```
/* thread_one runs in this function */
void *do_work_one(void *param)
{
   pthread_mutex_lock(&first_mutex);
   pthread_mutex_lock(&second_mutex);
   /**
    * Do some work
    */
   pthread_mutex_unlock(&second_mutex);
   pthread_mutex_unlock(&first_mutex);

   pthread_exit(0);
}

/* thread_two runs in this function */
void *do_work_two(void *param)
{
   pthread_mutex_lock(&second_mutex);
   pthread_mutex_lock(&first_mutex);
   /**
    * Do some work
    */
   pthread_mutex_unlock(&first_mutex);
   pthread_mutex_unlock(&second_mutex);

   pthread_exit(0);
}
```

Figure 7.4 Deadlock example.

7.3 The program example shown in Figure 7.4 doesn't always lead to deadlock. Describe what role the CPU scheduler plays and how it can contribute to deadlock in this program.

7.6 In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, and new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

a. Increase ***Available*** (new resources added).

b. Decrease ***Available*** (resource permanently removed from system).

c. Increase ***Max*** for one process (the process needs or wants more resources than allowed).

d. Decrease ***Max*** for one process (the process decides it does not need that many resources).

e. Increase the number of processes.

f. Decrease the number of processes.

7.12 Consider the following snapshot of a system:

	Allocation	*Max*
	A B C D	*A B C D*
P_0	3 0 1 4	5 1 1 7
P_1	2 2 1 0	3 2 1 1
P_2	3 1 2 1	3 3 2 1
P_3	0 5 1 0	4 6 1 2
P_4	4 2 1 2	6 3 2 5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.

a. ***Available*** $= (0, 3, 0, 1)$

b. ***Available*** $= (1, 0, 0, 2)$