CSE 421/521 – Operating Systems
Fall 2018 - Homework Assignment #1

The due date is: September 13th, Thursday @1:00pm. (Please do not submit during the class, as it disturbs the lecture. But you can submit at the end of the class). Late submissions are **not** allowed. The solution key will be provided next week after the due date.

Problem 1:

What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

Problem 2:

Give detailed answer to each of the following questions:

(a) What is the difference between fork() and exec() system calls in Unix?

(b) What resources are used when a thread is created? How do these differ from those used when a process is created?

(d) What are context switches used for and what does a typical context switch involve?

Problem 3:

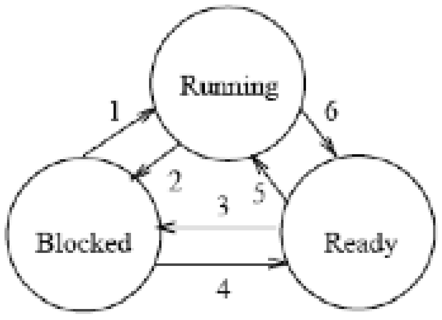
Consider a multiprocessor system and a multithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be more than the number of processors in the system. Discuss the performance implications of the following scenarios.

a. The number of kernel threads allocated to the program is less than the number of processors.

b. The number of kernel threads allocated to the program is equal to the number of processors.

c. The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.

Problem 4:
As shown below, processes can be in one of three states: running, ready and blocked. There are six possible state transitions (labeled 1-6). For each label, indicate whether the transition is valid or not valid. If valid, indicate when the transition is used for a process (i.e. give an example). If the transition is not valid then indicate why.



State transitions:

(a) 1: Blocked to Running
(b) 2: Running to Blocked
(c) 3: Ready to Blocked
(d) 4: Blocked to Ready
(e) 5: Ready to Running
(f) 6: Running to Ready

Problem 5:
In the code below, assume that (i) all fork and execvp statements execute successfully, (ii) the program arguments of execvp do not spawn more processes or print out more characters, and (iii) all pid variables (pid,…,pid6) are initialized to 0.

(a) How many processes will be created by the execution of this code? Show in a "process creation diagram" (like we did in the class) the order in which each processs is created, and the values of pid 1 to pid6 for each process.

(b) What will be the output of each process?

```
void main()
{
        pid1 = fork();
        if (pid1 == 0) {
                pid2 = fork();
                printf(“A”);
        }
        else{
                execvp(…)
        }
        printf(“B”);
        pid3 = fork();
        if (pid4 != 0) {
                printf(“C”);
                execvp(…);
        }
        else{
                if (pid1 != 0){
                        pid5 = fork();
                        execvp(…);
                        printf(“D”);
                }
        }
        if (pid2>0){
                pid6 = fork();
                printf(“E”);
        }
        else{
                printf(“F”)
                execvp(…);
        }
        printf(“G”);
}
```