When to Thread

1. Explain why a web server is a good program to multithread.
2. Explain why a GUI word processing program is a good program to multithread.

Models of Pthread Implementation

1. What are the three models of Pthread implementation, and what are their advantages and disadvantages?

Thread Safety

1. Explain why `strtok` is not thread-safe.

Thread IDs

1. What is wrong with the following code? How can you fix it?
   
   ```c
   pthread_t tid;
   if (tid == pthread_self()) do_something();
   ```

Thread termination

1. What is the effect of the following on the thread with ID `tid`? What about the process?
   
   ```c
   pthread_kill(tid, SIGKILL)
   ```

   Why is it desirable to have a function like `pthread_cancel`?
2. There is one special thread does not require the programmer specifically call `pthread_create`, the main thread, which starts running in the main function. What happens if this thread calls `pthread_exit`? Can another thread cancel the main thread? What would happen? Does this differ from what happens to other threads created by the process using `pthread_create`?
Parameters and Return Values

1. What is wrong with the following code?

```c
void *routine(void *);
pthread_t tid[10];
void *val[10];
int error,i;
for(i=0; i<10; i++) {
    error=pthread_create(tid+i, NULL, routine, &i );
    if (error) sys_error("Could not create thread\n");
}
for(i=0; i<10; i++) {
    error=pthread_join(tid[i], val+i);
    if (error) sys_error("Could not join thread\n");
do_work(val[i]);
}
exit(0);
```

2. Suppose the following code is part of routine

```c
void *routine(void *arg) {
    char *s1 = malloc(10*sizeof(char));
    char s2[10];
    static char s3[10];
do_work();
return NULL;
}
```

Which of the following are safe replacements for NULL as the parameter to return?

(a) s1
(b) s2
(c) s3
(d) "Hello World!"

Threads and signals

1. Can you have one thread fork a child and another thread wait on that child? (Hint: wait really means "wait until you recieve a SIGCHLD")
Molay, Chapter 14

The following questions come from Chapter 14 of Molay’s Molay, *Understanding Unix/Linux Programming*

1. On page 473 Molay gives some code for creating a detached thread (this is a thread which relinquishes its resources on termination without having to be joined.) Give another way to detach a thread. When would one method be preferrable to the other method?

2. The following code is extracted from the program *twordcount4.c* on pages 469-70 of Molay (in the function *count_words*)

   ```c
   pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
   pthread_cond_t flag = PTHREAD_COND_INITIALIZER;
   
   pthread_mutex_lock(&lock);
   if (mailbox != NULL)
     pthread_cond_wait(&flag, &lock);
   mailbox = args;
   pthread_cond_signal(&flag);
   pthread_mutex_lock(&lock);
   
   The code is designed to ensure that `mailbox` is NULL before assigning it a value: the predicate `mailbox == NULL` is true. But this is bad code. Read the POSIX specification for `pthread_cond_wait` and `pthread_cond_signal` and fix the code so that `mailbox` is assigned `args only` when `mailbox == NULL` is true.