University of Mary Washington

CPSC405 Introduction to Operating Systems Fall 2012

FINAL EXAM

- This exam is due 2:30pm on Tuesday 11 December. You are to submit the exam to the gmail address submit.o.bot. If your exam is delivered after this deadline, it will not be accepted.

- Your exam submission can be in one of three formats:

  - plain text file
  - pdf
  - postscript

  No other formats will be accepted.

- This test is to be completed **individually** without outside help. This includes no help from peers or the Internet. You are free to use any resources used for the class including the textbook, handouts, and any class notes you made. If there is evidence that any part of the exam was completed with outside help you will receive a 0 for the entire exam.

- To discourage guessing and brain-dump style answers you will receive 20% of the XP for problems you leave completely blank. If you attempt a problem you start at zero XP. By *problem* I mean any numbered problem--subproblems do not count. This means that if you attempt one part of a problem it is best to answer the remaining parts. If you are less than 100XP away from the grade you desire for the class you can submit a blank exam and you will receive 100XP. (Keep in mind that the minimum score you can receive for the exam is 0)

Part 1. Short Answer (20 XP per problem)

PROBLEM 1

You are working as part of a team on an iPad app that enables users to fill out claim forms on their iPads. The data is stored remotely on your company's servers. To improve robustness, the team decides to implement transactions. There are three competing ideas as to where to store the transaction log:

- on the server's disk
- in memory on the server
- in persistent memory on the iPad.

Please provide your opinions about these options.

PROBLEM 2

Before Rosemary, and in fact, before computer science was a separate department from Math, the department had a single machine called Sage, used for classroom instruction. Once the department had more than a dozen majors, students complained about the slow response time. The department bought an identical second machine and split the students equally between the machines. Oddly enough, students' response time got better by much more than a factor of two. What is the most likely reason for such a phenomenal speedup?

PROBLEM 3

The following code contains a monitor lock and condition variables. There are two types of threads that use this class: black threads and white threads. Black threads call on a frequent but random interval the black method, white ones call on a frequent but random interval the white method. Please describe what this class does.

```
class Shared:
    def __init__(self, start=0, capacity = 10):
        self.lock = Lock()
        self.MAX = capacity
        self.itemAdded = Condition(self.lock)
        self.itemRemoved = Condition(self.lock)
        self.nFull = 0

    def isFull(self):
        return True if self.nFull == MAX else False

    def isEmpty(self):
        return True if self.nFull == 0 else False
    def white(self):
        self.lock.acquire()
        while(self.isEmpty()):
            self.itemAdded.wait()
        self.nFull -= 1
        self.itemRemoved.notify()
        self.lock.release()


    def black(self):
        self.lock.acquire()
        while(self.isFull()):
            self.itemRemoved.wait()
        self.nFull += 1
        self.itemAdded.notify()
        self.lock.release()
```

PROBLEM 4

You are designing an operating system for an autonomous spacecraft. The application programs that will run on the OS are of three types: critical, normal, and background. Critical tasks are to enjoy better response time than normal tasks which in turn will enjoy better response time than background tasks. You are to design such a system so that all tasks will still get some progress, but with the indicated preferences in place.

1. Will a fixed priority scheme with pre-emption and 3 fixed priorities work? Why, or why not?

2. Will a UNIX-style multi-feedback queue work? Why, or why not?

3. If none of the above works, could you design a scheduling scheme that meets the requirements?

PROBLEM 5

Consider the following hard disk:

RPM	7200
Heads/Platters	6/3
sector size	4096 bytes
sectors per track	1024

cylinders	131,072
seek time	~ 0 ms if tracks adjacent otherwise 1 + .003 * t where t is the number of tracks that the disk is seeking over

What is the storage capacity of the disk in gigabytes? To get credit you must show your work.

What is the sequential transfer bandwidth expressed in bytes/second or megabytes/second? Explain briefly (for ex., show your work)

PROBLEM 6

Consider two computer systems which are identical except for the following characteristics.

	Mustang-A	**Mustang-B**
L3 Cache	16MB	4MB
Main Memory	4GB	16GB

Please explain under what circumstances Mustang-A would be the preferred purchase, and under what circumstances Mustang-B would be.

PROBLEM 7

You are the virtual memory guru on an team designing an operating system for a tablet. The OS will provide a kernel mode and facilities to enter kernel mode on interrupt, exception, and trap. Unfortunately, the RISC chip you are using has only primitive virtual memory support-- it has two base registers and two bounds registers for applications and a separate pair of base and bounds registers when in kernel mode. The hardware has no paging support. Unfortunately, all of the application software relies on 8 segments spread across a sparse address space. How could you modify the OS to support applications that require multiple segments per address space using this hardware?

PROBLEM 8

Consider the following experiment and **explain the observations.**
A program is run by itself on a paging machine. The program begins execution with its first procedure page. As it runs, the pages it needs are demand paged into available page frames. There is a dial external to the computer that allows a person to set the maximum number of page frames the program may use. In addition there is an LED display that indicates the total number of page faults that occurred during program execution. Initially, the dial is set to 2 frames and the program is run to completion. The dial is then set at 3 frames and again the program is run to completion. This process is continued until the dial is eventually set to the number of available pages in real storage, and the program is run for the last time. For each run, the execution time and number of page faults is recorded

Observation 1:
As the dial is changed from 2, to 3, to 4, the number of page faults declines. As the dial is changed to 5 the number of page faults rises. From 5 to 6 and onward, the number of page faults again declines.

Observation 2 (a completely separate study):
For a different program we follow the same procedure. As the dial is changed from 2, to 3, to 4, the execution time improves dramatically. From 4, 5, to 6, the execution time still improved each time, but less dramatically. With the setting of 7 and higher the execution time is essentially constant.

PROBLEM 9

You are designing a file system from scratch. The disk driver allows you complete control over the placement of data on the disk. Assuming that you have settled for a File Allocation Table (FAT) architecture, where would be the best place to store the table on disk?

PROBLEM 10

We are designing an operating system for a mobile device that uses a single core cpu. A member of our team says that we do not need to provide support for threads since multithreading would have no benefit on a single core CPU. Do you agree? Briefly explain.

PROBLEM 11

In our user level thread project we called getcontext in both ULT_Yield and ULT_CreateThread. For each of these functions, explain what specific changes we needed to make to the data structure returned by getcontext before adding it to the queue.

Part 2.

PROBLEM 1 (35XP)

Consider a 32-bit virtual address divided up as follows:

Here are the relevant tables (values in hexadecimal):

4 bit outer page number	12 bit inner page number	16 bit offset

Outer Page Table	
0	Page Table A
1	Page Table B
x	(rest invalid)

Page Table A	
0	CAFE
1	DEAD
2	BEEF
3	BA11
x	rest invalid

Page Table B	
0	F000
1	D8BF
x	(rest invalid)

Find the physical address corresponding to each of the following virtual addresses (answer 'bad address') if the virtual address is invalid)

1. 0x00000000
2. 0x20022002
3. 0x10015555
4. 0x10030099

PROBLEM 2 (50XP)

I run a company that rents out stunt dogs for movies. In this problem you will need to implement a monitor to help film production companies (modeled as threads) synchronize access to my 100 dogs and 50 handlers. Here are the constraints:

The relationship between handlers and dogs is that handler 0 can handle dogs 0 and 1, etc. That is, handler h works with dogs 2h and 2h + 1.

If a handler is being used with one dog, she cannot be used to handle another

A film production company requests a particular dog (which implies the need for a particular handler). However, there may be more production companies than dogs. If a company wants a particular dog, but can't get either that dog or its handler, it waits until the dog and handler are available. When a production company finishes with a dog, the monitor should notify any production companies that are waiting for the handler that she is now free.

You must allow multiple dogs to be out working at once, and you must not have busy waiting or spin loops.

Again, there may be many, many instances of film production companies (running in their own threads), all of which use the same monitor, mon.

In the skeleton code provided, please fill in the monitors remaining variable(s) and implement the monitor's `get_handler()` and `return_handler()` methods.

```
class Monitor:
    def __init__(self, numberOfHandlers):
        self.lock = Lock()
        ## ADD ANY OTHER VARIABLES YOU NEED HERE
        self.handlers = []
        for  i in range(numberOfHandlers):
           self.handlers.append('FREE')

    def get_handler(self, desired_handler):
         ## FILL IN THIS FUNCTION

    def return_handler(self, desired_handler):
        ## FILL IN THIS FUNCTION
```

PROBLEM 3 (35XP)

Consider the following virtual memory architecture for a 32-bit processor for a game console.

- Each page is 2KB (2^{11} bytes)
- Physical memory is 32GB (2^{35} bytes)2**10
- The paging system has an outer page table / inner page table design. Each process has an outer page table. Entries in the outer page table point to inner page tables.
- There are 1024 entries in the outer page table
- Associated with each virtual page are 7 bits in the page table. These include 3 bits controlled by the kernel (PTE_P, PTE_U, PTE_W), 2 bits by the hardware (accessed

and dirty) and two bits reserved for the kernel.

1. what is the minimum number of bits in an inner page table entry and explain briefly.
2. what is the total size of an inner page table (pad each entry to the nearest byte)
3. what is the minimum number of bits in an outer page table entry and explain briefly.
4. Consider changing the design so that instead of the address space being byte addressable it is word addressable (where a word equals 2 bytes). How does this affect the above design?

PROBLEM 4 (35XP)

Suppose 5 jobs arrive nearly simultaneously into an empty ready queue in a system with a single CPU core (The order is P0, P1, P2, P3, P4). The following Gantt chart shows the result from RR scheduling.

P0	P1	P2	P3	P4	P0	P1	P2	P4	P0	P2	P0
10	20	30	38	48	58	60	70	80	90	93	95

The times indicated are associated with the vertical lines on the Gantt chart. (The timeline is not to scale). The last job completes at time 95.

1. What is the quantum size (q) for this example?

2. Calculate the job sizes in this example

3. Draw the Gantt chart for RR with q=20; FCFS, and Shortest Remaining Time First.

PROBLEM 5 (40XP)

One day famed University of Mary Washington student Ada Codeslinger had an inspiration while stuck in traffic. She observes that most programs have most of their data at the beginning of the address space. For her homegrown Eagle OS, she decides that she is going to implement her page tables similar to the way Unix implements inodes. She calls this page table design Inode Page Tables. Inode Page Tables are essentially two-level page tables with the following twist: The first half of the page table entries in the master page table directly map physical pages, and the second half of the entries map to secondary page tables as normal. Call the first half the entries fast, and the second half normal.

For the following questions, assume that addresses are 32 bits, the page size is 4 KB, and that the master and secondary page tables fit into a single page.

1 How many virtual pages are *fast* pages?

2 How many virtual pages are *normal* pages?

3 What is the maximum size of an address space in bytes?

4 Inode Page Tables reduce the lookup time for fast pages by one memory read operation.
Do you think that this is an effective optimization? Briefly explain.

PROBLEM 6 (35XP)

Suppose a program references pages in the following sequence

ACBDBAEFBFAGEFA

1. Show how LRU-based paging would fault pages if it has 4 pages of physical memory.

2. Show how the optimal paging algorithm would fault pages if it has 4 pages of physical memory.

3. In the more general case, assume the reference string is of length *p* and *m* in the number of frames of physical memory.
   What is the upper bound on the number of page faults?
   What is the lower bound?

PROBLEM 7 (50XP)

Because of the rising fuel costs more people are riding bikes to the development company I work at. Jim Cowie, my boss, installs a small locker room so people can take a shower and change clothes before work. Jim wants to design a shared object system for the door that enforces the following synchronization constraints:

* There cannot be men and women in the locker room at the same time
* There should never be more than 3 employees squandering company time in the locker room.

Of course your solution should avoid deadlock and starvation. Please design the shared object using locks and condition variables.