Homework 10

Problem 1

A. The program in the figure below has a bug. The thread is supposed to sleep for 1 second and then print a string. However, when we run it on our system, nothing prints. Why?

B. You can fix this bug by replacing the exit function in line 9 with a Pthread function calls. What's it?

```
1   #include “csapp.h”
2   void *thread(void *vargp);
3
4   int main()
5   {
6       pthread_t tid;
7
8       Pthread_create(&tid, NULL, thread, NULL);
9       exit(0);
10  }
11
12   /* Thread routine */
13   void *thread(void *vargp)
14   {
15       Sleep(1);
16       printf(“Hello, world!\n”);
17       return NULL;
18  }
```

Solution

A. The problem is that the main thread calls exit without waiting for the peer thread to terminate. The exit call terminates the entire process, including any threads that happen to be running. So the peer thread is being killed before it has a chance to print its output string.

B. We can fix the bug by replacing the exit function with Pthread_exit, which waits for outstanding threads to terminate before it terminates the process.

Problem 2

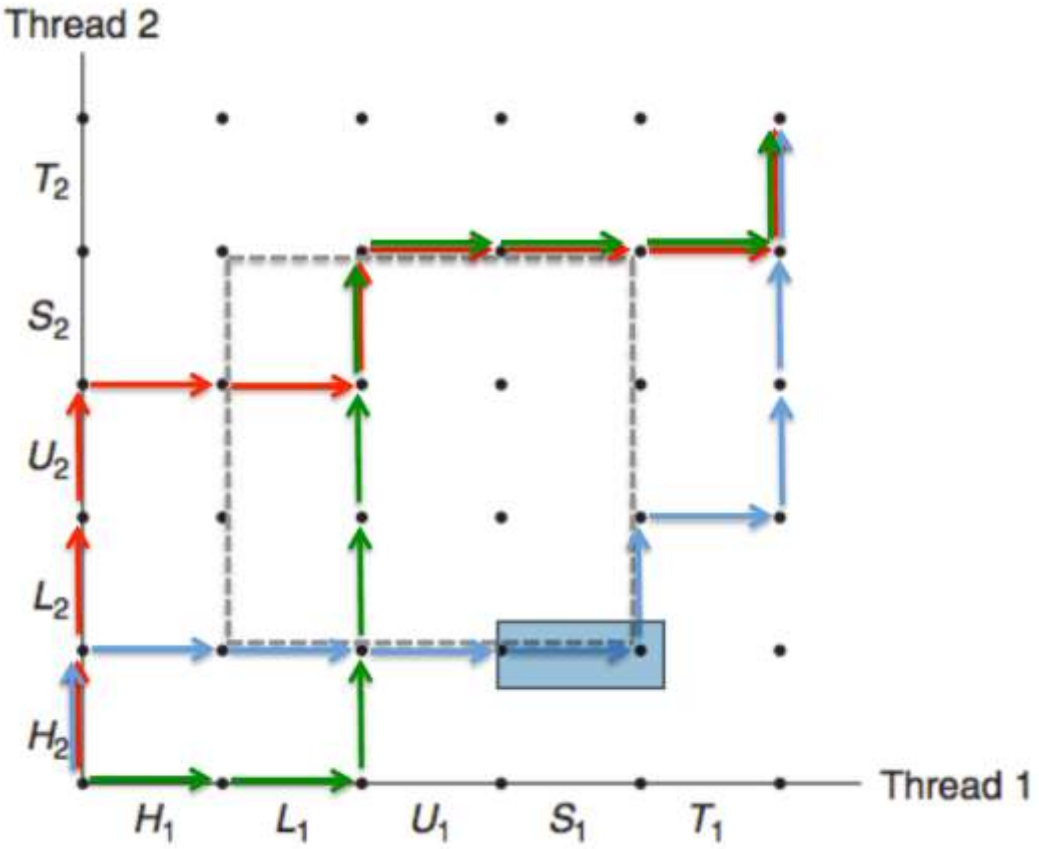
Using the progress graph in Figure 12.21, classify the following trajectories as either
safe or unsafe. And draw them out.

A. H2, L2, U2, H1, L1, S2, U1, S1, T1, T2

B. H2, H1, L1, U1, S1, L2, T1, U2, S2, T2

C. H1, L1, H2, L2, U2, S2, U1, S1, T1, T2

Solution



A: unsafe, red line

B: safe, blue line

C: unsafe, green line

Problem 3

1. Starvation (in relation to threads) refers to:
   (a) A thread waiting for a lock indefinitely.
   (b) A semaphore that gets locked but the thread never unlocks it after use.
   (c) A thread is spawned but never joins the main thread when finished.

(d) A process fails to spawn a new thread because it's hit the maximum number of threads allowed.

Solution

A

Problem 4

Can the following program deadlock? Why or why not?

Initially:a=1,b=1,c=1.

Thread 1:	Thread 2:
P(a);	P(c);
P(b);	P(b);
V(b);	V(b);
P(c);	V(c);
V(c);	
V(a);	

Solution

Thread 1 holds mutex pairs (a, b) and (a, c) simultaneously, but not mutex pair (b,c), while Thread 2 holdsmutex pair (c,b) simultaneously, not the other two. Since the sets are disjoint, there is no deadlock potential,even though Thread 2 locks its mutexes in the wrong order.

Problem 5

What are the advantages of using virtual memory instead of physical memory?

Solution

Isolation, larger memory space, etc.