CISC 324, Winter 2017 -- Assignment 6

This assignment is due on Monday April 9 (by midnight)

Please hand in your assignment using the onQ dropbox

To access the dropbox, click on *assessments* -> *dropbox*. The dropbox is named "Assignment 6".

Readings

Monitors Course reader page 42 and 100-102 about Java monitors. General description of monitors in textbook section 5.8.

There are no assignment questions about monitors. Lab 5/6 gives you experience with Java monitors: the keyword *synchronized* and the methods wait(), notify() and notifyAll().

I do not emphasize monitor coding in this course because Java's monitors are quite limited, allowing only one condition variable per monitor. I use semaphores, not monitors, in final exam questions that ask you to write or analyze concurrent code. However, the final exam might contain a short-answer question that tests whether you know what a monitor is, and how it is used in lab 5/6. In summary: a monitor is a module of source code that encapsulates shared data. It contains declaration, initialization and access procedures for the shared data. The monitor automatically provides mutually-exclusive access to the shared data by enforcing that only one process at a time can be executing an access procedure.

Message passing Course reader pages 43-44, Textbook section 3.4.2.

Optional Textbook section 3.5 describes message passing in POSIX, Mach and Windows. I will not ask final exam questions about these particular systems.

Optional Asynchronous programming Course reader page 44 describes this alternative to concurrent programming. This is not tested on the final exam, but you might find it useful in programming you do in the future.

CPU scheduling Textbook sections 6.1-6.3. (Optional: scheduling in Windows section 6.7.2; priority inversion section 5.6.4)

Threads Textbook section 4.1. (Optional: sections 4.2 to 4.7 provide details about kernel threads, user threads, etc.)

Security and protection

Access lists and capabilities: textbook sections 14.1-14.5, 14.7.

Buffer overflow attack: textbook section 15.2.4 and course reader pages 64-70.

Operating system structure Course reader pages 71-74 and textbook sections 2.7, 16.1, 16.3

OS code can be organized into layers, or into a kernel and processes. Virtual machines.

Optional Textbook Chapter 18 on Linux and Chapter 19 on Windows 7 These chapters provide excellent review of course material. [You do not need to memorize any aspects of Linux or Windows 7 for the final exam.] I encourage you to skim these chapters, focusing on topics that interest you. My summary of Chapter 19 is in course reader pages 72-74 and I will discuss parts of chapters 18 and 19 in lecture at the end of term.

Message passing

1) Processes can use shared memory or message passing to communicate and synchronize.

(a) State two reasons why shared memory isn't available in all computer systems. Hint: one reason is geographic separation where distant computers are connected by a network. Another reason is security; see textbook section 16.3.

(b) I start up one process on a computer in Kingston and another process on a computer in Winnipeg. These two computers are connected by a network. Can I use semaphores to synchronize these processes? (Keep in mind that semaphores are shared global variables.)

(c) Does a *remote procedure call* use synchronous or asynchronous message passing? (See course reader page 44.)

CPU Scheduling

2) What is the difference between *turnaround time* and *throughput*? Describe one scenario in which throughput is improved without changing average turnaround time. Describe another scenario in which turnaround time is improved without changing throughput. The scenarios could include changes to the number of jobs presented to a computer system, changes to the number of CPUs in the system, and changes to the speed of the CPUs in the system.

3) Compute the average turnaround time and the throughput for each of the following four situations. All jobs are sharing the use of a single I/O device; jobs have to wait if the I/O device is in use.
a) Situation A: One job is submitted every minute. Each job takes 30 seconds of I/O time, and CPU use is negligible.
b) Situation B: Same as situation A, but a job is submitted once every 30 seconds.
c) Situation C: Same as situation A, but a job is submitted once every 15 seconds.
d) Situation D: Same as situation A, but we've substituted an I/O device that is twice as fast.

4)Here is information about five processes that are being executed. Only one CPU burst is described for each process.

Process Name	Arrival Time	CPU burst length
P1	0	3
P2	1	5
P3	3	2
P4	9	5
P5	12	5

Show the scheduling of these jobs under the following policies. Use a timeline format similar to the Gantt charts shown in Section 6.3 of the textbook. Compute the average turnaround time under each of these scheduling algorithms.

- FCFS (First-Come, First-Served).
- Round-robin with time quantum Q equal to 1. (You can put a newly-arriving process at the front or the end of the ready queue. Either policy is fine, just pick one policy and use it consistently.)
- Round-robin with time quantum equal to 4.
- SJF (Shortest-Job-First). Use the non-preemptive version of shortest-job-first (at the start of section 6.3.2), rather than the preemptive version (shortest-remaining-time-first, at the end of section 6.3.2).

Remember to compute the average turnaround time for the schedule resulting from each of these scheduling algorithms.

5a) Using round-robin scheduling, what is the advantage of choosing a short time quantum? What is the advantage of choosing a long time quantum?
(b) Describe how multi-level feedback queue scheduling is able to give priority to processes that have short CPU bursts, while at the same time ensuring that processes with long CPU bursts do not starve.

Threads

6) Which of the following are private to a thread, and which are shared with other threads?

The program counter	The stack	The code area	The data area (also called the *heap*; this contains global variables)

For help, refer to the first paragraph of textbook Section 4.1.

Security and Protection

7) Briefly describe how a buffer overflow can be used to force the victim computer to execute arbitrary code. See pages 61-67 of the course reader and textbook section 15.2.4. You don't have to understand all the details in the course reader. It's enough if you write a brief overview of how a buffer overflow attack works, describing how the stack is involved.

8 a) Briefly define what access lists and capabilities are.

b) Process P creates a file F and wants to give other processes the right to access this file. However, sometime in the future process P may want to selectively revoke the access rights, so that certain processes can no longer access file F. How could can revocation be provided using access lists? How can revocation be provided using capability-based protection?

c) Suppose we are interested in fine-grain protection where each procedure operates on a strict "need-to-know" basis. Which mechanism (access lists or capabilities) would you recommend? Why?

d) Why is it desirable to incorporate certain operating systems security functions directly into hardware or microcode? Briefly discuss the hardware support needed for capability-based protection systems: describe the mechanisms for distinguishing capabilities from other data (see textbook section 14.5.3).

Operating System Structure

9) Can parts of an operating system be implemented as processes? Can the kernel of an operating system be a process?

Memory management review problem: does not have to be handed in

(Review) When a process accesses a segmented memory, the OS/hardware must check that the segment offset is in range. Describe how information from the segment table is used to carry out this test. Also describe how the segment table can be used to store protection information.

You do not have to hand in an answer. Think about this and study the answer provided with the assignment 6 solution.