CPSC 457
OPERATING SYSTEMS

MIDTERM EXAM SOLUTION

Department of Computer Science
University of Calgary
Professor: Carey Williamson

October 29, 2008

This is a CLOSED BOOK exam. Textbooks, notes, laptops, calculators, personal digital assistants, cell phones, and Internet access are NOT allowed.

It is a 50 minute exam, with a total of 50 marks. There are 10 questions, and 8 pages (including this cover page). Please read each question carefully, and write your answers legibly in the space provided. You may do the questions in any order you wish, but please USE YOUR TIME WISELY.

When you are finished, please hand in your exam paper and sign out. Good luck!

Student Name: ____________________

Student ID: ____________________

Score: ________ / 50 = _______ %

Multiple Choice

Choose the best answer for each of the following 6 questions, for a total of 6 marks.

1 1. The philosophy behind the original Unix system was:

(a) simple solutions are preferable to complicated solutions

(b) a small general-purpose kernel can support interactive time-sharing systems

(c) a high-level language like C is preferable to low-level assembly language

(d) system source code should be available for research use

(e) advanced functionality can be built from small composable utilities

(f) **all of the above**

1 2. The system calls `chown()`, `chmod()`, and `umask()` are examples of operating system functionality for:

(a) file manipulation

(b) device manipulation

(c) process control

(d) information maintenance

(e) **protection and security**

(f) inter-process communication

1 3. The `ioctl()` system call is typically used to:

(a) convert I/O addresses from octal (base 8) to decimal (base 10)

(b) convert I/O addresses from decimal (base 10) to octal (base 8)

(c) both (a) and (b)

(d) **manipulate I/O devices**

(e) convert processes into threads

(f) enhance CPU scheduler performance

1 4. The threading model supported by the typical Linux kernel is:

(a) **one-to-one**

(b) one-to-many

(c) many-to-one

(d) many-to-many

(e) two-level

(f) all of the above

1 5. Among CPU scheduling policies, First Come First Serve (FCFS) is attractive because:

(a) **it is simple to implement**

(b) it is fair to all processes

(c) it minimizes the total waiting time in the system

(d) it minimizes the average waiting time in the system

(e) it minimizes the average response time in the system

(f) it minimizes the average turnaround time in the system

1 6. Among CPU scheduling policies, Shortest Remaining Processing Time (SRPT) is attractive because:

(a) it is simple to implement

(b) it is fair to all processes in the system

(c) **it minimizes the average waiting time in the system**

(d) it minimizes the average response time in the system

(e) it minimizes the number of context switches in the system

(f) it maximizes the number of context switches in the system

Operating System Concepts and Definitions

12 7. For each of the following pairs of terms, define each term, making sure to clarify the key difference(s) between the two terms.

(a) (2 marks) "application software" and "system software"
Both are types of software that can run on a computer system.
Application software: designed by users or 3rd party vendors for specific problem domains (e.g., gaming, graphics, spreadsheets, word processor)
System software: operating system itself, plus support utilities (e.g., text editor, compiler, shell, command-line utilities)
Application software typically runs ''on top of'' system software, but can make use of system software and OS services.

(b) (2 marks) "user mode" and "kernel mode"
These are types of execution modes in an operating system.
User mode: conventional operating mode for user apps, which have restricted privileges for what they can do on the system
Kernel mode: provides raw access to physical hardware for the OS; process has full privileges for what it can do on the system
System calls allow a user process to transition from user mode to kernel mode for use of specific OS servies.

(c) (2 marks) "single-core" and "multi-core"
These are two types of architectures for processor (CPU) design.
Single core: only 1 processor on a chip; supports only 1 thread of control
Multi-core: more than 1 processor on a chip, each with own registers, program counter, etc,; supports multiple concurrent threads of control
Multi-core provides hardware support for mutli-threaded applications.
Faster execution possible, but more complicated to design and coordinate.
(note the question is not about uniprocessors versus multi-processors)

(d) (2 marks) "text segment" and "data segment"
These are two different pieces of a typical process address space.
Text segment: stores the code that is being executed; usually immutable
Data segment: stores the statically allocated data (constants, arrays, global variables) for the process; usually read and modified by the process

(e) (2 marks) "short-term scheduler" and "long-term scheduler"
These are two different concepts of CPU scheduling in an OS.
Short-term: decides which in-memory ready process gets the CPU next
Long-term: decides which jobs go in-memory and on the ready queue
The decision-making time scales differ (milliseconds versus seconds).

(f) (2 marks) "waiting time" and "service time"
These are two contributors to job turnaround time in CPU scheduling.
Waiting time: amount of time spent waiting your turn for the resource (CPU)
Service time: actual time spent using the resource (CPU) during your turn(s)

CPU Scheduling

10 8. Suppose that the following jobs arrive as indicated for scheduling and execution on a single CPU.

Job	Arrival Time	Size (msec)	Priority
J_1	0	12	1 (Gold)
J_2	2	4	3 (Bronze)
J_3	5	2	1 (Gold)
J_4	8	10	3 (Bronze) <———(Note: typo fixed now)
J_5	10	6	2 (Silver) <———(Note: typo fixed now)

(a) (2 marks) Draw a Gantt chart showing FCFS scheduling for these jobs, and calculate the average job waiting time.

J_1	J_2	J_3	J_4	J_5

0 12 16 18 28 34

```
J1:  0   J2:  10   J3:  11   J4:  10   J5:  18     Avg:  49/5 = 9.8 ms
```

(b) (2 marks) Draw a Gantt chart showing non-preemptive SJF scheduling for these jobs, and calculate the average job waiting time.

J_1	J_3	J_2	J_5	J_4

0 12 14 18 24 34

```
J1:  0   J2:  12   J3:  7   J4:  16   J5:  8     Avg:  43/5 = 8.6 ms
```

(c) (2 marks) Draw a Gantt chart showing preemptive SJF (SRPT) scheduling for these jobs, and calculate the average job waiting time.

J_1	J_2	J_3	J_1	J_5	J_1	J_4

0 2 6 8 10 16 24 34

```
J1:  12   J2:  0   J3:  1   J4:  16   J5:  0     Avg:  29/5 = 5.8 ms
```

(d) (2 marks) Draw a Gantt chart showing RR (quantum = 4) scheduling for these jobs, and calculate the average job waiting time.

J_1	J_2	J_3	J_4	J_5	J_1	J_4	J_5	J_1	J_4

0 4 8 10 14 18 22 26 28 32 34

```
J1:  20   J2:  2   J3:  3   J4:  16   J5:  12     Avg:  53/5 = 10.6 ms
```

(e) (2 marks) Draw a Gantt chart showing (preemptive) PRIORITY scheduling for these jobs, and calculate the average job waiting time.

J_1	J_3	J_5	J_2	J_4

0 12 14 20 24 34

```
J1:  0   J2:  18   J3:  7   J4:  16   J5:  4     Avg:  45/5 = 9.0 ms
```

Operating System Utilities

10 9. The output on the next page is from a moderately busy Linux system. Use the output and your knowledge of Linux systems to answer the following questions:

(a) (1 mark) How many users are using the system in this example?
`5 users (although only 3 are distinct human users)`

(b) (1 mark) On average, how many processes are currently in the ready queue?
`2`

(c) (1 mark) Has the system load been increasing, decreasing, or staying about the same in the last 15 minutes?
`about the same`

(d) (1 mark) What type of Linux shell is the user (carey) running?
`the C shell (csh)`

(e) (1 mark) What is the default scheduling priority for user processes on this system?
`75`

(f) (1 mark) What is the PID of the most recently created process for user jsmith?
`26068`

(g) (1 mark) Among all the indicated processes on the system, which has consumed the most CPU time so far?
`6321 (running Xvnc)`

(h) (1 mark) Which user owns the process identified in (g)?
`bushay (1367)`

(i) (1 mark) How many different terminal windows does the user in (h) have running?
`4 pseudo-terminals (pts)`

(j) (1 mark) Among all the indicated processes on the system, which process has the largest "memory footprint"?
`6321 (running Xvnc)`

```
[carey@csl ~]$ w
 11:38:22 up 89 days, 52 min,  5 users,  load average: 2.01, 2.02, 2.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
jsmith   pts/1    s0106001b11691c1 10:00    1:09m  0.03s  0.00s ./a.out
jsmith   pts/2    s0106001b11691c1 10:13    1:10m  0.02s  0.00s ./a.out
carey    pts/3    csg              11:38    0.00s  0.01s  0.00s w
bushay   pts/9    agren            06Oct08 20:54m  0.03s  0.03s -bash
bushay   pts/13   agren            Sat13   20:37m  0.01s  0.01s -bash

[carey@csl ~]$ ps
  PID TTY          TIME CMD
28016 pts/3    00:00:00 csh
28052 pts/3    00:00:00 ps

[carey@csl ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   214 28016 28015  0  75   0 -  1416 rt_sig pts/3    00:00:00 csh
0 R   214 28053 28016  0  75   0 -  1103 -      pts/3    00:00:00 ps

[carey@csl ~]$ ps -l -u bushay
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1367  6008     1  0  75   0 -  3684 -      ?        00:00:00 Xvnc
0 S  1367  6012     1  0  75   0 -  1059 -      ?        00:00:00 vncconfig
0 S  1367  6013     1  0  75   0 -  2711 -      ?        00:00:00 xterm
0 S  1367  6014     1  0  75   0 -  1464 -      ?        00:00:00 twm
0 S  1367  6016  6013  0  85   0 -  1155 -      pts/7    00:00:00 bash
0 S  1367  6321     1  0  75   0 -  4330 -      ?        00:00:22 Xvnc
0 S  1367  6325     1  0  75   0 -  1060 -      ?        00:00:00 vncconfig
0 S  1367  6326     1  0  75   0 -  2712 -      ?        00:00:00 xterm
0 S  1367  6327     1  0  77   0 -  1496 -      ?        00:00:00 twm
0 S  1367  6329  6326  0  75   0 -  1154 -      pts/10   00:00:00 bash
5 S  1367 15918 15914  0  75   0 -  2577 -      ?        00:00:04 sshd
0 S  1367 15921 15918  0  75   0 -  1159 -      pts/9    00:00:00 bash
5 S  1367 23770 23766  0  78   0 -  2578 -      ?        00:00:00 sshd
0 S  1367 23771 23770  0  78   0 -  1192 -      pts/13   00:00:00 bash

[carey@csl ~]$ ps -l -u jsmith
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S  2548 25055 25051  0  75   0 -  2544 -      ?        00:00:00 sshd
0 S  2548 25056 25055  0  75   0 -  1406 -      pts/1    00:00:00 tcsh
5 S  2548 25528 25526  0  75   0 -  2544 -      ?        00:00:00 sshd
0 S  2548 25529 25528  0  75   0 -  1390 -      pts/2    00:00:00 tcsh
0 S  2548 26065 25056  0  75   0 -   380 -      pts/1    00:00:00 a.out
0 S  2548 26068 25529  0  75   0 -   381 -      pts/2    00:00:00 a.out
```

Processes and Threads

12 10. Most modern operating systems provide support for both *processes* and *threads*.

(a) (3 marks) What is a *process*?

```
- ''a program in execution''
- an active entity that consumes system resources
- basic unit of resource allocation in an operating system
- has identity (PID) and attributes (owner, size, priority, etc.)
- represented by a Process Control Block (PCB)
```

(making any 3 of these points, or similar ones, would suffice)

(b) (3 marks) What is a *thread*?

```
- ''a flow of control within a process''
- a thread is lightweight entity (part of a process)
- has a stack and program counter of its own
- basic unit of CPU allocation in modern operating systems
```

(making any 3 of these points, or similar ones, would suffice)

(c) (2 marks) List two key differences between processes and threads.

```
- a process is heavyweight, while a thread is lightweight
- a thread is a part of a process (containment within)
- a process must have at least one thread
- a process can have multiple concurrent threads
- thread provides finer grain control for tasks, scheduling, etc.
- IPC and sharing is easier between threads than processes
```

(making any 2 of these points, or similar ones, would suffice)

(d) (2 marks) List two similarities between processes and threads.

```
- both are active entities
- both are supported on most modern operating systems
- both are examples of ''tasks'' in the Linux system
- both have many attributes (e.g., owner, parent, priority)
- both have stack space and program counter
- both require CPU scheduling in order to do their work
```

(making any 2 of these points, or similar ones, would suffice)

(e) (1 mark) What Linux system call is normally used for process creation?

`fork()`

(f) (1 mark) What Linux system call is normally used for thread creation?

`clone()` (note that `pthreadcreate()` is not a system call)

*** THE END ***