CPSC 457
OPERATING SYSTEMS

FINAL EXAM SOLUTION

Department of Computer Science
University of Calgary
Professor: Carey Williamson

April 21, 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 120-minute exam, with a total of 100 marks. There are 20 questions, and 13 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. The marks for each question are indicated in the margins.

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

Student Name: _____

Student ID: _____

Score: _____ / 100 = _____ %

Multiple Choice

Choose the best answer for each of the following 12 questions, for a total of 12 marks.

1 1. The **original designers/developers** of the Unix operating system were:

(a) **Ritchie and Thompson**

(b) Kernighan and Ritchie

(c) Gates and Balmer

(d) Simon and Garfunkel

(e) obviously insane

1 2. The **Mach operating system** was:

(a) an example of a layered OS design

(b) **an example of a micro-kernel OS design**

(c) later renamed Mac OS X

(d) developed at the University of Calgary in the 1980s

(e) developed at the University of Saskatchewan in the 1980s

1 3. **User Mode Linux (UML)** is an example of:

(a) **a virtual machine**

(b) a virtual processor

(c) a virtual memory

(d) a virtual file system

(e) a virtual reality

1 4. The `su` **command** on a Linux system allows a user to:

(a) suspend the execution of a selected process

(b) start up the system following a shutdown

(c) **temporarily assume the identify of the superuser (or another user)**

(d) invoke legal proceedings to collect financial damages for suffering

(e) remotely start their SUV in cold weather

1 5. One example of a **hardware solution** to the critical section problem is:

(a) Peterson's Algorithm

(b) Banker's Algorithm

(c) **Test and Set**

(d) Compare and Shop

(e) Compare and Pray

1 6. The **second-chance (clock) algorithm** is an efficient approximation technique for:

(a) **LRU page replacement**

(b) LFU page replacement

(c) benchmarking file system performance

(d) benchmarking raw disk I/O performancne

(e) two, but not all, of the above

1 7. With the **FIFO page replacement** policy, and enough space for storing 3 page frames, the memory page reference string 'ABCABDDCABCD' would produce:

(a) 5 page faults

(b) 6 page faults

(c) 7 page faults

(d) **8 page faults**

(e) none of the above

1 8. The primary difference between iSCSI and SCSI is that **iSCSI**:

(a) supports interactive data transfers

(b) supports direct data transfers between inodes

(c) is used for portable media devices like iPods

(d) **supports data transfers over a network using the Internet Protocol**

(e) none of the above

1 9. The **superblock** in a Linux file system is important because:

(a) it holds all of the inodes

(b) **it contains all file system configuration parameters**

(c) it is owned by the superuser

(d) it is owned by the superintendent

(e) it is owned by Superman

1 10. The **cylinder group** optimization in the Unix Fast File System was designed to:

(a) increase the size of data blocks used

(b) increase the storage capacity of disks

(c) increase the maximum file size permitted

(d) **reduce the number of head seeks during typical file operations**

(e) provide easy containment of viruses and malware

1 11. A typical **Linux file system** uses:

(a) contiguous space allocation for all files in the file system

(b) a linked-list approach for storing file data blocks

(c) a direct indexed approach for storing file data blocks

(d) an indirect indexed approach for storing file data blocks

(e) **a combination of both (c) and (d)**

1 12. Modern implementations of the **Network File System (NFS)**:

(a) use caching to improve file system performance

(b) use state information to improve system security (and performance)

(c) allow TCP and UDP as transport-layer protocols

(d) can operate over local-area and wide-area networks

(e) **all of the above**

OS Concepts and Definitions

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

(a) (3 marks) "short-term scheduler" and "long-term scheduler"

```
These terms are both related to CPU scheduling.
long-term scheduler:  decides which processes go in memory (versus on disk)
short-term scheduler:  decides which memory-resident process gets the CPU
```

(b) (3 marks) "binary semaphore" and "counting semaphore"

```
These are types of semaphores used for regulating access to shared resources.
binary semaphore:  has value 0 or 1; suitable for single instance of a resource.
counting semaphore:  has integer value from 0 to n; suitable for multiple
identical instances of a resource.
In both cases, the wait() and signal() operations are used as the means
to manipulate (i.e., decrement or increment) their values.
```

(c) (3 marks) "deadlock prevention" and "deadlock avoidance"

```
These are both different ways to deal with the potential problem of deadlock.
prevention:  constrain the resource allocation rules in your system so that
deadlock is provably impossible (e.g., numbered resources, no circular wait)
prevention:  allocate resources carefully, based on a priori information,
so that no deadlock can occur (e.g., Banker's Algorithm)
```

(d) (3 marks) "Network-Attached Storage (NAS)" and "Storage Area Network (SAN)"

```
These are both different ways to provide external storage.
NAS: storage is provided over a local area network (LAN) or wide
area network (WAN), rather than directly host-attached.
Typically done using Internet protocols (e.g., NFS, TCP/IP).
SAN: storage is provided using a dedicated special-purpose network,
or third-party provider of services (e.g., HP, IBM), with proprietary
protocols and infrastructure.
```

Processes and Inter-Process Communication (IPC)

10 14. Answer the following questions about processes and inter-process communication.

(a) (2 marks) What is a **process control block**?

PCB is a data structure to record information about a process, its attributes, and its resources. Called a task struct in Linux. Stores information such as PID, owner, parent PID, priority, memory map, open files, CPU usage, etc.

(b) (2 marks) What is a **context-switch**?

An explicit change in which process has (exclusive) use of the CPU. Involves suspending the execution of the currently executing process, saving its state information into its PCB, loading the state of the new process from its PCB, and starting the execution of the new process.

(c) (3 marks) Multiple processes can communicate with each other using **shared-memory** IPC techniques. How does shared-memory IPC work? Give some examples of the Linux system calls that can be used for shared-memory IPC.

Processes allocate a region of memory that is mapped into their address space, but marked in a special way as shared by more than one process. All processes sharing this memory can read or write arbitrary data to this region. In Linux, this is done using system calls such as shmget() or shmat(). Once the shared region is set up by the OS, there is no further involvement of the kernel in the IPC between the processes.

(d) (3 marks) Multiple processes can communicate with each other using **message-passing** IPC techniques. How does message-passing IPC work? Give some examples of the Linux system calls that can be used for message-passing IPC.

Message-passing IPC allows processes to send arbitrary messages back and forth to each other via the kernel. Process P1 sends the message, which is copied to the kernel, and then copied to process P2. Examples of system calls are send() and rcv(). Note that the kernel is involved in the delivery of every message.

Process Synchronization

10 15. Answer the following questions about process coordination and synchronization.

(a) (2 marks) What is a **race condition**?
A situation in which uncoordinated read/write access by multiple processes to the same shared data can result in non-deterministic, unpredictable, and often incorrect results.
The word ‘‘race’’ mean that any process might get there ‘‘first’’.

(b) (2 marks) What is a **critical section**?
A piece of code requiring mutually exclusive access to shared data. If more than one process does so at the same time, problems occur.

(c) (3 marks) What are the primary mechanisms that an **applications programmer** can use to ensure correct process synchronization when manipulating shared data? Give some examples from Linux or Windows if you wish.
Locking. Semaphores. Mutexes. Monitors.
For application programs, some support is available in the C library. PThreads also has thread management support for mutexes and locking.

(d) (3 marks) What are the primary mechanisms that a **systems programmer** can use to ensure correct process synchronization when manipulating shared data? Give some examples from Linux or Windows if you wish.
Careful coding is the best approach, since the C programming library is not available. One can use the kernel mechanisms for mutexes, reader/writer locks, spinlocks, etc. These are defined in the locks.c and locks.h source code files in Linux/UML.
Another option is to explicitly disable system interrupts before a critical section, and re-enable them afterwards.
Application programs don’t really have this option.

Memory Management

12 16. Answer the following questions about OS memory management.

(a) (3 marks) One motivation for page-based memory management is the concern about memory fragmentation. What is the "fragmentation" problem? Why does it occur?
This is an instance of the classical dynamic storage allocation problem. With variable-sized jobs arriving/departing at arbitrary times, one can end up in a state with just part of the available memory in use,but the unused parts are scattered all over the place in small pieces, many of which are too small to be useful. Wasted!

(b) (3 marks) In memory management, two different types of fragmentation can occur, namely **internal fragmentation** and **external fragmentation**. Clarify the differences between these two types of fragmentation. Which one is solved by the use of fixed-size page frames?
External fragmentation is the one defined above, wherein unallocated memory space between processes are too small to be useful individually, and so are wasted.
Internal fragmentation is when there is unused memory space within the allocation to a specific process (e.g., rounding up to the page size), and cannot be used by anyone else.
Fixed-size pages solve external fragmentation, not internal.

(c) (3 marks) In class, we made a logical (quantitative) argument for why the "sweet spot" of typical page frame sizes is usually between 1024 bytes (1 KB) and 4096 bytes (4 KB) in modern operating systems. What are the primary technical reasons for this? That is, give at least 2 reasons why page sizes smaller than this would be a poor choice, and at least 1 reason why page sizes larger than this would be a poor choice.
Smaller can be a bad choice because the number of pages required can be huge, the size of the page table is large, and multiple memory references are needed to resolve each logical address access. Larger can be a bad choice if internal fragmentation becomes high.

(d) (3 marks) Some operating systems have hardware support that allows *multiple* page sizes to be used, perhaps even simultaneously on the system (e.g., the Intel architecture that we discussed in class). Are multiple page sizes a good idea? Why, or why not?
Yes. This makes sense if the system has diverse types of jobs: some are CPU-intensive, some are I/O intensive, some are memory intensive, etc. Having a super page size makes things more efficient for working with large data sets (working sets), without compromising on internal fragmentation for the most common jobs with regular page sizes.

Virtual Memory Systems

12 17. While virtual memory systems allow processes to execute with only part of their address space in memory at a given time, it creates the possibility of a **page fault**, wherein a process must be temporarily suspended when it references a valid page of its address space that is not currently in main memory.

(a) (6 marks) What are the 6 main steps involved in servicing a page fault? Draw a diagram if you wish.

1. Suspend the executing process.
2. Trap into the kernel.
3. Find the neededpage on backing store.
4. Copy the page into a vacant page frame in memory (evicting one if needed).
5. Update the page table for this process (e.g., valid page entry).
6. Re-issue the instruction that originally caused the page fault.

(b) (2 marks) What hardware mechanisms are required in order to make virtual memory systems efficient? How do they work? Give at least two examples.

Memory Management Unit (MMU): translates logical to physical addresses.
Translation Lookaside Buffer (TLB): a cache to optimize the common case of referencing a page you have seen before (recently), and avoiding the the extra memory lookups required for general page references.
Other possible items: CAMs, page tables, system trap handlings, etc.

(c) (4 marks) In class, we discussed the "top 10 cool things" that you can do with virtual memory systems that would be difficult or impossible to do in classical operating systems based solely on swapping. List any 4 of these cool things, and explain why they are useful.

Arbitrary-size processes: logical address space can exceed physical memory.
Copy on write: only make a physical copy of a page when needed.
Zero fill on demand: common case of data initialization.
Virtual fork: efficient process creation.
Shared pages: much easier for shared-memory IPC.
Lots of fun control bits (reference bit, dirty bit, valid bit, etc.)

File Systems

12 18. Answer the following questions about file systems and the Linux file system.

(a) (3 marks) One of the decisions to be made in operating system design is whether to include the file system as part of the core kernel or not. Give at least 2 reasons why making the file system part of the OS would be a good idea, as well as at least 1 reason why implementing the file system outside the OS might make sense.
`Inside kernel: faster, protection and security, caching, buffering, close coupling to storage system and I/O optimization.`
`Outside kernel: flexibility, portability, OS independence.`

(b) (3 marks) Give 3 examples of file meta-data (i.e., file attributes) that are maintained in a typical file system (e.g., Linux or Windows).
`File size, owner, type, permissions, date, inode number.`

(c) (3 marks) One file system design decision is to choose between "structured" or "unstructured" files. Explain what is meant by each of these terms. Which approach is used in the Linux file system?
`Structured: explicit record-like structure, fixed size fields or regions.`
`Unstructured: sequence of bytes, ''big blob of data'', any format goes.`
`Linux uses unstructured files: an ordered sequence of bytes, with a name`

(d) (3 marks) Another file system design decision is to choose between "typed" or "untyped" files. Explain what is meant by each of these terms. Which approach is used in the Linux file system?
`Typed: files have an explicit type, and a stated file naming convention such as the file name suffixes used in Windows (e.g., .doc, .exe, .htm)`
`Untyped: no explicit type, name files however you wish (e.g., fred)`
`Linux uses untyped files. However, there is still a naming convention, and a ''magic number'' technique to identify most file types.`

Linux System Details

10 19. The following page shows the output from a particular user's recent session on a local Linux system. Use this output and your knowledge of Linux to answer these questions.

(a) (1 mark) What programming language is being used in this example?
`C`

(b) (1 mark) What is the name of the source code file being used in this example?
`meow.c`

(c) (1 mark) What inode number does the source code file have in the file system?
`17908148`

(d) (1 mark) What is the name of the object code file used in this example?
`meow`

(e) (1 mark) How large (in bytes) is the object code file used in this example?
`5331`

(f) (1 mark) What specific type of executable file is it?
`ELF 32-bit`

(g) (1 mark) How many processes are created when the final command is executed?
`Infinitely many, or as many as your Linux system can handle, until the program is killed by the user! Child forks a process every 60 seconds.`

(h) (3 marks) What is the output produced when the final command is executed?

```
I like chicken!
I like liver!
I like napping!
Sat Apr 17 13:50:00 MDT 2010
I like chicken!
I like liver!
I like napping!
Sat Apr 17 13:51:00 MDT 2010
I like chicken!
I like liver!
I like napping!
Sat Apr 17 13:52:00 MDT 2010
...
```

```
[carey@csl]$ ls
a.out      meow       meow.c

[carey@csl]$ cat meow.c
/* Test program for CPSC 457 final exam */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
  {
    int pid;

    printf("I like chicken!\n");
    pid = fork();
    if( pid == 0 )
      {
        printf("I like napping!\n");
        sleep(60);
        execlp("./meow", "meow", NULL);
        printf("I like noodles!\n");
      }
    else if( pid > 0 )
      {
        printf("I like liver!\n");
        execlp("/bin/date", "date", NULL);
        printf("I like tuna!\n");
      }
    printf("Please deliver!\n");
  }

[carey@csl]$ cc -o meow meow.c

[carey@csl]$ file meow
meow: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.9, dyna

[carey@csl]$ ls -lis
17908146   8 -rwxr-xr-x 1 carey profs   5254 Apr 17 13:38 a.out
17908147   8 -rwxr-xr-x 1 carey profs   5331 Apr 17 13:50 meow
17908148   4 -rw-r--r-- 1 carey profs    522 Apr 17 13:47 meow.c

[carey@csl]$ date
Sat Apr 17 13:48:31 MDT 2010

[carey@csl]$ ./meow
```

Future OS Trends

10 20. One recent trend in operating systems research has been on **virtualization**, particularly in the management and operation of large-scale systems, such as enterprise computing systems, data centres, high-performance computing clusters, or cloud computing facilities.

(a) (3 marks) What is a **virtual machine**? Why is this concept especially useful for those who manage large-scale computing systems?

A virtual machine allows one or more guest operating systems to run on a host OS on a given physical machine. This makes it easier to share physical resources amongst multiple competing jobs, often on a platform-independent basis (e.g., Windows, Linux). It also makes it easier to migrate jobs for load balancing or server consolidation.

(b) (3 marks) What is a **virtual file system**? Why is this concept especially useful for those who manage large-scale computing systems?

VFS is an abstracted view above the file system interface, allowing the same file system API to be used whether files are stored locally or remotely. It hides the details of the file system implementation, from user or application, and provides greater interoperability between heterogeneous platforms and file systems. Basically, data can be stored and accessed from almost anywhere, using the same VFS API.

(c) (4 marks) What are other additional technical issues that the managers of large-scale computing systems must consider? Are there virtualization strategies that might be appropriate for these issues, or not? What other solutions would you recommend?

Energy consumption. Dynamic electrical utility cost pricing.
Network connectivity. Need high-speed links for large data transfers.
Cooling. Need water nearby, and lots of it.
Security. Need data security, user authentication.
Usability. Single sign on. Web based services. Platform independence.
Scheduling. Diverse jobs. Shared resources. Fairness metrics.
Accounting. Usage based charging.

*** THE END ***