Name:

Operating Systems
Fall 2014
Final Exam
December 8, 2014

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. You may use a simple calculator (but you can do without).

You have 120 minutes to solve 7 problems. You get 10 bonus points for your perseverance. As with any exam, you should read through the questions first and start with those that you are most comfortable with. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Partial credit will be offered only for meaningful progress towards solving the problems.

Please read and sign below if you agree with the following statement:

In recognition of and in the spirit of the academic code of honor, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:_____

0	/10
1	/10
2	/10
3	/15
4	/10
5	/15
6	/10
7	/20
Total	/100

1. (10 points) **Short attention span**:

   a. A _____ is a control structure that contains the key information needed by a Unix operating system for a particular file.

   b. The _____ shows the frame location for each page of the process.

   c. _____ in a computer system is organized as a linear, or one-dimensional, address space, consisting of a sequence of bytes or words.

   d. _____ is the range of memory addresses available to a process.

   e. _____ is transparent to the programmer and eliminates external fragmentation providing efficient use of main memory.

   f. The _____ page replacement policy results in the fewest number of page faults.

   g. _____ is a storage allocation scheme in which secondary storage can be addressed as though it were part of main memory.

   h. When the system spends most of its time swapping pieces rather than executing instructions it leads to a condition known as _____ .

   i. The scheduling strategy where each process in the queue is given a certain amount of time, in turn, to execute and then returned to the queue, unless blocked, is referred to as:
      1. Prioritization
      2. Round-Robin
      3. LIFO
      4. All of the above

   j. The processor execution mode that user programs typically execute in is referred to as:
      1. User mode
      2. System mode
      3. Kernel mode
      4. None of the above

2. (10 points) Consider the following program executed by two different processes P1 and P2:

```
{
     shared int x;                          //s1
     x = 10;                                //s2
     while (1) {
          x = x - 1;                        //s3
          x = x + 1;                        //s4
          if (x != 10)                      //s5
               printf("x is %d",x)          //s6
     }
}
```

Consider that the processes P1 and P2 are executed on a uniprocessor system. Note that the scheduler in a uniprocessor system would implement pseudoparallel execution of these two concurrent processes by interleaving their instructions, without restriction on the order of the interleaving.

a. Show a sequence (i.e., trace the sequence of interleavings of statements) such that the statement "x is 10" is printed. Suggested format:
   Pi: <instruction> <relevant new value >
b. Show a sequence that leads to the statement "x is 8" be printed.

Hint: Remember that the increment/decrements at the source language level are not done atomically, e.g., the assembly language code below implements the single C increment instruction (x = x + 1).

```
LD R0,X /* load R0 from memory location x */
INCR R0 /* increment R0 */
STO R0,X /* store the incremented value back in X */
```

3. (15 points) Assume you are given a uniprocessor system with one gigabyte of memory and a 300 gigabyte disk. The OS on the machine has a demand paged virtual memory system with a local page replacement policy and a multi-level feedback queue (MLFQ) CPU scheduler. On the system there are two compute-intensive jobs running: Job-A and Job-B. Job-A has a working set* of 50 gigabytes while Job-B has a working set of 100 megabytes. Assume you left the system to run for a while until it reached a steady state with both jobs running.
    a. Which job would you expect to have a higher CPU scheduling priority from the MLFQ scheduler?
    b. Assume you add a second CPU to system, how would this affect the priorities of the jobs?
    c. Assume you switch from a local to a global page replacement policy, how does this change affect the priorities of the jobs?

    Justify your answer and state any assumptions you make.

    *Think of the working set as the memory footprint. That is, how much memory is seen as consumed by the process when, for example, you run the top command in a Unix terminal or use the Task Manager on a Windows machine.

4. **(10 points)** You are given a bunch of scientific code that contains loops of the form:

```
for (i = 0; i < n; i++)
     for(j = 0; j < n; j++)
          a[j][i] += b[j][i];
```

The matrices `a` and `b` are allocated using code such as:

```
a = malloc (n* sizeof(*a));
for (i = 0; i < n; i++)
     a[i] = malloc(n* sizeof(a[i]));
```

While running the program, you notice that the addition loop trashes the TLB much more than it should. Say why, and give a simple fix. In general, what do you expect to happen to paging performance?

5. (15 points) A process has four page frames allocated to it. (All the following numbers are decimal, and everything is numbered starting from zero.) The time of the last loading of a page into each page frame, the last access to the page in each page frame, the virtual page number in each page frame, and the referenced (R) and modified (M) bits for each page frame are as shown (the times are in clock ticks from the process start at time 0 to the event).

Virtual Page Number	Frame	Time Loaded	Time Referenced
20	0	60	161
22	1	130	160
24	2	10	162
31	3	20	163

A page fault to virtual page 23 has occurred at time 164. Which **frame** will have its contents replaced for each of the following memory management policies?

- LRU (least recently used)
- Optimal. Use the reference string generated by the following program, considering the following parameters: the system has 256-byte pages. The program is located at address 1020, and its stack pointer is at 8190 (the stack grows toward 0). Each instruction occupies 4 bytes (1 word), and both instruction and data references count in the reference string.

```
Load word from address 6144 into register 0
Push register 0 onto the stack
Call procedure from address 5120, stack the return address
Subtract the immediate constant 16 from the stack pointer
and place it in register RC
Compare the value in RC to the immediate constant 4
Jump if equal to 5152
```

Reference String:

Frame replaced by Optimal:

6. (10 points) In a UNIX File System, the size of a block is 1KB, and each block can hold a total of 128 block addresses. Assume that there are 12 direct block pointers and a singly, doubly and triply indirect pointer in each inode.
    a. Using the inode scheme, what is the maximum size of a file?
    b. Assuming no information other than the file inode is already in main memory, how many disk accesses are required to access the byte in position 13,423,956? (Hint: If you don't have a calculator with you, this information might save you time: $2^{23} < 13{,}423{,}956 < 2^{24}$)

7. (20 points) Sketch rough graphs ("back of napkin") illustrating the relationships among the following quantities. Note any important assumptions underlying your analysis.
    a. Page fault rate as a function of page size. What page sizes are typical for current processors and operating systems?
    b. Disk access latency as a function of rotation speed.
    c. CPU utilization as a function of the degree of multiprogramming. Consider a uni-processor system.
    d. Memory fragmentation as a function of page size. Is that internal or external?