Project 2 Report

姓名：吴本利 学号：522031910763

Contents

1 StoogeFarmers

1.1 Description

The program simulates three people planting trees. They have different task and there are some constraints.

```
./lcm
```

1.2 implementation

1.2.1 Semaphores

I define four semaphores.

Dig_hole: when it >0, Larry can dig holes. It can be the number of holes that can be dug by Larry. So I initialize it to MAX.

Plant_hole: when it >0, Moe can plant tree.

Fill_hole: when it > 0, Curly can fill holes.

Shovel : when it = 1, Larry and Curly can get the shovel to dig holes or fill holes.

```
// semaphores
//定义信号量
sem_t shovel;
sem_t dig_hole; // Larry can dig holes
sem_t plant_hole;//Moe can plant
sem_t fill_hole;//Curly can fill holes
```

variables

```
int dig_num = 0;
int plant_num = 0;
int fill_num = 0;
```

1.2.2 Larry

It's a function that Larry will call.

```
void *larry()
{
    // some code goes here
    int id = 0;
    while (1) {
        sem_wait(&dig_hole);
        sem_wait(&shovel);
        get_shovel(LARRY);

        dig(LARRY, ++id);
        dig_num++;
        drop_shovel(LARRY);
        sem_post(&shovel);
        sem_post(&plant_hole);
        if (dig_num == N)
            pthread_exit(0);
        usleep(rand() % 1000);
    }
}
```

1.2.3 Moe

It's a function that Moe will call.

```
void *moe() {
    // some code goes here
    int id = 0;
    while (1) {
        sem_wait(&plant_hole);
        plant(MOE, ++id);
        plant_num++;
        sem_post(&fill_hole);
        if (plant_num == N)
            pthread_exit(0);
        usleep(rand()%1000);
    }
}
```

1.2.4 Curly

It's a function that Curly will call.

```
void *curly() {
```

```
    // some code goes here
    int id = 0;
    while (1) {
        sem_wait(&fill_hole);
        sem_wait(&shovel);
        get_shovel(CURLY);
        fill(CURLY, ++id);
        fill_num++;
        drop_shovel(CURLY);
        sem_post(&shovel);
        sem_post(&dig_hole);
        if(fill_num==N)pthread_exit(0);
        usleep(rand() % 1000);
    }
}
```

2 Party

2.1 Description

The program simulates that students have party in a room. There is a dean who will search the room and break up the party. There are also some constraints.

```
./party
```

2.2 Implementation

2.2.1 Semaphores and variables

Mutex will protect different threads from change the student_num at the same time.

I define a variable named leave. It's because there will be some time delay between the dean entrance and breaking up the party. I don't want that the students leave before breaking up the party

```
sem_t mutex;

int student_num = 0;//在房间里的学生数量
int flag = 0;//标记老师是否在房间里
int leave = 1;//学生能不能离开
```

2.2.2 Dean

It's a function that Dean will call.

```
void *dean() {
    for (int i = 0; i < N; i++) {
        usleep((rand() % 500 + 200) * 1000);
        // some code goes here
```

```
        while(1){
        if(student_num==0||student_num>10){
        flag = 1;

        leave = 0;
        dean_enter();
        if (student_num == 0)
            search();
        else if(student_num>10)
        {
            breakup();
        }

        leave = 1;

        while(student_num!=0){
        }
        dean_leave();
        flag = 0;
        break;
        }
    }

    }
    pthread_exit(0);
}
```

2.2.3 Student

It's a function that students will call.

```
void *student(void *arg) {
    int id = *(int *)arg;
    // some code goes here
    while(1){
        if(flag==0){

            sem_wait(&mutex);
            student_num++;
            sem_post(&mutex);
            student_enter(id);

            party(id);
```

```
            while(leave==0){}
            student_leave(id);

            sem_wait(&mutex);
            student_num--;
            sem_post(&mutex);
            break;
        }
    }

    pthread_exit(0);
}
```