Instructor: Pawel Gburzynski

CMPUT 379, Fall 2000, Final Examination (A2)

Id number ....

Surname ....

Given names ....

This is an open book (and notes) exam. All questions have equal weights and must be answered in whatever space is available on this form. No additional sheets are allowed. Copying random excerpts from your class notes is pointless. It will (marginally) waste my time and (considerably) lower your mark.

Although all questions are multiple choice, you are required to explain your selections. Your explanations must be succinct and up to the point (one or two sentences each), yet they must demonstrate your understanding of the issue. Correct selections without or with incorrect explanations will result in zero marks.

The exam is marked out of 38 which is the percentage of its contribution to the final mark.

Question 1

Suppose that the CPU is running a program in the non-privileged (problem) state. Which one from the following statements is true?

1. The CPU will not switch to the supervisor state until the program suspends or terminates itself.
2. The CPU will not switch to the supervisor state until there is an interrupt or the program executes a supervisor call.
3. The CPU will not switch to the supervisor state until the CPU scheduler is run.
4. The CPU will not switch to the supervisor state until the program attempts to enter a critical section.

Question 2

You are on a system with variable partitions and memory protection based on fence registers. The minimum amount of context information on such a system includes:

1. All general purpose registers + instruction counter + PSW.
2. All of the above + the contents of fence registers.
3. All of the above + partition number.
4. All the above + the largest gap size.

Question 3

Imagine that you are designing a memory allocation scheme for a system with variable partitions and compaction. Virtual memory cannot be implemented on your machine, because there is no paging hardware, and machine instructions are not restartable. Every program is allowed to use two areas that in principle may grow dynamically: the heap and the stack. What is the most appropriate set of restrictions for such a system?

1. Every program must specify in advance two numbers, i.e., its maximum requirements for each of the two dynamic areas.
2. Every program must specify in advance one number - its maximum requirement for the stack, but need not specify the maximum requirement for the heap.
3. Every program must specify in advance one number - its maximum requirement for the heap, but need not specify the maximum requirement for the stack.
4. Every program must specify in advance one number - its maximum requirement for the combined size of the two areas.

Question 4

On a single-CPU system, the most basic synchronization tool is interrupt masking. Although not all interrupts can be masked, the unmaskable ones cause no problems because:

1. They are not asynchronous and cannot occur unless the CPU triggers them.
2. They are never serviced by the system, so nothing can go wrong if they occur.
3. They have a very low priority, and are only serviced if the system is not within a critical section.
4. They cannot cause rescheduling, so whoever is holding the critical section will not lose the CPU.

Question 5

A system with variable partitions, compaction, and swapping imposes a limit N on the number processes that can be simultaneously admitted to second level scheduling. Let M denote the size of physical memory available to processes and S be the size of the swap area in secondary storage. What is the minimum value of S guaranteeing that processes will never deadlock on memory allocation requests?

1. $(N - 1) * M$
2. $N * M$
3. $2N * M$
4. $(2N + 1) * M$

Question 6

Imagine that you are on a system with pure segmentation, like PDP 11/45. Which one of the following statements is true?

1. One process can have at most one down-extensible segment.
2. All processes sharing memory must have exactly the same access rights to that memory.
3. Fragmentation can be eliminated by copying portions of physical memory.
4. There is no way to avoid fragmentation, unless all programs are position independent.

Question 7

A program containing the following code:

```
double Array [HUGE][HUGE];
...
int i, j;
while ( ... ) {
   ...
   for (i = 0; i < HUGE; i++) {
     for (j = i+1; j < HUGE; j++) {
       Array [i][j] = Array [j][i];
     }
   }
   ...
}
```

was executed on a virtual memory system. The system started to run this program within very small physical memory, and the program thrashed heavily until the amount of physical memory allocated to the program (expressed in bytes) became slightly above:

1. HUGE
2. HUGE * sizeof (double)
3. HUGE * 2 * sizeof (double)
4. HUGE * HUGE * sizeof (double)

Select the smallest amount that made the biggest difference.

Question 8

Some concepts related to memory allocation are more difficult to implement with inverted than with straightforward page tables. For which one of the following concepts, inverted page tables make the biggest difference?

1. Working set.
2. Dynamic memory allocation.
3. Memory sharing.
4. Down-extensible segments.

Question 9

Suppose that the profile of jobs in a hypothetical high-performance computing center requires a combination of SJF and Round-Robin. The LEAST sensible way to combine these two scheduling policies is to:

1. Use "outer" Round-Robin between the two queues.
2. Use preemptive SJF with aging between the two queues.
3. Use fixed priority preemptive scheduling between the two queues, with higher priority given to SJF.
4. Use fixed priority preemptive scheduling between the two queues, with higher priority given to Round-Robin.

Question 10

Assuming that 1 block = 4 fragments, which one of the following statements about the UNIX filesystem with dual block size is FALSE?

1. A pointer block must be a block, i.e., it cannot be a fragment.
2. A pointer in a pointer block must point to a block rather than a fragment.
3. A single file may have no more than 3 fragments.
4. If the total amount of disk space allocated to a file is a multiple of block size, the file doesn't have any fragments.