Operating system Lab 6 report

Cigarette Smoker's synchronize problem

Yi Lin

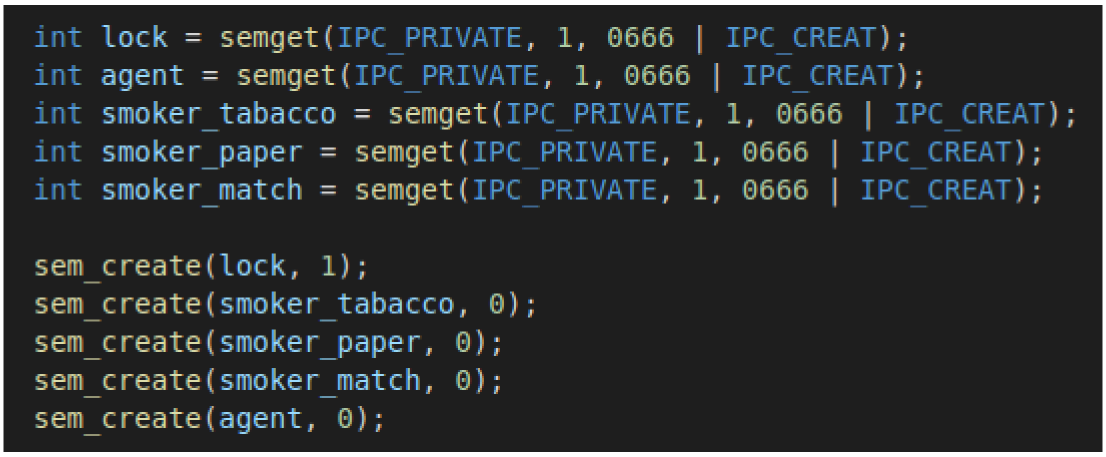
Class 34293

Table of contents

1. Using process and semaphore

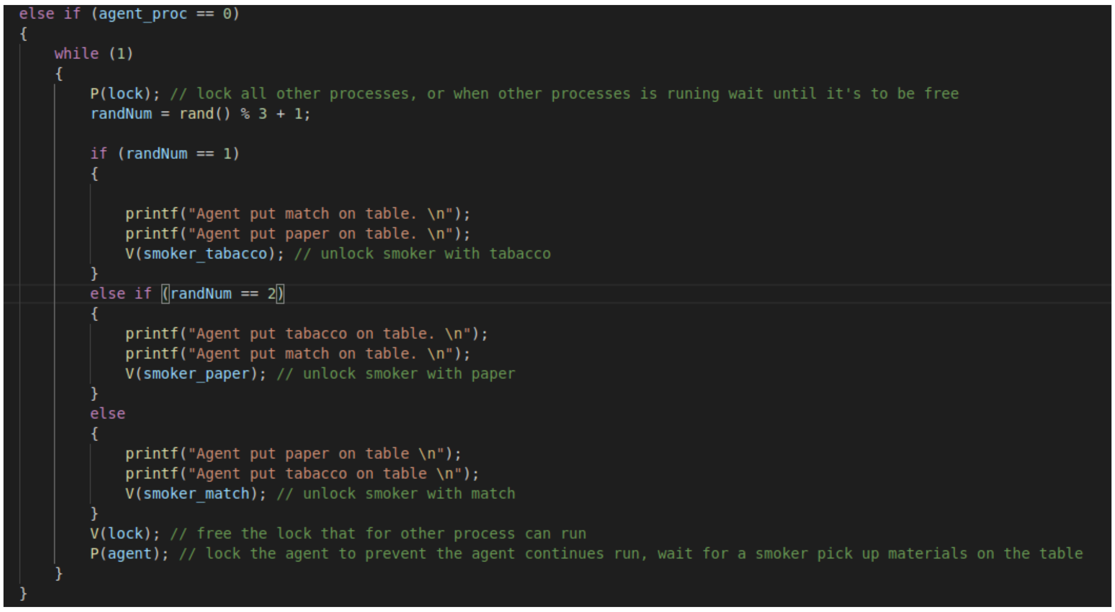
The first approach is using semaphore based on multi-processes.

First of all create 5 semaphores, semaphore lock to be a mutex that prevent when one process is running, any other process runs the code. It initialize to be 1 because it first needs to allow a process to run. The other semaphores are used to control the specific process to run, in this case, agent control the agent process, smoker_tabacco control the smoker process with tabacco, etc.



```
int lock = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);
int agent = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);
int smoker_tabacco = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);
int smoker_paper = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);
int smoker_match = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);

sem_create(lock, 1);
sem_create(smoker_tabacco, 0);
sem_create(smoker_paper, 0);
sem_create(smoker_match, 0);
sem_create(agent, 0);
```

Agent process:



```c
else if (agent_proc == 0)
{
    while (1)
    {
        P(lock); // lock all other processes, or when other processes is runing wait until it's to be free
        randNum = rand() % 3 + 1;

        if (randNum == 1)
        {

            printf("Agent put match on table. \n");
            printf("Agent put paper on table. \n");
            V(smoker_tabacco); // unlock smoker with tabacco
        }
        else if (randNum == 2)
        {
            printf("Agent put tabacco on table. \n");
            printf("Agent put match on table. \n");
            V(smoker_paper); // unlock smoker with paper
        }
        else
        {
            printf("Agent put paper on table \n");
            printf("Agent put tabacco on table \n");
            V(smoker_match); // unlock smoker with match
        }
        V(lock); // free the lock that for other process can run
        P(agent); // lock the agent to prevent the agent continues run, wait for a smoker pick up materials on the table
    }
}
```

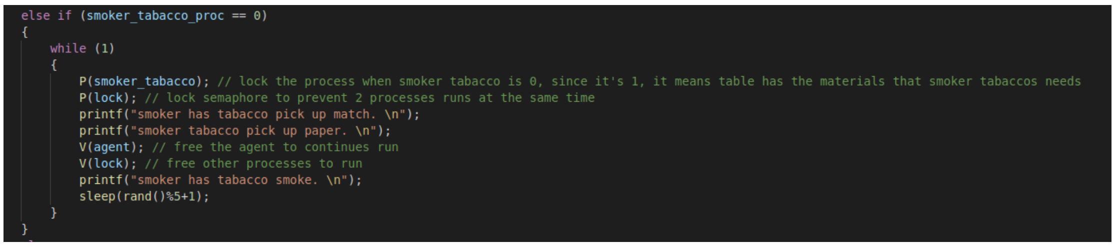
The agent process first P(lock) to lock all other processes, and also prevent it runs when other processes is running.

Then it will randomly put 2 materials on the table, then it will unlock the semaphore for that process.

Then it will free the lock semaphore V(lock) that other smoker prosses can run, and P(agent) to lock the agent to prevent the agent continues run, and wait for a smoker pick materials on the table.

Smoker processes:

I will only show the smoker tobacco case, since the other 2 are identical



```c
else if (smoker_tabacco_proc == 0)
{
    while (1)
    {
        P(smoker_tabacco); // lock the process when smoker tabacco is 0, since it's 1, it means table has the materials that smoker tabaccos needs
        P(lock); // lock semaphore to prevent 2 processes runs at the same time
        printf("smoker has tabacco pick up match. \n");
        printf("smoker tabacco pick up paper. \n");
        V(agent); // free the agent to continues run
        V(lock); // free other processes to run
        printf("smoker has tabacco smoke. \n");
        sleep(rand()%5+1);
    }
}
```

Here in a smoker process, it will first P(smoker_tabacco) since when the smoker tabacco is 0, and it also means that the table does not has all the materials that smoker tabacco needs, and it also prevent smoker process run 2 times consecutively. And when smoker tabacco is 1, then that means table has the materials that smoker tabacco needs.

Then it will P(lock) to prevent 2 processes runs at the same time

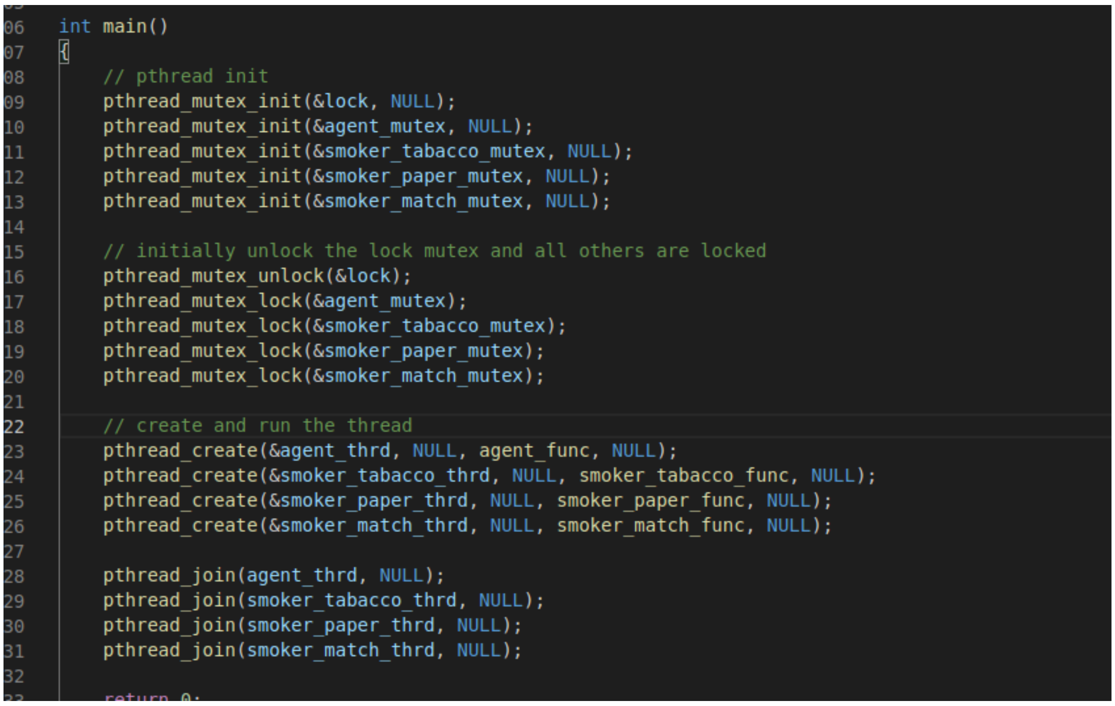
After smoker pick up the materials, then it will free the agent to continues runs and free other processes to run.

Then the smoker can smoke for a random time.

2. Using thread and pthread mutex

Compile using: gcc -Wall -pthread -lpthread thread.c -o thread

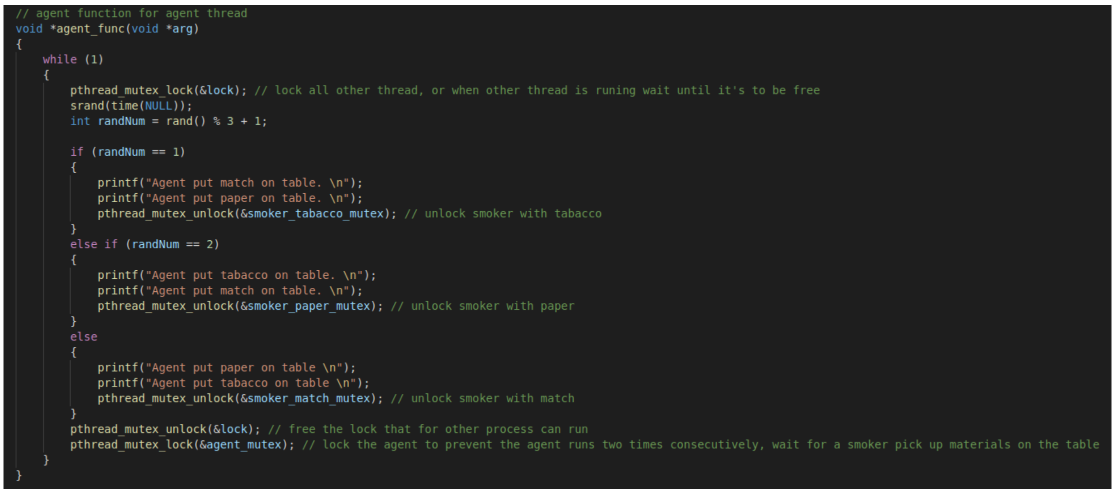
For mutex initialization, it’s very similar to process case, only mutex lock to prevent when one thread is running, any other thread runs the code. It initializes to be unlock because it first needs to allow a thread to run. The other mutexes are used to control the specific thread to run, in this case, agent_mutex control the agent thread, smoker_tabacco_mutex control the smoker thread with tabacco, etc.



```c
int main()
{
    // pthread init
    pthread_mutex_init(&lock, NULL);
    pthread_mutex_init(&agent_mutex, NULL);
    pthread_mutex_init(&smoker_tabacco_mutex, NULL);
    pthread_mutex_init(&smoker_paper_mutex, NULL);
    pthread_mutex_init(&smoker_match_mutex, NULL);

    // initially unlock the lock mutex and all others are locked
    pthread_mutex_unlock(&lock);
    pthread_mutex_lock(&agent_mutex);
    pthread_mutex_lock(&smoker_tabacco_mutex);
    pthread_mutex_lock(&smoker_paper_mutex);
    pthread_mutex_lock(&smoker_match_mutex);

    // create and run the thread
    pthread_create(&agent_thrd, NULL, agent_func, NULL);
    pthread_create(&smoker_tabacco_thrd, NULL, smoker_tabacco_func, NULL);
    pthread_create(&smoker_paper_thrd, NULL, smoker_paper_func, NULL);
    pthread_create(&smoker_match_thrd, NULL, smoker_match_func, NULL);

    pthread_join(agent_thrd, NULL);
    pthread_join(smoker_tabacco_thrd, NULL);
    pthread_join(smoker_paper_thrd, NULL);
    pthread_join(smoker_match_thrd, NULL);

    return 0;
```

The thread approach is pretty similar to the processes approach, but instead of using semaphore, we using pthread_mutex to control the thread, which is in pthread.h library.

Agent thread:



```c
// agent function for agent thread
void *agent_func(void *arg)
{
    while (1)
    {
        pthread_mutex_lock(&lock); // lock all other thread, or when other thread is runing wait until it's to be free
        srand(time(NULL));
        int randNum = rand() % 3 + 1;

        if (randNum == 1)
        {
            printf("Agent put match on table. \n");
            printf("Agent put paper on table. \n");
            pthread_mutex_unlock(&smoker_tabacco_mutex); // unlock smoker with tabacco
        }
        else if (randNum == 2)
        {
            printf("Agent put tabacco on table. \n");
            printf("Agent put match on table. \n");
            pthread_mutex_unlock(&smoker_paper_mutex); // unlock smoker with paper
        }
        else
        {
            printf("Agent put paper on table \n");
            printf("Agent put tabacco on table \n");
            pthread_mutex_unlock(&smoker_match_mutex); // unlock smoker with match
        }
        pthread_mutex_unlock(&lock); // free the lock that for other process can run
        pthread_mutex_lock(&agent_mutex); // lock the agent to prevent the agent runs two times consecutively, wait for a smoker pick up materials on the table
    }
}
```

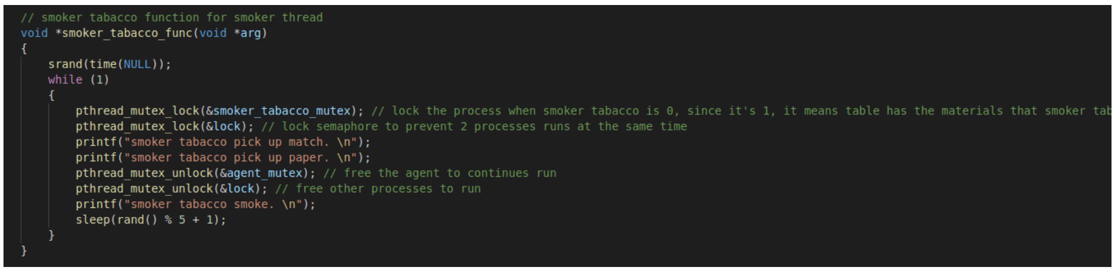
For the agent thread it first lock the lock mutex to lock all other processes, and also prevent it runs when other processes is running.

Then it will randomly put 2 materials on the table, then it will unlock the mutex for that thread.

Then it will unlock the lock mutex that other smoker thread can run, and lock the agent mutex to lock the agent thread to prevent the agent run 2 times consecutively, and wait for a smoker pick materials on the table.

Smoker thread:

I will only show the smoker tobacco case, since the other 2 are identical



```c
// smoker tabacco function for smoker thread
void *smoker_tabacco_func(void *arg)
{
    srand(time(NULL));
    while (1)
    {
        pthread_mutex_lock(&smoker_tabacco_mutex); // lock the process when smoker tabacco is 0, since it's 1, it means table has the materials that smoker tab
        pthread_mutex_lock(&lock); // lock semaphore to prevent 2 processes runs at the same time
        printf("smoker tabacco pick up match. \n");
        printf("smoker tabacco pick up paper. \n");
        pthread_mutex_unlock(&agent_mutex); // free the agent to continues run
        pthread_mutex_unlock(&lock); // free other processes to run
        printf("smoker tabacco smoke. \n");
        sleep(rand() % 5 + 1);
    }
}
```

Here in a smoker thread, it will first lock the smoker tabacco mutex, it prevent smoker thread run 2 times consecutively, and it controls only when the table has the material that the current smoker needs, it will run.

Then it will lock the lock mutex to prevent 2 processes runs at the same time

After smoker pick up the materials, then it will unlock the agent mutex to let the agent thread continues runs and free other thread to run.

Then the smoker can smoke for a random time.