CPE 453
“Operating Systems”

Midterm
January 19, 2004

- Time to complete the exam: 50 minutes
- You will receive partial credit only if you show your work.
- Answer all questions in the space provided. (In case you need more space for a specific problem, use the back of that specific problem and make an appropriate note.)
- Be concise and clearly mark your answer.
- Put your (registered) name on EVERY page.
- GOOD LUCK!

Registered Name (Last, First): ____________________________________________________

Student ID#: ____________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Put your name on every page</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (15 points)

A. (5 points) Fill in each blank with the letter that BEST describes the term. Letters are used no more than once.

1. _____Translation Lookaside Buffer          A. Increases CPU utilization
2. _____Not Frequently Used (NFU)              B. A movie that opens on November 5th.
3. _____Multiprogramming                      C. A small cache of virtual page number to page frame number mappings.
4. _____Lottery Scheduling                    D. A process scheduling algorithm that provides a given proportion to each process.
5. _____Matrix Revolutions                    E. A page replacement algorithm.

B. (10 points) Given:

- 12-bit addressing
- 2KB physical memory
- byte addressable memory
- 256 byte page frames

1. How large is virtual memory?

2. How many page frames are there in main memory?

3. How many entries in a page table?
Problem 2 (20 points)

A. (5 points) List the four necessary conditions for deadlock to occur.

B. (5 points) Give a detailed, specific solution on how you could eliminate one of the above conditions. Explain the disadvantages to your solution.

C. (10 points) Consider the sequence of events below (A, B, C are processes and X, Y are resources). Circle the one that results in deadlock or leave blank if the events do not result in deadlock. Draw the resource allocation graph when deadlock occurs or at the end of events if no deadlock occurs. Assume that processes wait in a FIFO (first in, first out) queue if blocked on a resource.

1. A requests Y
2. B requests Z
3. C requests Z
4. A requests Z
5. C requests Y
6. B releases Z
7. C releases Z
8. A releases Y
9. A releases Z
10. C releases Y
Problem 3 (20 points)

A. (5 points) In an operating system, a process exists in one of three states. Draw the diagram that depicts the transitions between these states for a process.

B. (5 points) List 5 major items found in a process control block (PCB).

C. (10 points) The Apache web server (Linux implementation) forks a child to handle each request for a web page. Argue for using processes over threads in this case.
Problem 4 (20 points)

A. (5 points) A semaphore is a sleep and wakeup solution invented as an alternative solution to busy waiting. In most cases, why is it better to use a semaphore?

B. (15 points) Let's say that you are a programmer in the year 2020 working for the department of motor vehicles. Cars are now controlled by a central server maintained by the DMV. Your new assignment is to control the automobile traffic between Fresno and SLO. The only difficulty is that there is a one lane bridge in between the two towns that cannot be avoided.

Here is the code describing the commute between Fresno and SLO:

```c
void car() {
    driveBeforeBridge();
    enterBridge();
    driveBridge();
    leaveBridge();
    driveAfterBridge();
}
```

Code the implementations of `enterBridge()` and `leaveBridge()` so that:

1. No accidents occur.
2. Up to 5 cars going in the same direction can cross the bridge sequentially at the same time.
3. A car doesn’t wait if no car is using the bridge.

Note: Use semaphores to code your solution. Use C syntax, but assume you have a semaphore type and the `down()` and `up()` operators as described in class and the book.
===== Scratch Work Page for Problem 4 =====
Problem 5 (20 points)

In lab 2, you had to implement a shell that could execute commands and handle pipelines, redirection, sequences, and running in the background. In this problem, implement a simple function that only runs the following command: `ls | sort`. The signature of the function you are to write is: `void executeLsPipeSort()`.

Here are the signatures of some useful functions:

- `pid_t fork()`
- `int pipe(int fd[2])`
- `int dup2(int oldfd, int newfd)`
- `int dup(int oldfd)`
- `int execvp(const char *file, char **argv[])`

Here are some useful tips:

- For this function, you just need to `fork()` once. Unlike our shell from lab 2, no process needs to wait for the commands to finish.
- You can use the macros `READ/WRITE` to index the read/write ends of the pipe file descriptor.
- Use `STDOUT_FILENO` and `STDIN_FILENO` to indicate the file descriptors for stdin and stdout.
- You don’t have to do any error checking in your code.
- To exec `ls`, you should have the following line in your code: `execvp("ls", NULL);`. You should have a similar line to exec `sort`. 
=====

Scratch Work Page for Problem 5 ==== 