CSCE 455/855 Distributed Operating Systems

Spring 2001
Steve Goddard

Homework 0, January 8

Evaluation of prerequisite knowledge

Due: 12:30pm January 12

The prerequisites for CSCE 455/855 are CSCE 451 and 420 (recommended). This purpose of this assignment is to evaluate your understanding of the prerequisite knowledge. You may consult any textbook or other resource to complete the assignment, but do the work on your own. This is not a group project.

1) (5 points) What is the purpose of system calls, and how do system calls differ from procedure calls?

2) (5 points) Define the difference between preemptive and non-preemptive scheduling. Explain why strictly non-preemptive scheduling is unlikely to be used in a general-purpose computer system.

3) (15 points) Consider a computer system with *n* processes sharing the processor in a round-robin fashion. If each context switch takes *s* seconds, what must be the quantum size *q* such that the overhead resulting from timesharing is minimized but, at the same time, each process is guaranteed to get its turn at the CPU at least every *t* seconds?

4) (15 points) Consider the following implementation of a general (counting) semaphore. This implementation assumes the existence of binary semaphore operations up_b and $down_b$ implemented with a test-and-set instruction (i.e., the binary semaphore operation $down_b$ busy waits if the test-and-set operation fails). Note that UP and DOWN semaphore operations were originally called V and P respectively by Dijkstra.

```
Procedure DOWN( S : semaphore):
   Down_b(mutex)
   S := S - 1
   If (S < 0) then
      up_b(mutex)
      down_b(delay)
   endif

   up_b(mutex)
end DOWN
```

```
procedure UP( S : semaphore):
   down_b(mutex)
   S := S + 1

   if (S ≤ 0) then
      up_b(delay)
   else
      up_b(mutex)
   endif
end UP
```

For each of the following scheduling policies, will the above code yield a correct implementation of a semaphore? (That is, assume a set of processes call `UP` and `DOWN` to coordinate their activities and that these processes are scheduled by one of the policies below. For each policy, explain the effect of the policy, if any, on the coordination of the processes.)

a) First-Come-First-Served

b) Shortest-Job-First (you may assume either a preemptive or non-preemptive version)

c) Priority (you may assign whatever priorities to the processes you wish)

d) Round-Robin

5) (15 points) Consider a system consisting of m resources of the same type, being shared by n processes. Resources can be requested and released by processes only one at a time. Show that the system is deadlock free if the following two conditions hold:

i) The maximum claim of each process is between 1 and m (inclusive) resources.

ii) The sum of all maximum claims is less than $m + n$.

6) (15 points) Consider a paged virtual memory system that has a page size of 512 bytes (2^9). Processes in this system can have a maximum virtual address space of 64K bytes (2^{16}). The system is currently configured with 8K bytes of physical memory.

a) How many frames in the physical address space?

b) How many pages in a virtual address space?

c) If each entry in the page table is 16 bits, what percentage of the physical memory does the page table (assuming the entire page table is stored in physical memory at all times) occupy?

d) A user process generates the virtual address 14,359 (11100000010111 in binary). Explain how the system establishes the corresponding physical address. Distinguish between software and hardware operations.

7) (15 points) On a computer that uses 1024 byte memory pages maintains the page table for each process in main memory. The overhead required for reading an entry in the page table is 300 ns. To reduce this overhead the computer has a TLB that caches 64 (virtual page, physical frame) mappings. A TLB lookup requires 60 ns. What TLB hit-rate is required to ensure an average virtual address translation time of 150 ns?

8) (15 points) Consider the concept of cylinder groups for file allocation.

a) Explain the concept of file allocation based on cylinder groups?

b) Are cylinder groups a good idea? Explain.

c) Compared to storing blocks of a file at random places on the disk, are cylinder groups better or worse for storing: (explain your answers)

- small files (*e.g.*, a simple e-mail message)
- medium size files (*e.g.*, the source code for a program)
- very large files (*e.g.*, an uncompressed, color image)