CPSC 457
OPERATING SYSTEMS

MIDTERM EXAM SOLUTION

Department of Computer Science
University of Calgary
Professor: Carey Williamson

March 9, 2010

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

It is a 75 minute exam, with a total of 60 marks. There are 11 questions, and 9 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: _____ / 60 = _____ %

Multiple Choice

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

1 1. Three important design principles in operating systems are:

(a) **caching, virtualization, and support for concurrency**

(b) abstraction, fairness, and context-switching

(c) throughput, response time, and support for priority

(d) parallelism, concurrency, and multi-anything

(e) policy, mechanism, and superuser control

(f) all of the above

1 2. Changing the permissions on your files is easy in Linux because:

(a) **there is a built-in command-line utility for doing so**

(b) there is a C programming library routine for doing so

(c) there is a shell program for doing so

(d) there is a Java program for doing so

(e) there is a superuser on call for doing so

(f) all of the above

1 3. In *message-passing* IPC on a Linux system, the maximum message size permitted is:

(a) 1 byte

(b) 4 bytes

(c) 32 bytes

(d) 64 bytes

(e) 140 bytes

(f) **none of the above**

1 4. When a process is created using the classical `fork()` system call, which of the following is **not** inherited by the child process?

(a) process address space

(b) **process ID**

(c) user ID

(d) open files

(e) signal handlers

(f) none of the above

1 5. The *text segment* of a process address space contains:

(a) the statically allocated data associated with the process

(b) the dynamically allocated data associated with the process

(c) **the executable code associated with the process**

(d) the inter-process communication (IPC) messages for the process

(e) the text-messaging chat messages for the process

(f) all of the above

1 6. The threading model supported by the Linux operating system is:

(a) many-to-one

(b) **one-to-one**

(c) one-to-many

(d) many-to-many

(e) all of the above

(f) none of the above

Operating System Principles

10 7. In class, we have illustrated operating system principles using examples from former and current operating systems, including Multics, Unix, Linux, Windows, and Mac OS.

(a) (2 marks) What is an *Operating System (OS)*?

An OS is that portion of the system software in charge of managing the physical hardware resources of the computer system, and sharing them appropriately amongst the processes and users of the system according to a resource allocation policy.

(b) (5 marks) List and briefly describe any 5 of the typical services provided by an OS.

user interface: command-line, batch, or GUI
usable system: environment for program development and execution
file system: a place to store and manipulate data objects
communication: a way for processes to communicate and interact
accounting: keeping track of system resource usage
error detection: detecting, reporting, recovering from system problems
security: protection of system from malicious users or actions

(c) (3 marks) Most operating system functionality can be provided using a variety of mechanisms, including system calls, built-in commands, and user-level programming support. Give 3 examples, from either Linux, MacOS, or Windows, of useful operating system functionality, indicating clearly the mechanism by which the service is provided.

file manipulation: built-in commands and utilities in Unix, as well as I/O redirection in the shell
GUI: graphical user interface (e.g., Windows) for managing terminal windows, desktop icons, etc, often as an application program
thread management: application-level and/or system-level support for the creation and management of threads of execution
timers: system-calls and/or built-in commands in Unix

Operating System Concepts and Definitions

12 8. 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) "process" and "processor"

process: a ‘‘program in execution’’; an active software entity, which has states and attributes, and can hold resources (mem, CPU).
processor: a hardware resource (CPU) for executing programs.
A process is executed on a processor.

(b) (2 marks) "`fork()`" and "`exec()`"

fork: a system call to create a new process, which by default is a complete copy of its parent, including same code and data.
exec: a system call to change the code associated with a process.
The fork() comes first, with an optional exec() afterwards.

(c) (2 marks) "signal" and "pipe"

signal: a primitive mechanism for IPC that can send simple integer valued messages from one process to another (e.g., shell job control).
pipe: a general data-stream IPC mechanism, pioneered as a command-line feature in the Unix shell. Conveys arbitrary data one-way from one process to another, by coupling stdout of one to stdin of the next.

(d) (2 marks) "C program" and "shell program"

C: application-level or system-level program written in a high-level language. Can use libraries and syscalls. Compiled to object code to run.
shell: a simple user-level program or script that makes use of built-in commands and utilities and files. Interpreted, not compiled.

(e) (2 marks) "pre-emptive" and "non-preemptive"

preemptive: ability to interrupt and suspend execution of a process at any time. A feature of CPU scheduling policies like RR and SRPT.
non-preemptive: inability to interrupt a running job. Can only do so when it completes, or voluntarily relinquishes CPU itself. A feature of scheduling policies like FIFO and SJF.

(f) (2 marks) "CPU-bound" and "I/O-bound"

CPU-bound: a characteristic of a job that needs lots of computation, and thus needs lots of CPU time to finish. Also called a CPU hog.
I/O-bound: a characteristic of a job that needs lots of input and output handling, whether from files or from interactive users. Spends more time in I/O wait state than in active usage of CPU.

Processes and Threads

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

(a) (2 marks) What is a *process*?
A process is a ‘‘program in execution’’. It is an active but relatively heavyweight entity, often with many attributes and resources associated with it. Fundamental unit of resource allocation in classic OS.

(b) (2 marks) What is a *thread*?
A thread is ‘‘a flow of control within a process’’. It is an active but relatively lightweight entity. Has attributes and resources associated with it. Fundamental unit of CPU allocation in modern OS.

(c) (3 marks) List any 6 of the attributes that an operating system maintains to keep track of information about a process.
process ID
owner (user ID)
parent
priority
memory resource allocation
open files
and many more...

(d) (3 marks) List the 3 most important attributes that an operating system maintains to keep track of information about a thread.
program counter (PC)
registers
stack
thread ID was also a possible answer, for partial marks

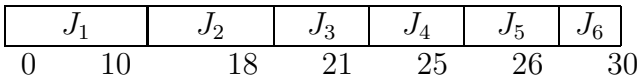
(e) (2 marks) List 2 reasons why the scheduling of processes and threads on a multi-processor system is more complicated than scheduling them on a uni-processor system.
Concurrently executing processes may require access to shared data.
More than one processor is available, so load-balancing is an issue.
Difficult to debug of non-deterministic execution sequences. Cache affinity needs to be considered in CPU scheduling decision.
Modern CPU architectures have many subtle features, such as multi-core, that are hard to exploit properly in user-level programming.

CPU Scheduling

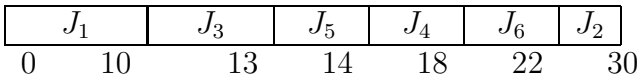
10 10. Consider the following set of jobs to be scheduled for execution on a single CPU system.

Job	Arrival Time	Size (msec)	Priority
J_1	0	10	2 (Silver)
J_2	2	8	1 (Gold)
J_3	3	3	3 (Bronze)
J_4	10	4	2 (Silver)
J_5	12	1	3 (Bronze)
J_6	15	4	1 (Gold)

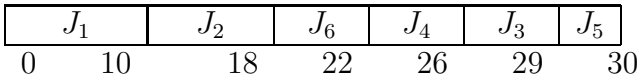
(a) (2 marks) Draw a Gantt chart showing FCFS scheduling for these jobs.



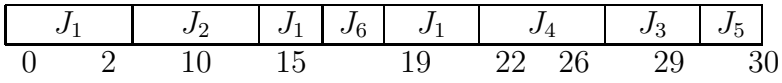
(b) (2 marks) Draw a Gantt chart showing (non-preemptive) SJF scheduling.



(c) (2 marks) Draw a Gantt chart showing non-preemptive PRIORITY scheduling.



(d) (2 marks) Draw a Gantt chart showing preemptive PRIORITY scheduling.



(e) (2 marks) Which of the foregoing scheduling policies provides the lowest waiting time for this set of jobs? What is the waiting time with this policy? (Show your work)

```
SJF.
J1:  0   J2:  20   J3:  7   J4:  4   J5:  1   J6:  3
Average is 35/6 time units.
```

Operating System Utilities

10 11. The output on the next page is from a lightly-used Linux system on the recent weekend. Use the output and your knowledge of Linux systems to answer the following questions:

(a) (1 mark) On what date (approximately) was this system most recently rebooted?
`September 9, 2009`

(b) (1 mark) How many distinct users are logged in on this sytem?
`2 (carey and jess, though carey is logged in twice)`

(c) (1 mark) Approximately how many processes are currently present on this system?
`179`

(d) (1 mark) How much physical memory (RAM) does this particular Linux system have?
`3.2 GB`

(e) (1 mark) What percentage (approximately) of the memory is currently in use?
`85%`

(f) (1 mark) Which process (PID or name) has consumed the most CPU time so far?
`29105 (linux)`

(g) (1 mark) Which process (PID or name) is currently consuming the most memory?
`3740 (fractal)`

(h) (1 mark) Which process (PID or name) is the oldest process on this system?
`1 (init)`

(i) (1 mark) Which process (PID or name) has created the most child processes?
`1179 (csh)`

(j) (1 mark) What is your professor's favourite editor?
`emacs`
(negative infinity for those who said vi!!)

```
[carey@csl]$ w
 09:11:09 up 184 days, 17:43,  3 users,  load average: 0.52, 0.37, 0.35
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
carey    pts/5    csg              06:32    0.00s  0.47s  0.00s w
carey    pts/6    ict736a          Thu09   19:41m  0.04s  0.04s -csh
jess     pts/8    ict624           08:16   31:25m  0.02s  0.06s -bash

[carey@csl]$ ps
  PID TTY          TIME CMD
 1179 pts/5    00:00:00 csh
 1407 pts/5    00:00:00 emacs-x
 3721 pts/5    00:00:00 sleep
 3740 pts/5    00:01:00 ./fractal
 3748 pts/5    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  1179  1178  0  75   0 -  1434 rt_sig pts/5    00:00:00 csh
0 T   214  1407  1179  0  75   0 -  3133 finish pts/5    00:00:00 emacs-x
0 S   214  3721  1179  0  77   0 -   985 -      pts/5    00:00:00 sleep
0 R   214  3753  1179  0  75   0 -  1102 -      pts/5    00:00:00 ps
0 R   214  3740  1179  0  75   0 -  4270 -      pts/5    00:01:00 ./fractal

[carey@csl]$ top
top - 09:11:25 up 184 days, 17:58,  3 users,  load average: 0.26, 0.34, 0.34
Tasks: 179 total,   2 running, 141 sleeping,  36 stopped,   0 zombie
Cpu(s):  2.5%us,  5.6%sy,  0.0%ni, 91.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%
Mem:   3369524k total,  2915008k used,   454516k free,   356296k buffers
Swap:  2040244k total,    54544k used,  1985700k free,  2361776k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29105 cristo    16   0  1290 1260 1260 R   23  3.8  54:13.56 linux
 3567 carey     15   0   400  400  400 T    8  0.6   0:00.24 top
 1407 carey     15   0  3133 2820  311 S    5  7.2   0:00.18 emacs-x
 3740 carey     15   0  4270 4100  170 S    2  8.1   0:01.00 ./fractal
 3721 carey     15   0   980  980  980 S    2  1.2   0:00.04 sleep
    1 root      15   0  2060  476  448 S    0  0.0   0:41.56 init
    2 root      RT  -5     0    0    0 S    0  0.0   0:07.75 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:07.17 ksoftirqd/0
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
```

*** THE END ***