The heart of any operating system is the set of system calls that it can handle. These tell what the operating system really does. For UNIX, we have looked at four groups of system calls. The first group of system calls relates to process creation and termination. The second group is for reading and writing files. The third group is for directory management. The fourth group contains miscellaneous calls.

Operating systems can be structured in several ways. The most common ones are as a monolithic system, a hierarchy of layers, microkernel, client-server, virtual machine, or exokernel.

PROBLEMS

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

2. 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).

3. What is the difference between timesharing and multiprogramming systems?

4. To use cache memory, main memory is divided into cache lines, typically 32 or 64 bytes long. An entire cache line is cached at once. What is the advantage of caching an entire line instead of a single byte or word at a time?

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?

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.

7. The family-of-computers idea was introduced in the 1960s with the IBM System/360 mainframes. Is this idea now dead as a doornail or does it live on?

8. One reason GUIs were initially slow to be adopted was the cost of the hardware needed to support them. How much video RAM is needed to support a 25-line × 80-row character monochrome text screen? How much for a 1200 × 900-pixel 24-bit color bitmap? What was the cost of this RAM at 1980 prices ($5/KB)? How much is it now?

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.

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

11. A 255-GB disk has 65,536 cylinders with 255 sectors per track and 512 bytes per sector. How many platters and heads does this disk have? Assuming an average cylinder seek time of 11 ms, average rotational delay of 7 msec and reading rate of 100 MB/sec, calculate the average time it will take to read 400 KB from one sector.

12. Which of the following instructions should be allowed only in kernel mode?

(a) Disable all interrupts.
(b) Read the time-of-day clock.
(c) Set the time-of-day clock.
(d) Change the memory map.

13. Consider a system that has two CPUs, each CPU having two threads (hyperthreading). Suppose three programs, *P0*, *P1*, and *P2*, are started with run times of 5, 10 and 20 msec, respectively. How long will it take to complete the execution of these programs? Assume that all three programs are 100% CPU bound, do not block during execution, and do not change CPUs once assigned.

14. A computer has a pipeline with four stages. Each stage takes the same time to do its work, namely, 1 nsec. How many instructions per second can this machine execute?

15. Consider a computer system that has cache memory, main memory (RAM) and disk, and an operating system that uses virtual memory. It takes 1 nsec to access a word from the cache, 10 nsec to access a word from the RAM, and 10 ms to access a word from the disk. If the cache hit rate is 95% and main memory hit rate (after a cache miss) is 99%, what is the average time to access a word?

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?

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

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?

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

20. For each of the following system calls, give a condition that causes it to fail: fork, exec, and unlink.

21. What type of multiplexing (time, space, or both) can be used for sharing the following resources: CPU, memory, disk, network card, printer, keyboard, and display?

22. Can the

```
count = write(fd, buffer, nbytes);
```

call return any value in *count* other than *nbytes*? If so, why?

23. A file whose file descriptor is *fd* contains the following sequence of bytes: 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5. The following system calls are made:

```
lseek(fd, 3, SEEK_SET);
read(fd, &buffer, 4);
```

where the lseek call makes a seek to byte 3 of the file. What does *buffer* contain after the read has completed?

24. Suppose that a 10-MB file is stored on a disk on the same track (track 50) in consecutive sectors. The disk arm is currently situated over track number 100. How long will it take to retrieve this file from the disk? Assume that it takes about 1 ms to move the arm from one cylinder to the next and about 5 ms for the sector where the beginning of the file is stored to rotate under the head. Also, assume that reading occurs at a rate of 200 MB/s.

25. What is the essential difference between a block special file and a character special file?

26. In the example given in Fig. 1-17, the library procedure is called *read* and the system call itself is called read. Is it essential that both of these have the same name? If not, which one is more important?

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

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?

29. Figure 1-23 shows that a number of UNIX system calls have no Win32 API equivalents. For each of the calls listed as having no Win32 equivalent, what are the consequences for a programmer of converting a UNIX program to run under Windows?

30. A portable operating system is one that can be ported from one system architecture to another without any modification. Explain why it is infeasible to build an operating system that is completely portable. Describe two high-level layers that you will have in designing an operating system that is highly portable.

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

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

33. Here are some questions for practicing unit conversions:

(a) How long is a nanoyear in seconds?
(b) Micrometers are often called microns. How long is a megamicron?
(c) How many bytes are there in a 1-PB memory?
(d) The mass of the earth is 6000 yottagrams. What is that in kilograms?

34. Write a shell that is similar to Fig. 1-19 but contains enough code that it actually works so you can test it. You might also add some features such as redirection of input and output, pipes, and background jobs.

35. If you have a personal UNIX-like system (Linux, MINIX 3, FreeBSD, etc.) available that you can safely crash and reboot, write a shell script that attempts to create an unlimited number of child processes and observe what happens. Before running the experiment, type sync to the shell to flush the file system buffers to disk to avoid

ruining the file system. You can also do the experiment safely in a virtual machine. **Note**: Do not try this on a shared system without first getting permission from the system administrator. The consequences will be instantly obvious so you are likely to be caught and sanctions may follow.

36. Examine and try to interpret the contents of a UNIX-like or Windows directory with a tool like the UNIX *od* program. (*Hint*: How you do this will depend upon what the OS allows. One trick that may work is to create a directory on a USB stick with one operating system and then read the raw device data using a different operating system that allows such access.)