CS356-Project1

- ***Problem1***

  > To use my *pstree* module, you should go to directory *Problem1* in Linux terminal and type **make** command. Make sure that you have installed *sdk* and *ndk* correctly. Then you will get a file named *pstree.ko*, which you should push to your virtual device. After that you can use **insmod** to install this module and use **lsmod** to check.

- ***Problem2***

  > To test my *pstree* module, you should go to directory *Problem2* and type **ndk-build** command in Linux terminal. Make sure that you have installed *sdk* and *ndk* correctly. Then you will get a file named *test_pstreeARM*, push it to your virtual device and run it to test my *pstree* module in problem1.

- ***Problem3***

  > The way to compile and use the source code in problem3 is similar to problem2. And this source code will just create a new process and run the test program write in problem2 in child process. Make sure that you put *test_pstreeARM* and *parent_childARM* under the same directory.

- ***Problem4***

  > The way to compile and use the source code in problem3 is similar to problem2. Here are pseudo codes for this problem.

```
CASHIER{
    while(true)
    {
        wait(mutex1)
        if(remain_customer_count<=0){
            signal(mutex1);
            break;
        }
        else{
            remain_customer_count--;
            signal(mutex1);
        }
        wait(customer);
        //take order....
        wait(burger_on_rack);
        signal(empty_slots_on_rack);
        signal(cashier);
        //get burger for customer;
    }
}
```

```
CUSTOMER{
    //sleep for random time
    // customer coming.....
    signal(customer);
    wait(cashier);
}
```

```
COOK{
    //determine how long costs to make a burger for this cook
    while(true){
        //sleep for making burger time;
        wait(mutex2);
        if(total_burger>=customer_size){
            signal(mutex2);
            break;
        }
        else{
            total_burger++;
            signal(mutex2);
        }
        //make burger.....
        wait(empty_slots_on_rack);
        signal(burgers_on_rack);
    }
}
```

I also provide the pseudo code in the *BurgerBuddiesProblem.c* . For customer, each of them will sleep for random time and wake up to get his burger and exit. What's more, you can change *CUSTOMER_TIME* in line 8 to change the range of random sleep time. For cook, each of them will have a fixed time to make a burger, just like different cooks vary in skill so their speed to make a burger is not the same. When cooks make burgers' number is equal to customers' size, cooks will exit. Similarly, you can change *BURGER_TIME* in line 9 to change the range of random time to make a burger. For cashier, when all the customers are served, they will exit.

To run this program, you should use command like this format:

```
./BBC ${COOK_SIZE} ${CASHIER_SIZE} ${CUSTOMER_SIZE} ${RACK_SIZE}
```