ICS'2 Homework 12

Problem 1

```
int main(int argc, char **argv){
    int listenfd, connfd, port, clientlen;
    struct sockaddr_in clientaddr;
    pthread_t tid;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <port>\n", argv[0]);
        exit(0);
    }
    listenfd = open_listenfd(port);
    while (1) {
        clientlen = sizeof(clientaddr);
        connfd = Accept(listenfd,
             (SA *)&clientaddr, &clientlen);
        Pthread_create(&tid, NULL, thread, &connfd);
    }
}

/* thread routine */
void *thread(void *vargp)
{
    int connfd = *((int *)vargp);
    echo_r(connfd); /* reentrant version of echo() */
    Close(connfd);
    return NULL;
}
```

1. Read the code carefully and tell what would happen to 'connfd' in the threads created by the main thread? Why?

It's undefined in the threads created by the main thread because 'vargp' points to somewhere in the main thread's stack, which is varying all the same time. Should use heap variable or static variable instead.

2. We sometimes find memory leaks in this program. Can you give the reason and the solution it?

Because the threads are not detached from the main thread. Add 'Pthread_detach(pthread_self());' to the thread routine.

Problem 2

Compare the three implementations of our echo server below:

(1) process-based

(2) thread-based

(3) I/O multiplexing

You may consider aspects such as efficiency, resource sharing, overhead, code complexity.

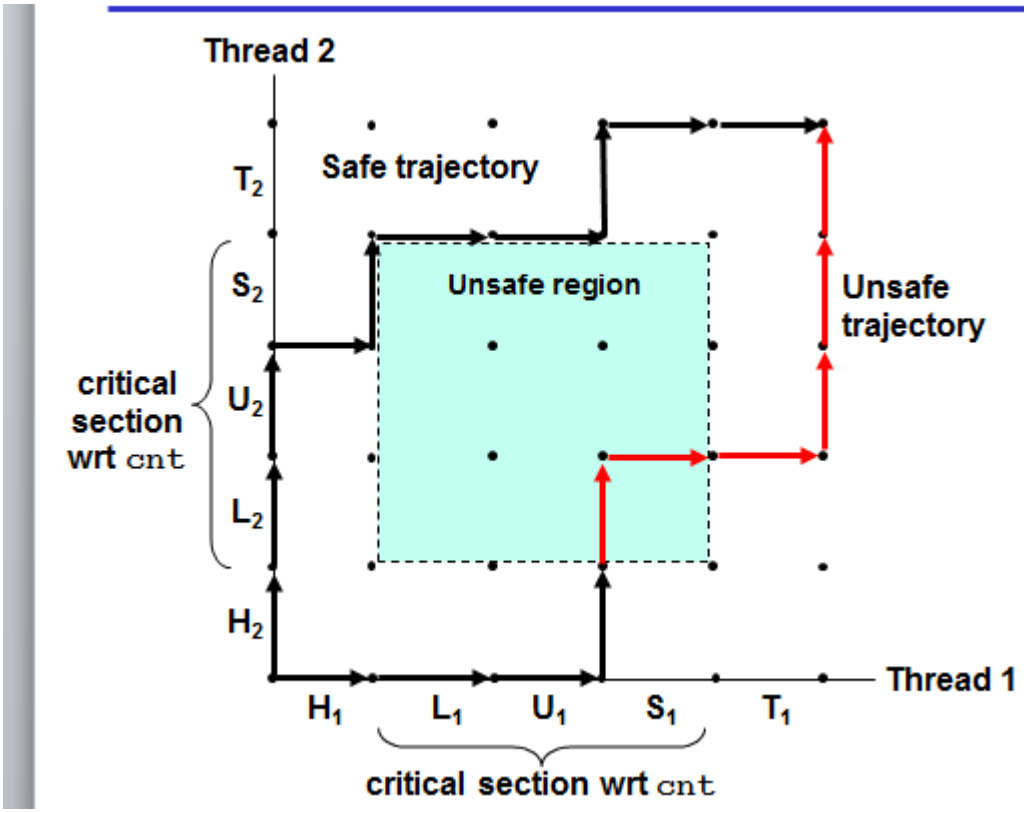
Both process-based and thread-based versions have much higher efficiency and lower code complexity than I/O multiplexing. Thread-based and I/O multiplex versions make resource sharing much easier than process-based one. Thread-based has lower overhead than process-based and I/O multiplexing has almost no overhead.

Problem 3

```
int n1;
// thread routine.
void *t(void *argp)
{
    static int n2;
    int i;
    for (i = 0; i < 100; ++i)
        n1++;
    return NULL;
}
```

Suppose there are two threads that are concurrently running (see the above thread routine):

1. How many instances do 'n1', 'n2' and 'i' have respectively?
   n1 and n2 have only one instance. i has two instances.
2. Draw the process graph, specify the unsafe region and give a safe trajectory and an unsafe one on the graph.



Problem 4

Suppose we have a restaurant that can serve up to 1000 clients (implemented as threads) at the same time.

1. The semaphore is named 'sem'. Write a statement to initialize it using 'Sem_init()'.
   Sem_init(&sem, 0, 1000);
2. The only thing a client will do is eating, which is indicated by the function 'eat()'. Write the thread routine (you have to ensure thread safety).

```
void *client(void *argp)
{
    P(&sem);
    eat();
   V(&sem);
   return NULL;
}
```