CSE344 – System Programming - Homework #4 Report

Fun with threads and semaphores

Student Name: Ozan Şelte

Student Number: 161044061

In this homework to be POSIX compatible with the latest standart I decided to define "_POSIX_C_SOURCE" as "200809", "_FILE_OFFSET" as "64" and "_GNU_SOURCE".

```c
#define INGR_COUNT (5)

enum Ingredient {
    GULLAC=0,
    INGR_M=1,
    INGR_F=2,
    INGR_W=3,
    INGR_S=4
};
```

```c
struct ChefSpecs {
    enum Ingredient a;
    enum Ingredient b;
    size_t id;
};

union semun {
    int val;
    struct semid_ds *buf;
    unsigned short *array;
    struct seminfo *__buf;
};
```

Problem #1:

Chefs must be aware of which ingredients are came to the street from wholesaler.

Solution #1:

Every chef waits their own 2 ingredients with System V semaphores concurrently.

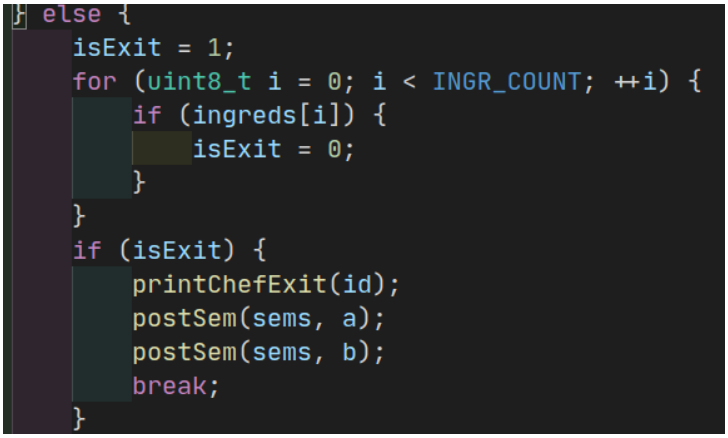
```c
printWholesalerExit();
memset(ingreds, '\0', INGR_COUNT*sizeof(uint8_t));
for (size_t i = 0; i < INGR_COUNT; ++i) {
    postSem(sems, i);
}
```

Problem #2:

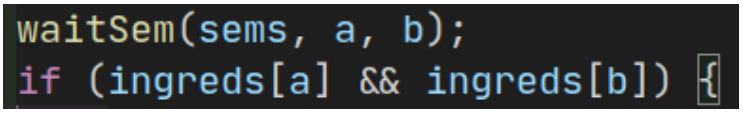
When there is not any new line for the wholesaler, it have to alert all the chefs. It is not a big problem but wholesaler should not know the cheft's count and use should use only semaphores.

Solution #2:

The wholesaler posts all the ingredient semaphores but puts nothing to the street. If a chef wakes up from wait and cannot find the ingredients, it understand that, it is the exit signal. When exit, chef posts its own waiting ingredients semaphores. This way every chef can exit gracefully.



```
} else {
    isExit = 1;
    for (uint8_t i = 0; i < INGR_COUNT; ++i) {
        if (ingreds[i]) {
            isExit = 0;
        }
    }
    if (isExit) {
        printChefExit(id);
        postSem(sems, a);
        postSem(sems, b);
        break;
    }
```



```
waitSem(sems, a, b);
if (ingreds[a] && ingreds[b]) {
```