13. Dining Philosopher's Problem

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
pthread_t *philosophers;
pthread_mutex_t *forks;
int philosophers_count;
void eat(int i){
   printf("Philosopher %d is eating\n",i+1);
 sleep(1 + rand()%10);
}

void* philosopher(void* args){
        int i = 0,first,second;
        while(!pthread_equal(*(philosophers+i),pthread_self()) && i < philosophers_count){
                i++;
        }

        while(1){

                printf("Philosopher %d is thinking\n",i+1);

                sleep(1 + rand()%10);

                first = i;
                second = (i+1)%philosophers_count;
        pthread_mutex_lock(forks + (first>second?second:first));
```

```
                pthread_mutex_lock(forks + (first<second?second:first));
                eat(i);
                pthread_mutex_unlock(forks+first);
                pthread_mutex_unlock(forks+second);
        }
        return NULL;
}
int main(void){
        int i,err;

        srand(time(NULL));

        printf("Enter number of philosophers:");
        scanf("%d",&philosophers_count);
        philosophers = (pthread_t*) malloc(philosophers_count*sizeof(pthread_t));
        forks = (pthread_mutex_t*) malloc(philosophers_count*sizeof(pthread_mutex_t));

        for(i=0;i<philosophers_count;++i)
                if(pthread_mutex_init(forks+i,NULL) != 0){
                        printf("Failed initializing fork %d\n",i+1);
                        return 1;
                }

        for(i=0;i<philosophers_count;++i){
                err = pthread_create(philosophers+i,NULL,&philosopher,NULL);

                if(err != 0){
                        printf("Error creating philosopher: %s\n",strerror(err));
                }else{
                        printf("Successfully created philosopher %d\n",i+1);
                }
```

```
        }

        for(i=0;i<philosophers_count;++i)
                pthread_join(*(philosophers+i),NULL);

        free(philosophers);
        free(forks);

        return 0;
}
```