same event, mutex, or semaphore, rather than having one of them create the object and then make duplicate handles for the others, although the latter approach is certainly an option as well.

6.5 SUMMARY

The operating system can be regarded as an interpreter for certain architectural features not found at the ISA level. Chief among these are virtual memory, virtual I/O instructions, and facilities for parallel processing.

Virtual memory is an architectural feature whose purpose is to allow programs to use more address space than the machine has physical memory, or to provide a consistent and flexible mechanism for memory protection and sharing. It can be implemented as pure paging, pure segmentation, or a combination of the two. In pure paging, the address space is broken up into equal-sized virtual pages. Some of these are mapped onto physical page frames. Others are not mapped. A reference to a mapped page is translated by the MMU into the correct physical address. A reference to an unmapped page causes a page fault. Both the Pentium 4 and the UltraSPARC III have MMUs that support virtual memory and paging.

The most important I/O abstraction present at this level is the file. A file consists of a sequence of bytes or logical records that can be read and written without knowledge of how disks, tapes, and other I/O devices work. Files can be accessed sequentially, randomly by record number, or randomly by key. Directories can be used to group files together. Files can be stored in consecutive sectors or scattered around the disk. In the latter case, normal on hard disks, data structures are needed to locate all the blocks of a file. Free disk storage can be kept track of using a list or a bit map.

Parallel processing is often supported and is implemented by simulating multiple processors by timesharing a single CPU. Uncontrolled interaction between processes can lead to race conditions. To solve this problem, synchronization primitives are introduced, of which semaphores are a simple example. Using semaphores, producer-consumer problems can be solved simply and elegantly.

Two examples of sophisticated operating systems are UNIX and XP. Both support paging and memory-mapped files. They also both support hierarchical file systems, with files consisting of byte sequences. Finally, both support processes and threads and provide ways to synchronize them.

PROBLEMS

1. Why does an operating system interpret only some of the level 3 instructions, whereas a microprogram interprets all the ISA level instructions?

2. A machine has a 32-bit byte-addressable virtual address space. The page size is 4 KB. How many pages of virtual address space exist?

3. Is it necessary to have the page size be a power of 2? Could a page of size, say, 4000 bytes be implemented in theory? If so, would it be practical?

4. A virtual memory has a page size of 1024 words, eight virtual pages, and four physical page frames. The page table is as follows:

Virtual page	Page frame
0	3
1	1
2	not in main memory
3	not in main memory
4	2
5	not in main memory
6	0
7	not in main memory

a. Make a list of all virtual addresses that will cause page faults.
b. What are the physical addresses for 0, 3728, 1023, 1024, 1025, 7800, and 4096?

5. A computer has 16 pages of virtual address space but only four page frames. Initially, the memory is empty. A program references the virtual pages in the order

0, 7, 2, 7, 5, 8, 9, 2, 4

a. Which references cause a page fault with LRU?
b. Which references cause a page fault with FIFO?

6. In Sec. 6.1.4 an algorithm was presented for implementing a FIFO page replacement strategy. Devise a more efficient one. *Hint:* It is possible to update the counter in the newly-loaded page, leaving all the others alone.

7. In the paged systems discussed in the text, the page fault handler was part of the ISA level and thus was not present in any OSM level program's address space. In reality, the page fault handler also occupies pages, and might, under some circumstances (e.g., FIFO page replacement policy), itself be removed. What would happen if the page fault handler were not present when a page fault occurred? How could this be fixed?

8. Not all computers have a hardware bit that is automatically set when a page is written to. Nevertheless, it is useful to keep track of which pages have been modified, to avoid having to assume worst case and write all pages back to the disk after use. Assuming that each page has hardware bits to separately enable access for reading, writing, and execution, how can the operating system keep track of which pages are clean and which are dirty?

9. A segmented memory has paged segments. Each virtual address has a 2-bit segment number, a 2-bit page number, and an 11-bit offset within the page. The main memory contains 32 KB, divided up into 2-KB pages. Each segment is either read-only, read/execute, read/write, or read/write/execute. The page tables and protection are as follows:

Segment 0		Segment 1		Segment 2	Segment 3	
Read only		Read/execute		Read/write/execute	Read/write	
Virtual page	Page frame	Virtual page	Page frame		Virtual page	Page frame
0	9	0	On disk	Page table not in main memory	0	14
1	3	1	0		1	1
2	On disk	2	15		2	6
3	12	3	8		3	On disk

For each of the following accesses to virtual memory, tell what physical address is computed. If a fault occurs, tell which kind.

Access	*Segment*	*Page*	*Offset within page*
1. fetch data	0	1	1
2. fetch data	1	1	10
3. fetch data	3	3	2047
4. store data	0	1	4
5. store data	3	1	2
6. store data	3	0	14
7. branch to it	1	3	100
8. fetch data	0	2	50
9. fetch data	2	0	5
10. branch to it	3	0	60

10. Some computers allow I/O directly to user space. For example, a program could start up a disk transfer to a buffer inside a user process. Does this cause any problems if compaction is used to implement the virtual memory? Discuss.

11. Operating systems that allow memory-mapped files always require files to be mapped at page boundaries. For example, with 4-KB pages, a file can be mapped in starting at virtual address 4096, but not starting at virtual address 5000. Why?

12. When a segment register is loaded on the Pentium 4, the corresponding descriptor is fetched and loaded into an invisible part of the segment register. Why do you think the Intel designers decided to do this?

13. A program on the Pentium 4 references local segment 10 with offset 8000. The BASE field of LDT segment 10 contains 10000. Which page directory entry does the Pentium 4 use? What is the page number? What is the offset?

14. Discuss some possible algorithms for removing segments in an unpaged, segmented memory.

15. Compare internal fragmentation to external fragmentation. What can be done to alleviate each?

16. Supermarkets are constantly faced with a problem similar to page replacement in virtual memory systems. They have a fixed amount of shelf space to display an ever-increasing number of products. If an important new product comes along, say, 100%

efficient dog food, some existing product must be dropped from the inventory to make room for it. The obvious replacement algorithms are LRU and FIFO. Which of these would you prefer?

17. In some ways, caching and paging are very similar. In both cases there are two levels of memory (the cache and main memory in the former and main memory and disk in the latter). In this chapter we looked at some of the arguments in favor of large disk pages and small disk pages. Do the same arguments hold for cache line sizes?

18. Why do many file systems require that a file be explicitly opened with an open system call before being read?

19. Compare the bit-map and hole-list methods for keeping track of free space on a disk with 800 cylinders, each one having 5 tracks of 32 sectors. How many holes would it take before the hole list would be larger than the bit map? Assume that the allocation unit is the sector and that a hole requires a 32-bit table entry.

20. To be able to make some predictions of disk performance, it is useful to have a model of storage allocation. Suppose that the disk is viewed as a linear address space of $N \gg 1$ sectors, consisting of a run of data blocks, then a hole, then another run of data blocks, and so on. If empirical measurements show that the probability distributions for data and hole lengths are the same, with the chance of either being i sectors as 2^{-i}, what is the expected number of holes on the disk?

21. On a certain computer, a program can create as many files as it needs, and all files may grow dynamically during execution without giving the operating system any advance information about their ultimate size. Do you think that files are stored in consecutive sectors? Explain.

22. Studies of different file systems have shown that more than half the files are a few KB or smaller, with the vast majority of files less than something like 8 KB. On the other hand, the largest 10 percent of all files usually occupies about 95 percent of the entire disk space in use. From this data, what conclusion can you draw about disk block size?

23. Consider the following method by which an operating system might implement semaphore instructions. Whenever the CPU is about to do an up or down on a semaphore (an integer variable in memory), it first sets the CPU priority or mask bits in such a way as to disable all interrupts. Then it fetches the semaphore, modifies it, and branches accordingly. Finally, it enables interrupts again. Does this method work if

a. There is a single CPU that switches between processes every 100 msec?
b. Two CPUs share a common memory in which the semaphore is located?

24. The Nevercrash Operating System Company has been receiving complaints from some of its customers about its latest release, which includes semaphore operations. They feel it is immoral for processes to block (they call it "sleeping on the job"). Since it is company policy to give the customers what they want, it has been proposed to add a third operation, peek, to supplement up and down. peek simply examines the semaphore without changing it or blocking the process. In this way, programs that feel it is immoral to block can first inspect the semaphore to see if it is safe to do a down. Will this idea work if three or more processes use the semaphore? If two processes use the semaphore?