Exam 1

Software Systems
Fall 2012

This exam is open book, open notes, open laptop.
For questions with numerical answers, please include appropriate, human-comprehensible units
and indicate your answer clearly.
For short answer questions, your answers should be correct, concise, clear, grammatical and
legible!

1 Virtual memory

1. Imagine a system with 24-bit virtual addresses and 4 KB pages. How many bits are there
   in the page and offset parts of a virtual address?

2. In the same system, each page-table entry takes 32 bits. How big is the page table of a single
   process (if it is using all of its pages)?
3. What is LRU and why might it be a good idea?

4. In Homework 3 we discovered that the stack usually lives in the highest memory addresses and grows down, and the heap lives in low memory addresses and grows up.
   
   On a machine like an Arduino that has a small physical memory (and no virtual memory), would this layout still be a good idea? If so, explain why this layout makes sense. If not, explain what you might do differently.
2 Scheduling

1. Suppose there are two threads ready and available for scheduling: Thread A will take 4 timeslices to complete and Thread B will take 3 timeslices. After 2 timeslices Thread C becomes available and will take 2 timeslices.

If your goal was to minimize the average completion time (time between arrival and completion) in this scenario, would SJF be the optimal scheduling algorithm? Explain why or why not.

2. If, again, your goal was to minimize the average completion time, how would you schedule them? Show which job you would schedule during each timeslice.

   Explain your strategy in general terms and explain why you think it is a good idea.
3 Synchronization

1. Semaphores and monitors provide different ways to manage software synchronization.

   Explain one advantage of monitors over semaphores and give an example of a scenario where using a monitor might be a good choice.

   Explain one advantage of semaphores over monitors and give an example of a scenario where using sempahores might be a good choice.

Note: this question effectively asks for four pieces of information; be sure to cover all four.
2. For Homework 7 I gave you a lock implementation called lock.c that was broken. You might want to review it and remind yourself what the problem was. Suppose I try to fix the problem like this:

```c
void acquire (Lock *lock)
{
    int my_pid = (int) getpid();
    while (lock->value != my_pid) {
        if (lock->value == 0) {
            lock->value = my_pid;
        }
    }
}
```

The idea is that multiple threads might write their process ID numbers into the lock value, but only the thread who writes last will be able to exit the loop and return from `acquire` holding the lock.

Does this implementation work? If not, what sequence of events could cause it to fail?

---

4 C programming

1. On the back of this page write a C function named `replace` that takes three null-terminated strings named `s`, `old` and `new`. If `old` appears as a substring in `s`, it should be replaced with `new`. If it appears more than once, only the first appearance should be replaced. In any case, the return value should be a new, dynamically-allocated string.

You should use the string functions from the C library that are declared in `string.h`. But you should not search for or use someone else's solution to this problem.
2. Explain two reasons why it might be a good idea to split a big program into separate modules, each with its own header file.

3. Find three bugs in the following function. Briefly explain what each problem is and how you would fix it.

```c
/* Takes two strings and returns a heap-allocated string that contains the two strings joined end to end.
For example catenate("bob", "sled") should return "bobsled"
*/

char *catenate(char *s1, char *s2)
{
    char *result;

    int len = strlen(s1) + strlen(s2);

    strcpy(result, s1);

    strcat(result, s2);

    result[len] = '0';

    return *result;
}
```