt happens immediately, the printer will be working during the interrupt, making it print slightly faster. A disadvantage of this approach is slightly longer dead time when other interrupts may be disabled

## Problem 13

Explain how an OS can facilitate installation of a new device without any need for recompiling the OS.

Ans: UNIX does it as follows. There is a table indexed by device number, with each table entry being a C struct containing pointers to the functions for opening, closing, reading, and writing and a few other things from the device. To install a new device, a new entry has to be made in this table and the pointers filled in, often to the newly loaded device driver

## Problem 14

In which of the four I/O software layers is each of the following done. (a) Computing the track, sector, and head for a disk read. (b) Writing commands to the device registers. (c) Checking to see if the user is permitted to use the device. (d) Converting binary integers to ASCII for printing.

Ans: (a) Device driver. (b) Device driver. (c) Device-independent software. (d) User-level software

## Problem 16

Why are output files for the printer normally spooled on disk before being printed?

Ans: If the printer were assigned as soon as the output appeared, a process could tie up the printer by printing a few characters and then going to sleep for a week.

## Problem 20

RAID level 3 is able to correct single-bit errors using only one parity drive. What is the point of RAID level 2? After all, it also can only correct one error and takes more drives to do so.

Ans: RAID level 2 can not only recover from crashed drives, but also from undetected transient errors. If one drive delivers a single bad bit, RAID level 2 will correct this, but RAID level 3 will not.

## Problem 32

A slight modification of the elevator algorithm for scheduling disk requests is to always scan in the same direction. In what respect is this modified algorithm better than the elevator algorithm?

Ans: In the worst case, a read/write request is not serviced for almost two full disk scans in the elevator algorithm, while it is at most one full disk scan in the modified algorithm

# Ch.8 Multiprocessors

## Problem 2

What happens if three CPUs in a multiprocessor attempt to access exactly the same word of memory at exactly the same instant?

Ans: Depending on how CPUs are connected to memory, one of them gets through first, for example, seizes the bus first. It completes its memory operation, then another one happens, etc. It is not predictable which one goes first, but if the system has been designed for sequential consistency, it should not matter.

## Problem 4

Suppose that the wire between switch 2A and switch 3B in the omega network of Fig. 8-5 breaks. Who is cut off from whom?

Ans: CPUs 000, 010, 100, and 110 are cut off from memories 010 and 011.

## Problem 14

Affinity scheduling reduces cache misses. Does it also reduce TLB misses? What about page faults?

Ans: Affinity scheduling has to do with putting the right thread on the right CPU. Doing so might well reduce TLB misses since these are kept inside each CPU. On the other hand, it has no effect on page faults, since if a page is in memory for one CPU, it is in memory for all CPUs

## Problem 24

Some multicomputer allow running processes to be migrated from one node to another. Is it sufficient to stop a process, freeze its memory image, and just ship that off to a different node? Name two hard problems that have to be solved to make this work.

Ans: The table of open files is kept in the kernel, so if a process has open files, when it is unfrozen and tries to use one of its files, the new kernel does not know about them. A second problem is the signal mask, which is also stored on the original kernel. A third problem is that if an alarm is pending, it will go off on the wrong machine. In general, the kernel is full of bits and pieces of information about the process, and they have to be successfully migrated as well.

## Problem 30

Migrating virtual machines may be easier than migrating processes, but migration can still be difficult. What problems can arise when migrating a virtual machine?

Ans: Physical I/O devices still present problems because they do not migrate with the virtual machine, yet their registers may hold state that is critical to the proper functioning of the system. Think of read or write operations to devices (e.g., the disk) that have been issued but have not yet completed. Network I/O is particularly difficult because other machines will continue to send packets to the hypervisor, unaware that the virtual machine has moved. Even if packets can be redirected to the new hypervisor, the virtual machine will be unresponsive during the migration period, which can be long because the entire virtual machine, including the guest operating system and all processes executing on it, must be moved to the new machine. As a result packets can experience large delays or even packet loss if the device/hypervisor buffers overflow

## Problem 31

When a browser fetches a Web page, it first makes a TCP connection to get the text on the page (in the HTML language). Then it closes the connection and examines the page. If there are figures or icons, it then makes a separate TCP connection to fetch each one. Suggest two alternative designs to improve performance here.

Ans: One way would be for the Web server to package the entire page, including all the images, in a big zip file and send the whole thing the first time so that only one connection is needed. A second way would be to use a connectionless protocol like UDP. This would eliminate the connection overhead but would require servers and browsers to do their own error control.

## Problem 33

When multiple processes need access to data, in what way is object-based access better than shared memory?

Ans : Shared memory works with whole pages. This can lead to false sharing, in which access to unrelated variables that happen to lie on the same page causes thrashing. Putting each variable on a separate page is wasteful. Object-based access eliminates these problems and allows a finer grain of sharing.

# Ch. 7 Virtualization

## Problem 1

Give a reason why a data center might be interested in virtualization.

Ans: There are numerous reasons, among them consolidating servers to save hardware investment cost, rack space, and electrical power, and make management of thousands of servers easier.

## Problem 2

Give a reason why a company might be interested in running a hypervisor on a machine that has been in use for a while.

Ans: If the hardware configuration was upgraded, virtualization could hide this and allow old software to continue working.

## Problem 3

Give a reason why a software developer might use virtualization on a desktop machine being used for development

Ans: There are various reasons. A key one is to have many platforms such as Windows 7, Windows 8, Linux, FreeBSD, OS X, etc. available on a single desktop machine to test the software being developed. Also, rebooting a virtual machine after a crash induced by a software bug is much faster

## Problem 4

Give a reason why an individual at home might be interested in virtualization.

Ans: After upgrading to a new computer and operating system, the person might want to run some software that he had on the old one. Virtualization makes it possible to run the old system and new one on the same computer, thus preserving the old software.

## Problem 5

Why do you think virtualization took so long to become popular? After all, the key paper was written in 1974 and IBM mainframes had the necessary hardware and software throughout the 1970s and beyond.

Ans: Very few programmers had access to an IBM mainframe. Starting on the 1980s, the Intel x86 series dominated computing and it was not virtualizable. While binary translation could solve that problem, that idea was not thought of until the late 1990s.

## Problem 8

What is the difference between full virtualization and paravirtualization? Which do you think is harder to do? Explain your answer.

Ans: Full virtualization means emulating the hardware exactly so every operating system running on the virtual machine behaves exactly as it would on the bare metal. Paravirtualization consists of changing the operating system so it does not do anything that is hard to virtualize. Full virtualization in the absence of hardware support is complicated on any architecture that is complex, like the x86. It is easier on RISC machines. If virtualization hardware is present, full virtualization is not so difficult. So, which is harder probably depends on whether hardware support is available. If it is, then Para virtualizing an operating system is probably more work. If there is no hardware support, it may be easier to change the operating system to be more friendly. If there are many operating systems that must be Para virtualized, that could be more work.

## Problem 9

Does it make sense to paravirtualize an operating system if the source code is available? What if it is not?

Ans: Yes, of course. Linux has been paravirtualized precisely because the source code is available. Windows has been paravirtualized by Microsoft (which has the source code) but has not released any paravirtualized versions.

## Problem 10

Consider a type 1 hypervisor that can support up to n virtual machines at the same time. PCs can have a maximum of four disk primary partitions. Can n be larger than 4? If so, where can the data be stored?

Ans: Virtual machines have nothing to do with disk partitions. The hypervisor can take a disk partition and divide it up into sub partitions and give each virtual machine one of them. In principle, there can be hundreds. It can either statically partition the disk into n pieces or do this on demand. In hosted virtual machines, it is common to use files on the host to store disk images of the guest.

## Problem 11

Briefly explain the concept of process-level virtualization.

Ans: An application or process is virtualized during runtime, by using a virtualization layer between the application and the OS. This layer executes the application’s instructions, modifying them as required prior to execution. The application is transparent to the presence of the underlying layer. Windows Emulator (WINE) is an example, where Microsoft Windows binary executables can be executed on another operating system such as Linux. This is done using onthe-fly mapping of Windows API calls to POSIX calls.

## Problem 12

Why do type 2 hypervisors exist? After all, there is nothing they can do that type 1 hypervisors cannot do and the type 1 hypervisors are generally more efficient as well.

Ans: Type 1 hypervisors generally require changing the boot procedure of the computer to load the hypervisor first, then create virtual machines, and then install operating systems in them. At data centers run by expert system administrators, this is not a problem, but for most ordinary users, doing this is far too complicated. Type 2 hypervisors were invented to make installing a hypervisor no more difficult than installing an application program, something that users frequently do. Also, by using a host operating system to service local peripherals, it was not necessary for the hypervisor to have drivers for all of them since it could use the ones inside the host OS.

## Problem 13

Is virtualization of any use to type 2 hypervisors?

Ans: Yes. When a guest OS does I/O, for example, the virtualization hardware catches it and gets control to the type 2 hypervisor, which then figures out what to do. Usually this will involve making a request to the host OS to perform the I/O, but not having to worry about trapping the I/O instruction definitely simplifies matters for the hypervisor.

## Problem 28

Give one advantage of cloud computing over running your programs locally. Give one disadvantage as well.

Ans: Using cloud services means you do not have to set up and maintain a computing infrastructure. You may also be able to outsource making backups. Furthermore, if your computing needs change rapidly, you can add or remove machines easily. On the downside, the cloud provider could easily steal your confidential data, and the promised expandability might be illusory if you need extra capacity just at the moment Walmart or some other big customer decides to grab 10,000 machines. Also, the bandwidth between you and the cloud might be an issue. It is likely to be far less than the local bandwidth, so if a lot of data needs to move between you and the cloud, that could be an issue. Also, if you are doing real-time work, the bandwidth between you and the cloud could vary wildly from moment to moment, causing trouble.

## Problem 29

Give an example of IAAS, PAAS, and SAAS.

Ans: Obviously there are many, but a provider offering empty virtual x86 machines would be offering IAAS. A provider offering Windows 8 or Linux machines would be offering PAAS. A provider offering a word-processing program, such as Microsoft Word, running in the cloud would be offering software as a service.

## Problem 30

Why is virtual machine migration important? Under what circumstances might it be useful?

Ans: Suppose many virtual machines were started up on a single server. Initially, all of them did about the same amount of work and required the same resources and the situation was fine. Then suddenly, one of them began using massive resources (CPU, memory, etc.) disturbing all the other virtual machines. This might be a good time to migrate it to a dedicated server of its own

## Problem 31

Migrating virtual machines may be easier than migrating processes, but migration can still be difficult. What problems can arise when migrating a virtual machine?

Ans: Physical I/O devices still present problems because they do not migrate with the virtual machine, yet their registers may hold state that is critical to the proper functioning of the system. Think of read or write operations to devices (e.g., the disk) that have been issued but have not yet completed. Network I/O is particularly difficult because other machines will continue to send packets to the hypervisor, unaware that the virtual machine has moved. Even if packets can be redirected to the new hypervisor, the virtual machine will be unresponsive during the migration period, which can be long because the entire virtual machine, including the guest operating system and all processes executing on it, must be moved to the new machine. As a result, packets can experience large delays or even packet loss if the device/hypervisor buffers overflow.

## Problem 32

Why is migration of virtual machines from one machine to another easier than migrating processes from one machine to another?

Ans: In order to migrate a specific process, process state information must be stored and then transferred, including open files, alarms, signal handlers, etc. Errors may creep in during the state capture task leading to potentially incorrect, incomplete or inconsistent state information. In the case of VM migration, the entire memory and disk images are moved to the new system, which is easier.

## Problem 33

What is the difference between live migration and the other kind (dead migration?)?

Ans: Standard (dead) migration consists of stopping the virtual machine and saving its memory image as a file. The file is then transported to the destination, installed in a virtual machine, and restarted. Doing so causes the application to stop for a little while during transport. In many circumstances having the application stop is undesirable. With live migration, the pages of the virtual machine are moved while it is running. After they all arrive at the destination, a check is made to see if any of them have changed since being migrated. If so, they are copied again. This process is repeated until all the pages at the destination are up to date. Working this way (live migration) means applications can be moved with no downtime.

# Ch.10 Linux

## Problem 1

Explain how writing UNIX in C made it easier to port it to new machines.

Ans: Since assembly language is specific to each machine, a port of UNIX to a new machine required rewriting the entire code in the new machine’s assembly language. On the other hand, once UNIX was written in C, only a small part of the OS (e.g., device drivers for I/O devices) had to be rewritten.

## Problem 2

The POSIX interface defines a set of library procedures. Explain why POSIX standardizes library procedures instead of the system-call interface.

Ans: System call interface is tightly coupled to the OS kernel. Standardizing the system call interface would have put severe restrictions (less flexibility) on the design of the OS kernel. It would also make UNIX less portable.

## Problem 3

Linux depends on gcc compiler to be ported to new architectures. Describe one advantage and one disadvantage of this dependency.

Ans: This allows Linux to use special capabilities (like language extensions) of the gcc compiler that range from providing shortcuts and simplifications to providing the compiler with hints for optimization. The main disadvantage is that there are other popular, feature-rich C compilers like LLVM that cannot be used to compile Linux. If at some future time LLVM or some other compiler becomes better than gcc in all ways, Linux will not be able to use it. This could become a serious problem.

## Problem 7

Why does Linux distinguish between standard output and standard error, when both default to the terminal?

Ans: They are separate, so standard output can be redirected without affecting standard error. In a pipeline, standard output may go to another process, but standard error still writes on the terminal.

## Problem 16

Why do you think the designers of Linux made it impossible for a process to send a signal to another process that is not in its process group?

Ans: Malicious users could wreak havoc with the system if they could send signals to arbitrary unrelated processes. Nothing would stop a user from writing a program consisting of a loop that sent a signal to the process with PID i for all i from 1 to the maximum PID. Many of these processes would be unprepared for the signal and would be killed by it. If you want to kill off your own processes, that is all right, but killing off your neighbor’s processes is not acceptable.

## Problem 18

In general, do you think daemons have higher or lower priority than interactive processes? Why?

Ans: Generally, daemons run in the background doing things like printing and sending e-mail. Since people are not usually sitting on the edge of their chairs waiting for them to finish, they are given low priority, soaking up excess CPU time not needed by interactive processes.

## Problem 20

In every process’ entry in the task structure, the PID of the parent is stored. Why?

Ans: When the process exits, the parent will be given the exit status of its child. The PID is needed to be able to identify the parent so the exit status can be transferred to the correct process

## Problem 25

When booting Linux (or most other operating systems for that matter), the bootstrap loader in sector 0 of the disk first loads a boot program which then loads the operating system. Why is this extra step necessary? Surely it would be simpler to have the bootstrap loader in sector 0 just load the operating system directly.

Ans: The program loaded from block 0 is a maximum of 512 bytes long, so it can not be very complicated. Loading the operating system requires understanding the file system-layout in order to find and load the operating system. Different operating systems have very different file systems; it is asking too much to expect a 512-byte program to sort all this out. Instead, the block 0 loader just fetches another loader from a fixed location on the disk partition. This program can be much longer and system specific so that it can find and load the OS.

# Ch.11 Windows

## Problem 2

A mouse can have one, two, or three buttons. All three types are in use. Does the HAL hide this difference from the rest of the operating system? Why or why not?

Ans: The HAL is simple and straightforward. Including the mouse, the disk, and all the other device drivers in it would make it unwieldy and destroy its function as a thin layer that hides certain basic hardware differences of the computer itself, but not the I/O devices.

## Problem 7

An alternative to using DLLs is to statically link each program with precisely those library procedures it actually calls, no more and no less. If this scheme were to be introduced, would it make more sense on client machines or on server machines?

Ans: It would make more sense on servers. Client machines have fewer concurrent processes. Shared libraries make sense only if there are multiple processes sharing them. Otherwise, it is more efficient to statically link the libraries and accept duplication. The advantage of static linking is that only those procedures that are actually needed are loaded. With DLLs there may be procedures in memory that no one is using

## Problem 12

When initializing a global variable in a multithreaded program, a common programming error is to allow a race condition where the variable can be initialized twice. Why could this be a problem? Windows provides the InitOnceExecuteOnce API to prevent such races. How might it be implemented?

Ans: The variable might be a pointer to a dynamically allocated data structure, where the initialization and allocation of the structure was the responsibility of the first thread that wanted to use it. If two threads attempt to initialize the variable at nearly the same time, two different structures might be allocated and one of threads might use the wrong instance. **InitOnceExecuteOnce** uses an extra variable, say to record the state of each separate initialization. Valid states are: uninitialized, initializing, initialized. The thread that will actually do the initialization records atomically sets foo to initializing. When initialization is complete it atomically sets foo to initialized and calls **WakeByAddressAll**. Atomically setting and examining a variable can be performed either by using a lock or a hardware instruction like compare&swap. If a thread finds that foo is set to initialized, then it skips the initialization. If it finds foo set to initializing, it calls **WaitOnAddress** to wait for initialization to complete.

## Problem 13

Name three reasons why a desktop process might be terminated. What additional reason might cause a process running a modern application to be terminated?

Ans: (a) The last thread exits. (b) A thread executes ExitProcess. (c) Another process with a handle to this one kills it. (Modern apps) The OS decided to terminate it to reclaim room in the swap file or because the application was being serviced.

## Problem 14

Modern applications must save their state to disk every time the user switches away from the application. This seems inefficient, as users may switch back to an application many times and the application simply resumes running. Why does the operating system require applications to save their state so often rather than just giving them a chance at the point the application is actually going to be terminated?

Ans: The operating system terminates modern applications mainly when the system is low on memory or is being rebooted. If applications were to attempt to run in the former case, there might not be enough resources for them to successfully save their state. And in the latter case they might indefinitely delay the shutdown of Windows, as often happens in desktop Windows. Though users might switch between applications often, the frequency of those switches is at the human scale of seconds or minutes. For a well-written application, the few milliseconds required to save state does not have much impact. Additionally, writing applications this way creates a better user experience were the application to crash, since the last state save is available as a checkpoint.

## Problem 17

In Windows, the current priority is always greater than or equal to the base priority. Are there any circumstances in which it would make sense to have the current priority be lower than the base priority? If so, give an example. If not, why not?

Ans: Having your priority lowered below the base priority could be used as a punishment for using excessive CPU time or other resources.

## Problem 34

Suppose that you wanted to build Windows Lite. Which of the fields of Fig. 11-45 could be removed without weakening the security of the system?

Ans: All except the user SID could be removed without affecting the strength of the security.

# Ch.12 Design

## Problem 8

Operating systems often do naming at two different levels: external and internal. What are the differences between these names with respect to (a) Length? (b) Uniqueness? (c) Hierarchies?

Ans: External names can be as long as needed and of variable length. Internal names are generally 32 or 64 bits and always fixed length. External names need not be unique. Two names can point to the same object, for example, links in the UNIX file system. Internal names must be unique. External names may be hierarchical. Internal names are generally indices into tables and thus form a flat namespace

## Problem 9

One way to handle tables whose size is not known in advance is to make them fixed, but when one fills up, to replace it with a bigger one, copy the old entries over to the new one, then release the old one. What are the advantages and disadvantages of making the new one 2× the size of the original one, as compared to making it only 1.5× as big?

Ans: If the new table is 2× as big as the old one, it will not fill up quickly, reducing the number of times an upgraded table will be needed. On the other hand, so much space may not be needed, so it may waste memory. This is a classic time vs. space trade-off

## Problem 12

Indirection is a way of making an algorithm more flexible. Does it have any disadvantages, and if so, what are they?

Ans: Yes. It makes the code slower. Also, more code means more bugs.

## Problem 13

Can reentrant procedures have private static global variables? Discuss your answer.

Ans: Not easily. Multiple invocations at the same time could interfere with one another. It might be possible if the static data were guarded by a mutex, but that would mean that a call to a simple procedure might unexpectedly block.

## Problem 16

In Fig. 12-8, we saw how GIF files use 8-bit values to index into a color palette. The same idea can be used with a 16-bit-wide color palette. Under what circumstances, if any, might a 24-bit color palette be a good idea?

Ans: No circumstances. The ‘‘compressed’’ color value would be as big as the original, and in addition, a huge color palette could be needed. It makes no sense at all.

## Problem 17

One disadvantage of GIF is that the image must include the color palette, which increases the file size. What is the minimum image size for which an 8-bit-wide color palette breaks even? Now repeat this question for a 16-bit-wide color palette

Ans: The 8-bit-wide color palette contains 256 entries of 3 bytes each for a total of 768 bytes. The saving per pixel is 2 bytes. Thus, with more than 384 pixels, GIF wins. A 16-bit-wide color palette contains 65,536 entries of 3 bytes each, for 196,608 bytes. The saving here is 1 byte per pixel. Thus, with more than 196,608 pixels, the 16-bit compression wins. Assuming a 4:3 ratio, the break-even point is an image of 512 × 384 pixels. For VGA (640 × 480), 16-bit color requires less data than true 24-bit color.

Problem 23

Using Brooks’ figure of 1000 lines of code per programmer per year, make an estimate of the amount of money it took to produce Windows 8. Assume that a programmer costs $100,000 per year (including overhead, such as computers, office space, secretarial support, and management overhead). Do you believe this answer? If not, what might be wrong with it?

Ans: If a programmer can produce 1000 lines of code for a cost of $100,000, a line of code costs $100. Windows 8 consists of 50–100 million lines of code, which comes to $5–10 billion. That seems like an awful lot. Probably Microsoft has managed to improve programmer productivity using better tools so that a programmer can produce thousands of lines of code per year. Also, large pieces of Windows 8 were taken unmodified from Windows 7, so the amount of new code in Windows 8 is only a fraction of its total size. On the other hand, Microsoft’s annual revenue is around $70 billion, so spending billions of dollars on Windows 8 is possible.

## Problem 25

Name some features of a conventional operating system that are not needed in an embedded system used inside an appliance.

Ans: Suppose that memory costs $10 per GB (check against current prices). Then a low-end machine with a 100-GB disk needs $1000 worth of RAM for its disk. If the rest of the PC is $500, the total cost comes to $1500. This is too expensive for the low-end market.