Page by: Duong Anh Duc

Review question: Process

Page by: Duong Anh Duc

▶ Summary

Consider a hypothetical 32-bit microprocessor that has 32-bit instructions composed of two fields. The first byte contains the opcode and the remainder an immediate operand or an operand address.

- What is the maximum directly addressable memory capacity (in bytes)?

2(32-8) = 16,777,216 bytes = 16 MB

- Discuss the impact on the system speed if the microprocessor data bus has
- a 32-bit local address bus and a 16-bit local data bus.

Instruction and data transfers would take three bus cycles each - one for the address and two for the data.

- a 16-bit local address bus and a 16-bit local data bus.

Instruction and data transfers would take four bus cycles each - two for the address and two for the data.

- How many bits are needed for the program counter and the instruction register?

24 bits for the PC (24-bit addresses), 32 bits for the IR (32-bit addresses)

In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access to main memory. Why?

If a processor is held up in attempting to read or write memory, usually no damage occurs except a slight loss of time. However, a DMA transfer may be to or from a device that is receiving or sending data in a stream (e.g., disk or network), and cannot be stopped. Thus, if the DMA module is held up (denied continuing access to main memory), data will be lost.

Why does a batch system need memory protection? Why does a multiprogrammed batch system need memory management and scheduling? Why do both types of systems need interrupts?

A batch system needs memory protection so that programs can't modify the monitor.

A multiprogrammed batch system needs memory management so that multiple jobs can be kept in memory at the same time. Memory management will determine which portion of memory each job can use. The system also needs a scheduling algorithm to determine which job can run at any given time.

Both types of systems need interrupts so that the operating system can regain control of the CPU. A batch system needs this to time-out a long job and a multiprogrammed system needs this to share the processor among the active processes.

Review question: Process

Page by Duong Anh Duc

What is the purpose of system calls, and how do system calls relate to the operating system and to the concept of dual-mode (kernel mode and user mode) operation?

With a time sharing system, the primary concern is turnaround time. A round-robin scheduler would given every process a chance to run on the CPU for a short time, and reduce the average turnaround time. If the scheduler instead let one job run until completion, then the first job would have a short turnaround time, but later ones would have to wait for a long time.

In a batch system, the primary concern is throughput. In this case, the time spent switching between jobs is wasted, so a more efficient scheduling algorithm would be first-come, first-served, and let each job run on the processor as long as it wants.

In IBM's mainframe operating system, OS/390, one of the major modules in the kernel is the System Resource Manager (SRM). This module is responsible for the allocation of resources among address spaces (processes). The SRM gives OS/390 a degree of sophistication unique among operating systems. No other mainframe operating system, and certainly no other type of operating system, can match the functions performed by SRM. The concept of resource includes processor, real memory and I/O channels. SRM accumulates statistics pertaining to utilization of processor, channel and various key data structures. Its purpose is to provide optimum performance based on performance monitoring and analysis. The installation sets forth various performance objectives, and these serve as guidance to the SRM, which dynamically modifies installation and job performance characteristics based on system utilization. In turn, the SRM provides reports that enable the trained operator to refine the configuration and parameter settings to improve user service.

This problem concerns one example of SRM activity. Real memory is divided into equal-sized blocks called frames, of which there may be many thousands. Each frame can hold a block of virtual memory referred to as a page. SRM receives control approximately 20 times per second and inspects each and every page frame. If the page has not been referenced or changed, a counter is incremented by 1. Over time, SRM averages these numbers to determine the average number of seconds that a page frame in the system goes untouched. What might be the purpose of this and what action might SRM take?

The system operator can review this quantity to determine the degree of "stress" on the system. By reducing the number of active jobs allowed on the system, this average can be kept high. A typical guideline is that this average should be kept above 2 minutes [IBM86]. This may seem like a lot, but it isn't.

What main advantage does a system with virtual memory have over a system without it? How does virtual memory help with memory protection?

With virtual memory, a process image does not need to be entirely in physical memory for it to run. In addition, it is not restricted to the size of physical memory. The process can address as much memory as it wants, and the operating system brings blocks from secondary storage into physical memory as they are needed. Virtual memory also helps with memory

protection because one process cannot address memory that belongs to another process. All addresses are mapped by the operating system into its own address space.

Page by: Duong Anh Duc

What is the difference between a symmetric multiprocessing system and a distributed operating system?

In a symmetric multiprocessing system, a group of processors with the same capabilities share memory and are located within the same computer system. The existence of multiple processors is transparent to the user. In a distributed operating system, the group of processors each addresses its own memory, are located in different computers connected by a network, and can have differing capabilities. The operating system allows the user to treat the collection of systems as one computer.

What shortcoming does the two-state process model have? How does the Blocked state fix this problem?

In the two-state process model, if a process is interrupted for I/O, it is placed in a queue with other processes that are ready to run. When the operating system wants to find a new process to run on the CPU, it must search through the queue for a process that is ready. Using a model with a Blocked state means that there are two separate queues: one for ready processes and one for those that are waiting for I/O. When the operating system wants to find a process that is ready, it can simply take the first process on the ready queue.

What shortcoming does the five-state process model have? How does swapping solve this problem?

Assuming virtual memory is not being used, then only a limited number of processes can be active at any time. In the five-state process model, it is possible that all active processes are waiting on an I/O event and none of them are in the ready queue. In this case, the CPU will be idle.

Swapping can solve this problem by taking a process out of memory and writing it out to disk, then bringing in a new process that can run on the CPU. The five-state model is augmented with a new state called Suspended to indicate the process is swapped out.

What is contained in a process image? Some older operating systems require the entire process image to be stored in main memory while the process is executing. What problem does this create and how is it solved?

The process image contains user data (for program data and the stack), the user program, the system stack (for procedure calls), and the process control block (process identifiers, processor state, various types of control information).

If the entire image must be in main memory, then the image must be small and only a limited number of processes can be active at any time. This problem can be solved by using virtual memory, so that only a portion of the process image (the portion currently being used) needs to be in memory at any given time.

It states that UNIX is unsuitable for real-time applications because a process

It states that UNIX is unsuitable for real-time applications because a process executing in kernel mode may not be preempted. Elaborate.

Because there are circumstances under which a process may not be preempted (i.e., it is executing in kernel mode), it is impossible for the operating system to respond rapidly to real-time requirements.

Use the ps program to list all the processes running on a Linux machine (including system processes and processes for all users). List one process for each unique state that is listed (i.e. one per unique entry in the state column) and describe what state this process is in. See the manual page for help with ps. Next, create five new processes in quick succession and leave them running. Using ps, list their process identifiers. Is there any relation between the identifiers they are assigned?

You can use ps ax to list all the processes on a Linux machine. Here is a list of some of the processes I found:

PID TTY STAT TIME COMMAND

1 ? S 0:04 init [3]

2 ? SN 0:00 [ksoftirqd/0]

3 ? S< 0:02 [events/0]

4 ? S< 0:00 [khelper]

9 ? S< 0:00 [kthread]

18 ? S< 0:06 [kacpid]

118 ? S< 0:00 [kblockd/0]

176 ? S 0:00 [pdflush]

179 ? S< 0:00 [aio/0]

178 ? S 0:01 [kswapd0]

771 ? S 0:04 [kseriod]

829 ? S< 0:00 [ata/0]

843 ? S< 0:00 [reiserfs/0]

983 ? Ss 0:00 /sbin/devfsd /dev

5192 ? S 0:00 [khubd]

6328 ? Ss 0:00 metalog [MASTER]

6332 ? S 0:00 metalog [KERNEL]

6366 ? Ss 0:00 /usr/sbin/acpid -c /etc/acpi/events

6991 ? Ss 0:00 /usr/sbin/cupsd

7326 ? Ss 0:00 /usr/sbin/fcron

7422 ? Ss 0:00 /usr/sbin/speedfreqd -P /var/run/speedfreq.pid -p pow

There are many more. The TTY column indicates whether the process is attached to a termanl, and STAT gives the state of the process. Most processes are sleeping (S), one has been given a low priority (N), and others have been given a high

priority (<). Notice that typically important system processes have a high priority.

Review question: Process

If you create five processes in quick succession, they will typically have sequential process identifiers.

Page by: Duong Anh Duc

How is a process switch different from a mode switch?

A process switch involves the operating system having to do a lot of work, whereas a mode switch is done completely in hardware. A mode switch involves saving the state of the current process (in hardware), changing to supervisor mode, then giving control to the operating system. The idea is that the operating system will execute a short handler, then return control to the processor. The processor will restore the state of the original process and continue.

With a process switch, the operating system must give control of the CPU to a different process, rather than resuming with the previous process that was running. Thus it must save the context of the current process in its PCB, update the PCB state and accounting information, move the PCB to a new queue, execute the scheduling algorithm to choose a new process, remove the PCB of that process from the ready queue, update its memory management structures, and restore its context into the processor.

In a threaded program, which resources are shared by the entire process and which are associated with each thread?

The process address space, structures used for interprocess communication, files, and I/O resources are associated with the entire process. Each thread gets its own execution state, context (registers), stack, and local variables.

Consider the following code:

```
for (i = 0; i < 10; i++)
for (j = 0; j < 10; j++)
a[i] = a[i] * j
```

- Give one example of the spatial locality in the code.

Spatial locality occurs when the array a is accessed sequentially.

- Give one example of the temporal locality in the code.

Temporal locality occurs when i is used repeatedly with the same value in the second loop.

Consider a memory system with the following parameters:

- Tc = 100 ns
- Tm = 1200 ns
- Cc = 0.01 cents/bit
- Cm = 0.001 cents/bit

What is the cost of 1 MByte of main memory?

Cost = Cm x 8 x 220 = $83.89

What is the cost of 1 Mbyte of main memory using cache memory technology?

Cost = Cc x 8 x 220 = $838.86

If the effective access time is 10% greater than the cache access time, what is the hit ratio H?

if the effective access time is 10% greater than the cache access time, what is the hit ratio H?

Review question: Process

1.1 x Tc = Tc + (1 - H)Tm110 = 100 + (1 - H) x 12001200 x H = 1190H = 1190/1200 = 99.2%

Page by: Duong Anh Duc

A computer has a cache, main memory, and a disk used for virtual memory. If a referenced word is in the cache, 20 ns are required to access it. If it is in main memory but not in the cache, 60 ns are needed to load it into the cache (this includes the time to originally check the cache), and then the reference is started again. If the word is not in main memory, 12 ms are required to fetch the word from disk, followed by 60 ns to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main memory hit ratio is 0.6. What is the average time in ns required to access a referenced word on this system?

There are three cases to consider:

Location of Word	Probability	Total Time for Access (ns)
In cache	0.9	20
Not in cache, but in main memory	(0.1)(0.6) = 0.06	60 + 20 = 80
Not in cache or main memory	(0.1)(0.4) = 0.04	12ms + 60 + 20 = 12,000,080

So the average access time would be:

Avg = (0.9)(20) + (0.06)(80) + (0.04)(12000080) = 480,026 ns

Suppose that we have a multi-programmed computer in which each job has identical characteristics. In one computation period, T, for a job, half the time is spent in I/O and the other half in processor activity. Each job runs for a total of N periods. Assume that simple round-robin scheduling is used, and that I/O operations can overlap with processor operation. Define the following quantities:

- Turnaround time = actual time to complete a job.
- Throughput = average number of jobs completed per time period T
- Processor utilization = percentage of time that the processor is active (waiting)

Compute these quantities for one, two, and four simultaneous jobs, assuming that the period T is distributed in each of the following ways:

I/O first half, processor second half

When there is one job, it can do I/O or run on the processor whenever it wants. So the quantities are:

- Turnaround Time = N*T
- Throughput = 1/N
- Processor Utilization = 50%

When there are two jobs, one starts right away and does I/O. When it switches to run on the CPU, the second can start its I/O. This delays the second job for 1/2*N, but otherwise they alternate between I/O and CPU. Assume the jobs are long, so

the extra 1/2 a cycle is insignificant. Then:

- Turnaround Time = N*T
- Throughput = 2/N
- Processor Utilization = 100%

Review question: Process

Page by: Duong Anh Duc

When there are 4 jobs, the CPU is round-robin among the four, as is the I/O. This means the jobs are interleaved as:

Job1: I/O CPU I/O CPU

Job2: I/O CPU I/O CPU

Job3: I/O CPU I/O CPU

Job4: I/O CPU I/O CPU

A job can execute for one cycle T, then it must wait for T before doing another cycle. Again assume the jobs are long so that any initial wait is insignificant. Then:

- Turnaround Time = (2N-1)*T
- Throughput = 2/N
- Processor Utilization = 100%
- I/O first and fourth quarters, processor second and third quarter

The answers for this part are the same as the first. This is easy to see for the case of 1 job and 2 jobs. When there are 4 jobs, the CPU is round-robin among the four, as is the I/O. This means the jobs are interleaved as:

Job1: I CP O I CP O

Job2: I CP O I CP O

Job3: I CP O I CP O

Job4: I CP O I CP O

An I/O-bound program is one that, if run alone, would spend more time waiting for I/O than using the processor. A processor-bound program is the opposite. Suppose a short-term scheduling algorithm favors those programs that have used little processor time in the recent past.

- Explain why this algorithm favors I/O-bound programs.

The algorithm favors I/O bound processes because these will be much less likely to have used the processor recently. As soon as they are done with an I/O burst, they will get to execute on the processor.

- Explain why this algorithm does not permanently deny processor time to processor-bound programs.

This algorithm does not permanently deny processor time to processor-bound programs, because once they have been prevented from using the processor for some time they will be favored by the scheduling algorithm.

Review question: Process

Page by: Duong Anh Duc