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

Answer: Because a thread is smaller than a process, thread creation typically uses fewer resources
than process creation. Creating a process requires allocating a process control block (PCB), a rather
large data structure. The PCB includes a memory map, a list of open files, and environment variables.
Allocating and managing the memory map is typically the most time-consuming activity. Creating either
a user thread or a kernel thread involves allocating a small data structure to hold a register set,
stack, and priority.

4.11) Can a multithreaded solution using multiple user-level threads achieve better performance on a
multiprocessor system than on a single-processor system? Explain.

Answer: The many-to-one model does not permit parallelism while the other models do for the reasons
discussed in lecture.

4.13) Is it possible to have concurrency but not parallelism? Explain.

Answer: Yes. Concurrency means that more than one thread is progressing. However, it does not
imply that the threads are running simultaneously. The scheduling of tasks allows for concurrency, but
parallelism is supported only on systems with more than one processing core.

4.19) The program shown in Figure 4.23 uses the Pthreads API. Whatwould be the output from the program at
LINE C and LINE P?

```
#include <pthread.h>
#include <stdio.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);
   }
```

Answer: Output at LINE C is 5. Output at LINE P is 0.

4.20) Consider a multicore system and a multithreaded program written using the many-to-many threading model.

Let the number of user-level threads in the program be greater than the number of processing cores 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 processing cores.

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

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

Answers:

a. When the number of kernel threads is less than the number of processors, then some of the processors will remain idle, since the kernel scheduler maps only kernel threads to processors and not user-level threads to processors.

b. When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors will be utilized simultaneously. However, when a kernel thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor will remain idle.

c. When there are more kernel threads than processors, a blocked kernel thread can be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the

multiprocessor system