Some exercises on Threads

1. Does the multithreaded web server below exhibit task or data parallelism?

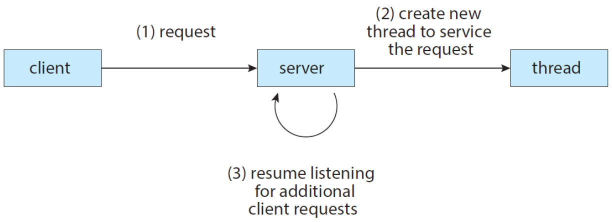
(1) request
(2) create new thread to service the request
client
server
thread
(3) resume listening for additional client requests

Figure 4.2 Multithreaded server architecture.

2. Does every process has a kernel thread?

3. Which of the following components of program state are shared across threads in a multithreaded process? 1- Register values; 2- Heap memory; 3- Static variables; 4- Stack memory

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

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

6. Consider the following code segment:

```
pid t pid;
pid = fork();
if (pid == 0) { /* child process */
fork();
thread create( . . .);
}
fork();
```

   (a) How many unique processes are created?

   (b) How many unique threads are created?

7. Consider the program on the next page, what would be the output from the program at LINE C and LINE P?

8. Provide two programming examples of multithreading giving improved performance over a single-threaded solution.

9. What are two differences in terms of context switch and scheduling between user-level threads and kernel-level threads?

10. Describe some of the actions taken by a kernel to context switch 1- Among threads; 2- Among processes

```
#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);
}
```

11. Describe some of the actions taken by a kernel to context-switch between kernel level threads

12. What are some of the resources used when a thread is created? How do they differ from those used when a process is created?