

CS 201 OPERATING SYSTEMS (FALL 2005)
SOLUTIONS TO ASSIGNMENT 1

(For reference only -- some questions may have several acceptable answers)

Q 1

1.7 (10 pts)

We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?

Answer:

Single-user systems should maximize use of the system for the user. A GUI might “waste” CPU cycles, but it optimizes the user’s interaction with the system. For example, when we use PCs or handheld computers, it’s appropriate for the operating system to forsake this principle and to “waste” resources. Because those operating systems are designed mostly for individual usability, for ease of use, with some attention paid to performance, and none paid to resource utilization.

2.4 (10 pts)

For what types of operations is DMA useful? Explain your answer.

Answer:

DMA is useful for transferring data between memory and devices if large volume of data is to be transferred, or the devices have small response times. Because after setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU. Only one interrupt is generated per block, rather than the one interrupt per byte (or word) generated for low-speed devices.

Alternatively, you may simply say:

DMA is useful for transferring large quantities of data between memory and devices. It eliminates the need for the CPU to be involved in the transfer, allowing the transfer to complete more quickly and the CPU to perform other tasks concurrently.

2.5 (10 pts)

Which of the following instructions should be privileged?

- a. Set value of timer.
- b. Read the clock.
- c. Clear memory.
- d. Turn off interrupts.
- e. Switch from user to monitor mode.

Answer: The following 4 instructions should be privileged:

- a. Set value of timer.
- c. Clear memory.
- d. Turn off interrupts.
- e. Switch from user to monitor mode.

2.10 (10 pts)

Writing an operating system that can operate without interference from malicious or undebugged user programs requires some hardware assistance. Name three hardware aids for writing an operating system, and describe how they could be used together to protect the operating system.

Answer:

A. Monitor/user mode. With the mode bit, we are able to distinguish between a task that is executed on behalf of the operating system, and one that is executed on behalf of the user. 0: monitor mode. 1: user mode.

B. Memory protection with base and limit registers. The base register holds the smallest legal physical memory address; the limit register contains the size of the range. This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other user's memory results in a trap to the monitor, which treats the attempt as a fatal error.

C. Timer. We can use the timer to prevent a user program from running too long, which would ensure that the operating system maintains control. Before turning over control to the user, the operating system ensures that the timer is set to interrupt. If the timer interrupts, control transfers automatically to the operating system.

D. Privileged instructions.

3.1 (10 pts)

What are the five major activities of an operating system in regard to process management?

Answer:

- _ The creation and deletion of both user and system processes
- _ The suspension and resumption of processes
- _ The provision of mechanisms for process synchronization
- _ The provision of mechanisms for process communication
- _ The provision of mechanisms for deadlock handling

4.6 (10 pts)

The correct producer-consumer algorithm in Section 4.4 allows only $n-1$ buffers to be full at any one time. Modify the algorithm to allow all buffers to be utilized fully.

Answer:

```
# define BUFFER_SIZE 10
typedef struct {
    ...
} item;
item buffer [BUFFER_SIZE];
int in=0;
int out=0;

Boolean isfull [BUFFER_SIZE];
isfull [0:BUFFER_SIZE-1] = 0; // initialize all flags to false
// The code for the producer process
```

```

While(1){
    ...
    produce an item in nextProduced
    ...
    While ( in == out && isfull [in] = 1); // full, do no-op
    buffer [in] = nextProduced;
    isfull [in] = 1;
    in = (in+1) % BUFFER_SIZE;
}

// The code for the consumer process
while(1){
    while ( in == out && isfull [out] == 0); //empty, do no-op
    nextConsumed = buffer [out];
    isfull [out] = 0;
    out = (out+1) % BUFFER_SIZE;
    ...
    consume the item in nextConsumed
    ...
}

```

Q 2 (10 pts)

Consider a floppy disk with 5 millisecond seek time, 1 millisecond rotational latency time, and 6 megabytes/second data transfer rate. How much time does it take to completely read a given track which stores 60 kilobytes of data starting at a given sector?

Answer:

seek time= 5 millisecond
rotational latency time=1 millisecond
data transfer time = 60 kilobytes / 6 megabytes/second
= $(60 \times 1024) / 6 \times 1024 \times 1024$
= 0.00977 second
= 0.00977 millisecond
total time = 5+1+9.77 = 15.77 millisecond (\approx 16 millisecond)

Q 3 (10 pts)

Identify and briefly describe the system calls performed during the execution of the following program segment in C.

Answer:

fopen (attempt to open the file named "sample.txt"),
printf (output to console),
fclose (file close),
exit (program terminations with status 0(normal termination) and status 1(termination with error))

Q 4 (10 pts)

Consider a batch system which runs at most one job at a time. Once it starts executing a job it runs the job through completion. Suppose that there are n jobs with execution times $1, 2, \dots, n$ all of which are ready to run at time zero. Consider that the system selects as the next job to run one which demands the largest computation time among the remaining jobs. Is this a good strategy to minimize the average overall response time for this case? Why? What would be an optimal strategy (no formal proof is necessary, but briefly explain why)? Compare the result obtained by the above strategy with the optimum achievable (no formal proof for the optimality is necessary, but include an expression for the optimum result)? (Note that since all jobs are ready at time zero the response time for each job is the same as its completion time, and we would like to minimize the total completion times divided by the number of jobs, n in this case).

Answer:

The given strategy is not a good one because the largest possible execution time is added to each process' response time, and thus, the average overall response time is

$$(n + (n+n-1) + (n+n-1+n-2) + \dots + (n+n-1+\dots+2+1)) / n.$$

An optimal strategy chooses a job with the smallest execution time next to run, which gives the overall average response time as

$$(1 + (1+2) + (1+2+3) + \dots + (1+2+\dots+n)) / n.$$

This is much smaller than the previous expression.