**Assignment 3**

1. Provide two programming examples in which multithreading does not provide better performance than a single threaded solution. (No code required, just a description)

2. Provide three programming examples in which multithreading provides better performance than a single-threaded solution. (No code required, just a description)

3. What are two differences between user-level threads and kernel-level threads?

4. Describe the actions taken by a kernel to context-switch between kernel level threads.

5. What resources are used when a thread is created? How do they differ from those used when a process is created?

6. The following program uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

#include <pthread.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

int value = 0;

void \*runner(void \*param); /\* the thread \*/

int main(int argc, char \*argv[])

{

pid\_t pid;

pthread\_t tid;

pthread\_attr\_t attr;

pid = fork();

if (pid == 0) { /\* child process \*/

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,runner,NULL);

pthread\_join(tid,NULL);

printf("CHILD: value = %d",value); /\* LINE C \*/

}

else if (pid > 0) { /\* parent process \*/

wait(NULL);

printf("PARENT: value = %d",value); /\* LINE P \*/

}

}

void \*runner(void \*param) {

value = 5;

pthread\_exit(0);

}

7. Consider the following code segment.

pid t pid;

pid = fork();

if (pid == 0) { /\* child process \*/

fork();

thread create( . . .);// This thread no longer creates processes and threads

}

fork();

a. How many unique processes are created?

b. How many unique threads are created?