GEBZE TECHNICAL UNIVERSITY

CSE 344 SYSTEMS PROGRAMMING

HW3 REPORT

ESRA NUR ARICAN

161044028

OBJECTIVE

We need to prepared a small bakers simulation for this homework. There are 4 ingredients needed to prepared güllaç. These ingredients are:

-M (milk)

-W (walnuts)

-S (sugar)

-F (flour)

We have to write two version of this program, one is uses named semaphores and the other one uses unnamed semaphores for synchronization.

Also there should be one wholesaler process and 6 chef processes.

SOLUTION

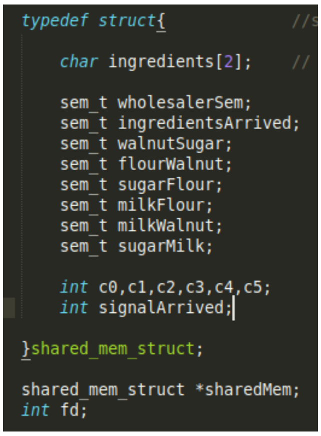
Unnamed Semaphore Version

To use unnamed semaphores in this version, first i created a shared memory segment. Because to provide synchronization between multiple processes with unnamed semaphores, we have to place our semaphores inside a shared memory that every process can reach.

My shared memory struct consists necesarry semaphores, a flag for signals and a char array data structure for keeping ingredients. There are 8 unnamed semaphores. Wholesalersem is for wholesaler, ingredientsArrived semaphore is used for orginizing new arrived ingredients.

Other semaphores are for created every lack of ingredients. To clarify this; if a chef has endless supply of milk and flour, he waits walnuts and sugar to arrive. So walnutSugar semaphore is used here. Other semaphores (flourWalnut, milkFlour etc.) are also created for same purpose.

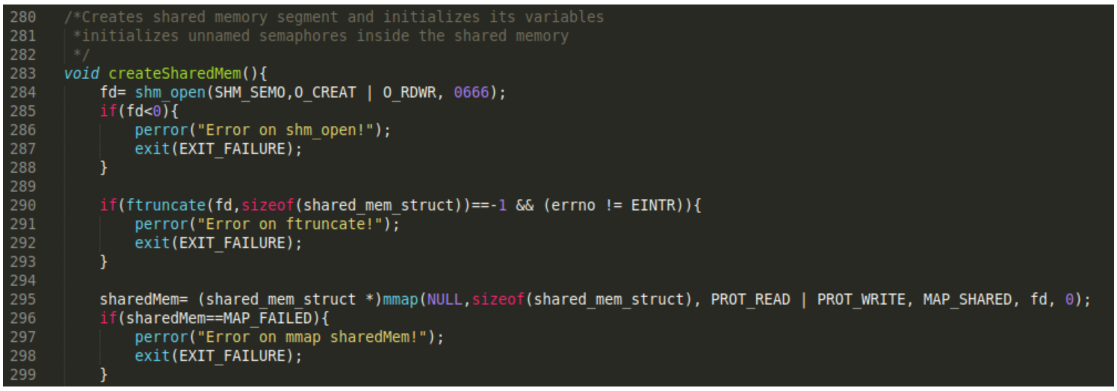
After defining the shared memory struct, I declared a sharedMem variable type of the struct in global area to make it accessible from every processes. Also a global file descriptor is defined for using opening the shared memory later.



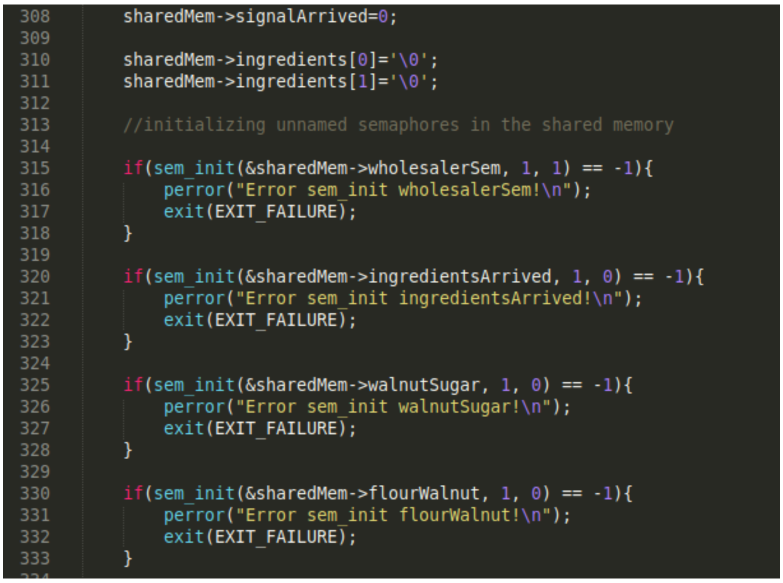
```
typedef struct{          //s

    char ingredients[2];    //

    sem_t wholesalerSem;
    sem_t ingredientsArrived;
    sem_t walnutSugar;
    sem_t flourWalnut;
    sem_t sugarFlour;
    sem_t milkFlour;
    sem_t milkWalnut;
    sem_t sugarMilk;

    int c0,c1,c2,c3,c4,c5;
    int signalArrived;

}shared_mem_struct;

shared_mem_struct *sharedMem;
int fd;
```

In main function, program first takes commandline arguments using getopt, assigns them into variables and controls given commandline arguments. If user tries to run program with invalid arguments, program prints the correct usage.

The shared memory created with a function named **createSharedMemory().**This function creates and initializes shared memory with shm_open(), mmap(), ftruncate(). Also inside this function all unnamed semaphores are initialized to their starting values.

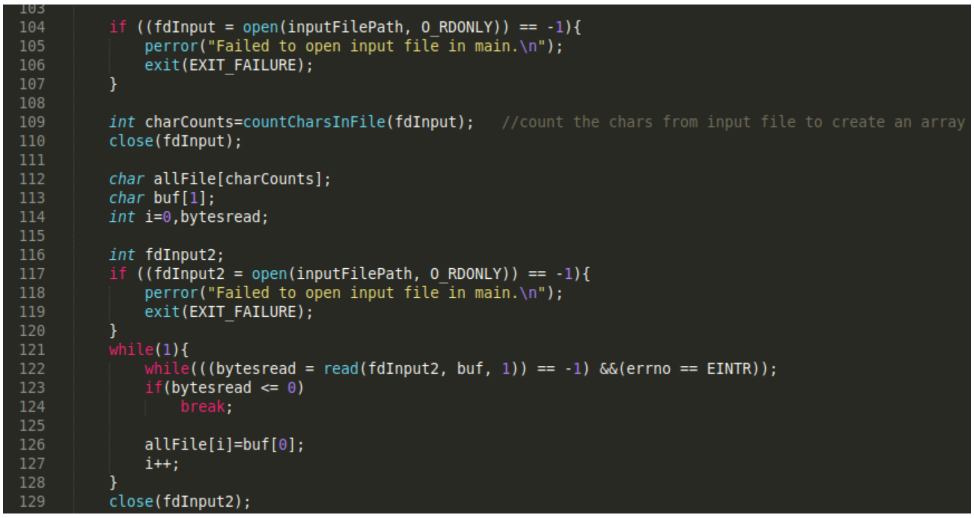


```
/*Creates shared memory segment and initializes its variables
 *initializes unnamed semaphores inside the shared memory
 */
void createSharedMem(){
    fd= shm_open(SHM_SEMO,O_CREAT | O_RDWR, 0666);
    if(fd<0){
        perror("Error on shm_open!");
        exit(EXIT_FAILURE);
    }

    if(ftruncate(fd,sizeof(shared_mem_struct))==-1 && (errno != EINTR)){
        perror("Error on ftruncate!");
        exit(EXIT_FAILURE);
    }

    sharedMem= (shared_mem_struct *)mmap(NULL,sizeof(shared_mem_struct), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if(sharedMem==MAP_FAILED){
        perror("Error on mmap sharedMem!");
        exit(EXIT_FAILURE);
    }
```



```c
    sharedMem->signalArrived=0;

    sharedMem->ingredients[0]='\0';
    sharedMem->ingredients[1]='\0';

    //initializing unnamed semaphores in the shared memory

    if(sem_init(&sharedMem->wholesalerSem, 1, 1) == -1){
        perror("Error sem_init wholesalerSem!\n");
        exit(EXIT_FAILURE);
    }

    if(sem_init(&sharedMem->ingredientsArrived, 1, 0) == -1){
        perror("Error sem_init ingredientsArrived!\n");
        exit(EXIT_FAILURE);
    }

    if(sem_init(&sharedMem->walnutSugar, 1, 0) == -1){
        perror("Error sem_init walnutSugar!\n");
        exit(EXIT_FAILURE);
    }

    if(sem_init(&sharedMem->flourWalnut, 1, 0) == -1){
        perror("Error sem_init flourWalnut!\n");
        exit(EXIT_FAILURE);
    }
```

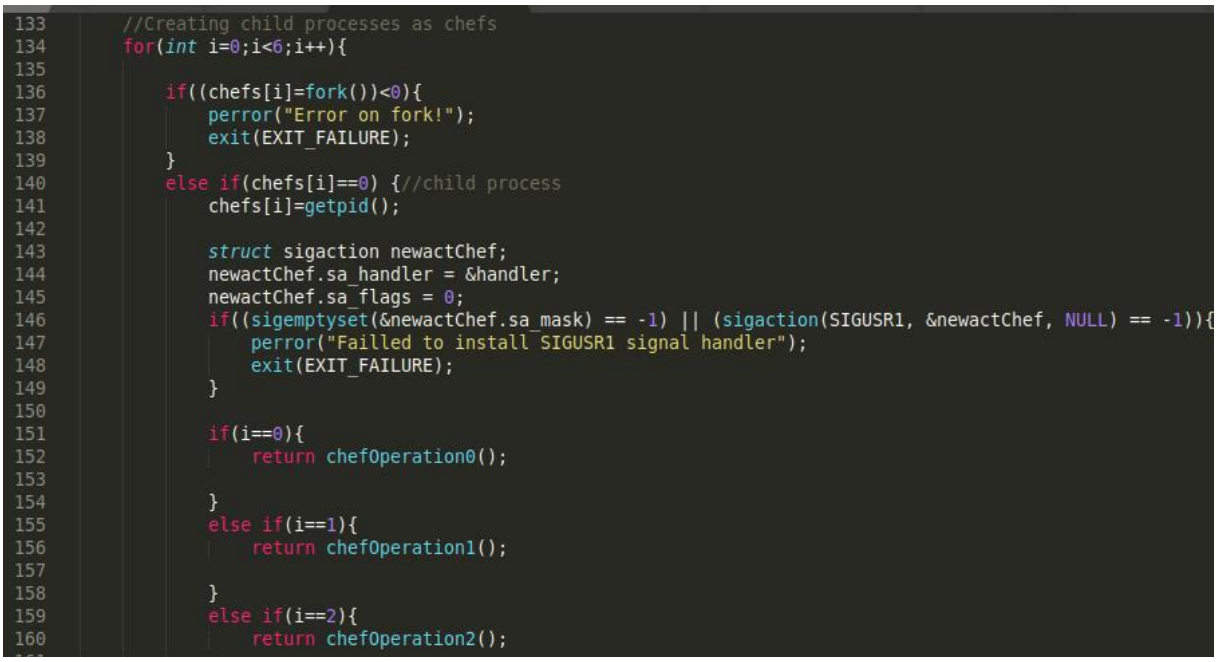
The input file taken from commandline, is opened and file readen. All characters in the file are counted with my **countCharsInFile()** function. This function is necesarry to decide file’s size and create a temporary array named allFile[]. After file operations are finished, all files are closed.



```c
    if ((fdInput = open(inputFilePath, O_RDONLY)) == -1){
        perror("Failed to open input file in main.\n");
        exit(EXIT_FAILURE);
    }

    int charCounts=countCharsInFile(fdInput);   //count the chars from input file to create an array
    close(fdInput);

    char allFile[charCounts];
    char buf[1];
    int i=0,bytesread;

    int fdInput2;
    if ((fdInput2 = open(inputFilePath, O_RDONLY)) == -1){
        perror("Failed to open input file in main.\n");
        exit(EXIT_FAILURE);
    }
    while(1){
        while(((bytesread = read(fdInput2, buf, 1)) == -1) &&(errno == EINTR));
        if(bytesread <= 0)
            break;

        allFile[i]=buf[0];
        i++;
    }
    close(fdInput2);
```

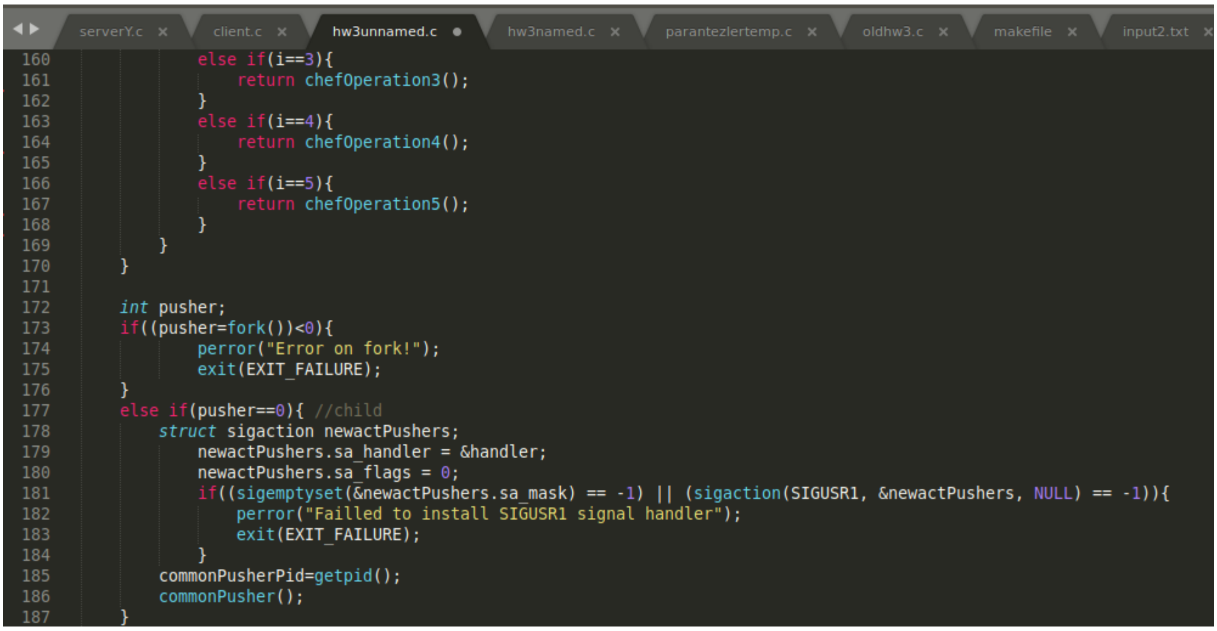
In main function, after file operations, I have created children processes. 6 chefs and 1 pusher will be child processes.

Before explaining the formation of processes here, I would like to explain the purpose of the pusher process in my design. I thought of developing a pusher for each material and a semaphore for each chef, just like the solution developed there for the cigarette smokers problem we normally see in class. However, with the ingredientsArrived semaphore that I specified in my shared memory segment, I was able to solve the problem by using one pusher instead ofmultiple pushers. Here, by looking at the pusher process ingredientsArrived semaphore, it posts the semaphores of the relevant materials when the material arrives.



```
    //Creating child processes as chefs
    for(int i=0;i<6;i++){

        if((chefs[i]=fork())<0){
            perror("Error on fork!");
            exit(EXIT_FAILURE);
        }
        else if(chefs[i]==0) {//child process
            chefs[i]=getpid();

            struct sigaction newactChef;
            newactChef.sa_handler = &handler;
            newactChef.sa_flags = 0;
            if((sigemptyset(&newactChef.sa_mask) == -1) || (sigaction(SIGUSR1, &newactChef, NULL) == -1)){
                perror("Failled to install SIGUSR1 signal handler");
                exit(EXIT_FAILURE);
            }

            if(i==0){
                return chefOperation0();

            }
            else if(i==1){
                return chefOperation1();

            }
            else if(i==2){
                return chefOperation2();
```

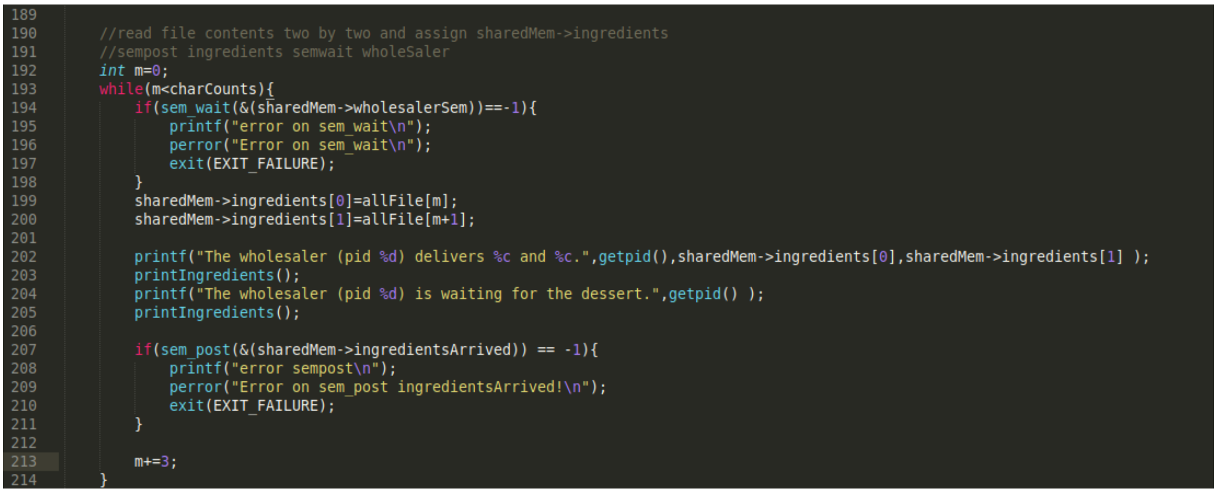
Creating chef processes



```
            else if(i==3){
                return chefOperation3();
            }
            else if(i==4){
                return chefOperation4();
            }
            else if(i==5){
                return chefOperation5();
            }
        }
    }

    int pusher;
    if((pusher=fork())<0){
            perror("Error on fork!");
            exit(EXIT_FAILURE);
    }
    else if(pusher==0){ //child
        struct sigaction newactPushers;
            newactPushers.sa_handler = &handler;
            newactPushers.sa_flags = 0;
            if((sigemptyset(&newactPushers.sa_mask) == -1) || (sigaction(SIGUSR1, &newactPushers, NULL) == -1)){
                perror("Failled to install SIGUSR1 signal handler");
                exit(EXIT_FAILURE);
            }
        commonPusherPid=getpid();
        commonPusher();
    }
```

Creating pusher process

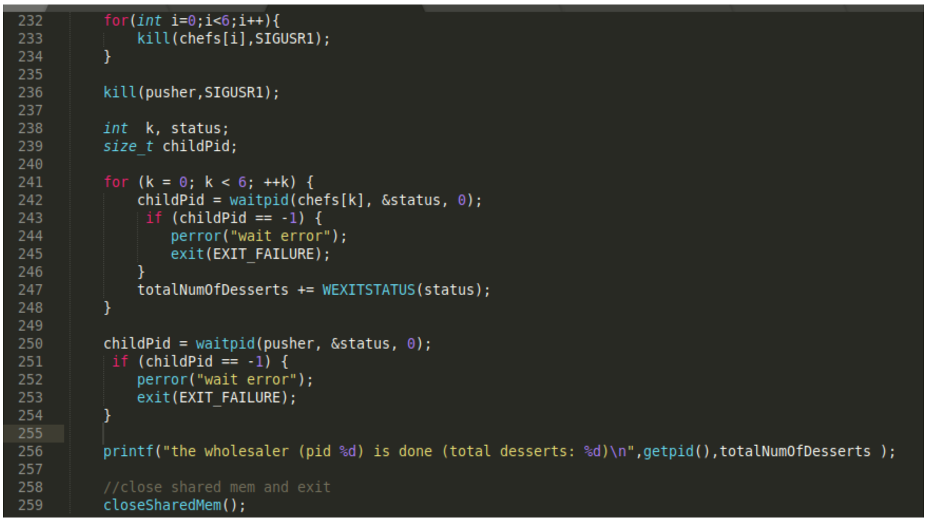
After creating children processes as chefs and pusher, I wrote a loop in the main method. This loop takes two ingredients from the file, since the wholesaler will take the ingredients it operates wait() on wholesaler. Then wholesaler takes ingredients, and finally it posts ingredientArrived semaphore. By operating post() on ingredientArrived ,now we know that there are ingredients and necesarry chef can perform their actions.



```c
    //read file contents two by two and assign sharedMem->ingredients
    //sempost ingredients semwait wholeSaler
    int m=0;
    while(m<charCounts){
        if(sem_wait(&(sharedMem->wholesalerSem))==-1){
            printf("error on sem_wait\n");
            perror("Error on sem_wait\n");
            exit(EXIT_FAILURE);
        }
        sharedMem->ingredients[0]=allFile[m];
        sharedMem->ingredients[1]=allFile[m+1];

        printf("The wholesaler (pid %d) delivers %c and %c.",getpid(),sharedMem->ingredients[0],sharedMem->ingredients[1] );
        printIngredients();
        printf("The wholesaler (pid %d) is waiting for the dessert.",getpid() );
        printIngredients();

        if(sem_post(&(sharedMem->ingredientsArrived)) == -1){
            printf("error sempost\n");
            perror("Error on sem_post ingredientsArrived!\n");
            exit(EXIT_FAILURE);
        }

        m+=3;
    }
```

While loop in the main function

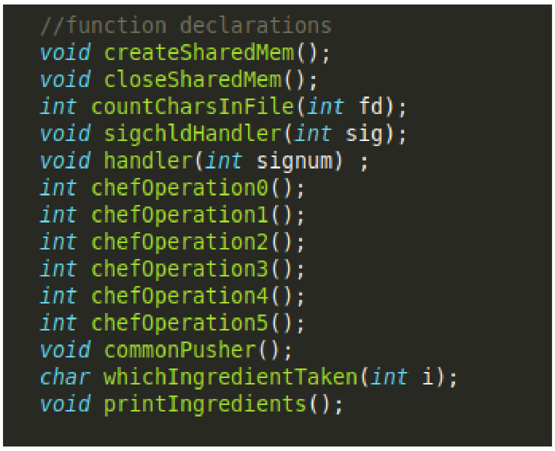
The main function finally kills all children processes after their work is done, and takes chef’s return values. It prints total dessert number. After calling closeSharedMemory() function, main method is done.



```c
    for(int i=0;i<6;i++){
        kill(chefs[i],SIGUSR1);
    }

    kill(pusher,SIGUSR1);

    int  k, status;
    size_t childPid;

    for (k = 0; k < 6; ++k) {
        childPid = waitpid(chefs[k], &status, 0);
         if (childPid == -1) {
            perror("wait error");
            exit(EXIT_FAILURE);
        }
        totalNumOfDesserts += WEXITSTATUS(status);
    }

    childPid = waitpid(pusher, &status, 0);
     if (childPid == -1) {
        perror("wait error");
        exit(EXIT_FAILURE);
    }

    printf("the wholesaler (pid %d) is done (total desserts: %d)\n",getpid(),totalNumOfDesserts );

    //close shared mem and exit
    closeSharedMem();
```

Final operations in main function

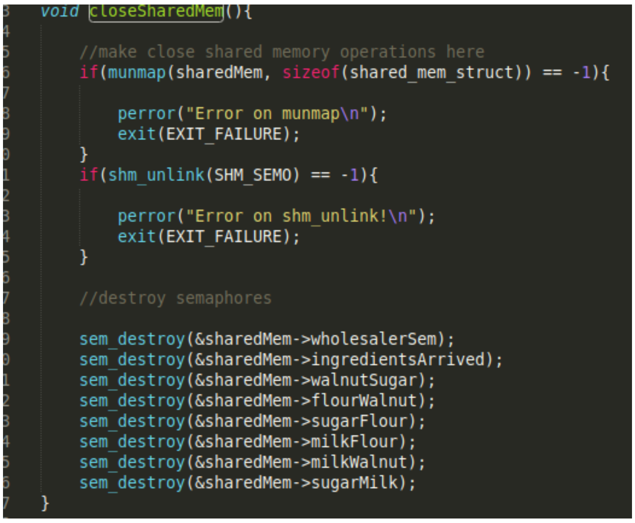
I explained the general flow of the main method. Now I will explain other functions used here.



```c
//function declarations
void createSharedMem();
void closeSharedMem();
int countCharsInFile(int fd);
void sigchldHandler(int sig);
void handler(int signum) ;
int chefOperation0();
int chefOperation1();
int chefOperation2();
int chefOperation3();
int chefOperation4();
int chefOperation5();
void commonPusher();
char whichIngredientTaken(int i);
void printIngredients();
```

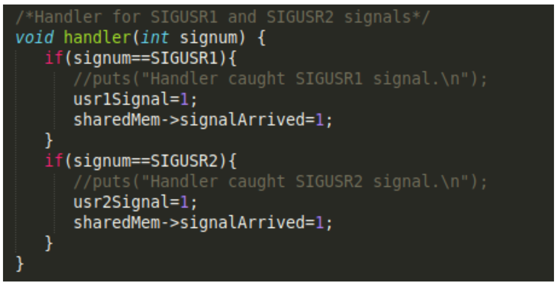
CreateSharedMem and countCharsInFile are explained inside main method flow.

CloseSharedMem() : Closes opened shared memory, and unlinks all semaphores used in the program. Also unlinking a semaphore doesn't means that it has been deleted so this method uses sem_destroy() after unlink semaphores.



```
void closeSharedMem(){

    //make close shared memory operations here
    if(munmap(sharedMem, sizeof(shared_mem_struct)) == -1){

        perror("Error on munmap\n");
        exit(EXIT_FAILURE);
    }
    if(shm_unlink(SHM_SEMO) == -1){

        perror("Error on shm_unlink!\n");
        exit(EXIT_FAILURE);
    }

    //destroy semaphores

    sem_destroy(&sharedMem->wholesalerSem);
    sem_destroy(&sharedMem->ingredientsArrived);
    sem_destroy(&sharedMem->walnutSugar);
    sem_destroy(&sharedMem->flourWalnut);
    sem_destroy(&sharedMem->sugarFlour);
    sem_destroy(&sharedMem->milkFlour);
    sem_destroy(&sharedMem->milkWalnut);
    sem_destroy(&sharedMem->sugarMilk);
}
```

handler(int signum): Signal handler method. It is used to send signals and handle them while making operations with children processes.



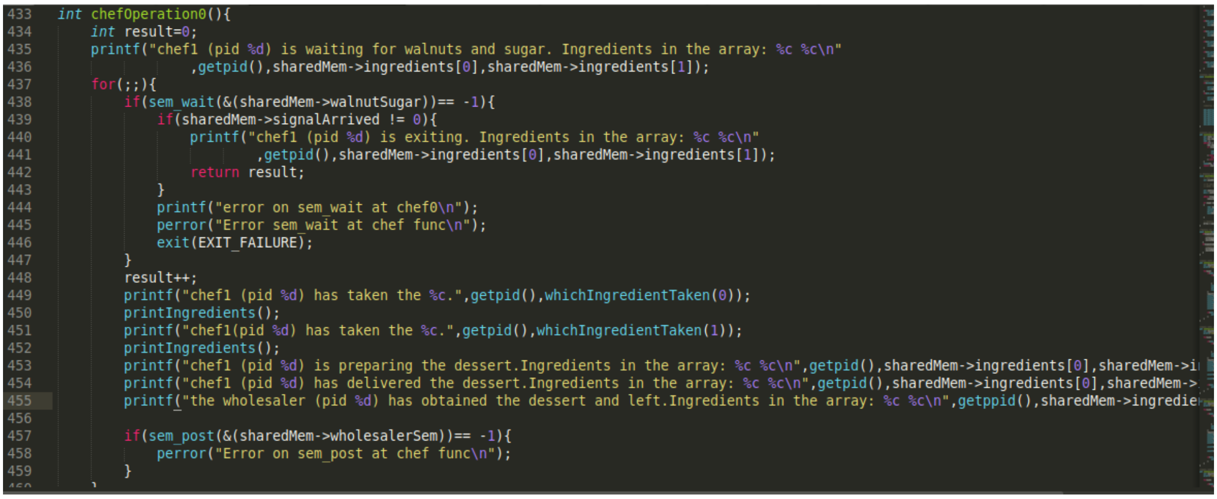
```
/*Handler for SIGUSR1 and SIGUSR2 signals*/
void handler(int signum) {
    if(signum==SIGUSR1){
        //puts("Handler caught SIGUSR1 signal.\n");
        usr1Signal=1;
        sharedMem->signalArrived=1;
    }
    if(signum==SIGUSR2){
        //puts("Handler caught SIGUSR2 signal.\n");
        usr2Signal=1;
        sharedMem->signalArrived=1;
    }
}
```

chefOperation0() | chefOperation1() | chefOperation2() | chefOperation3() | chefOperation4() | chefOperation5(): These functions are serves to same aim, they all performs chefs operations. Only difference between these methods is change of used unnamed semaphore. For example: while chefOperation0 using the walnutSugar semaphore, chefOperation1 uses flourWalnut semaphore. Except this difference, since the operation is the same, I'll just explain the chef0 function:

The result variable in the chef function is to get the number of desserts made by that chef.

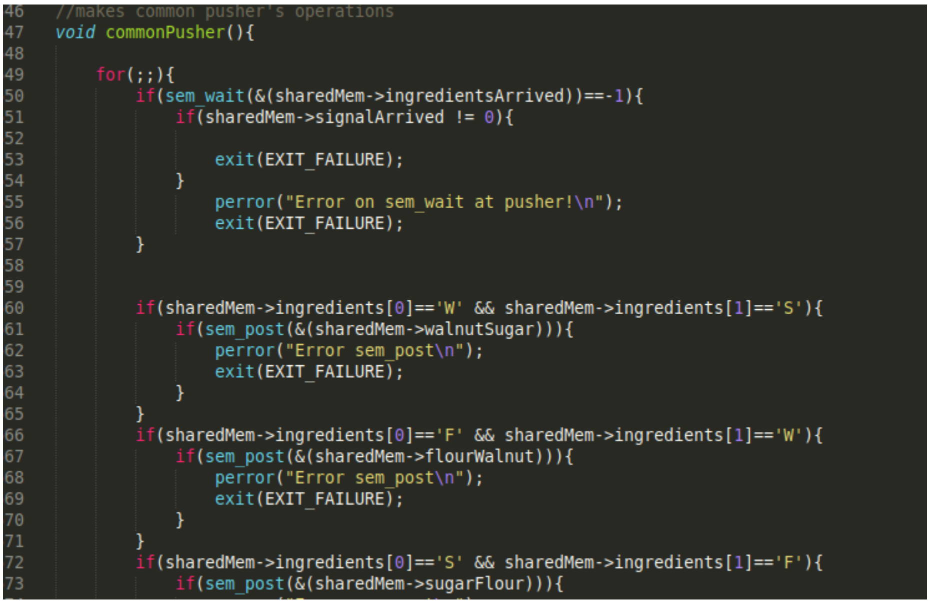
When it comes to the function, the materials that the chef is waiting for are printed on the screen. Then, in an endless for loop, the chef takes the ingredients, prepares the dessert and delivers it to the wholesaler. At the end of all processes, the wholeSaler semaphore is posted so that he can take this dessert and bring other ingredients.

The condition for exiting the endless loop is the SIGUSR1 signal that comes to the process if the material semaphore used by that chief does not give an error to the wait(). As the loop ends, the chef process returns the number of desserts it has made.



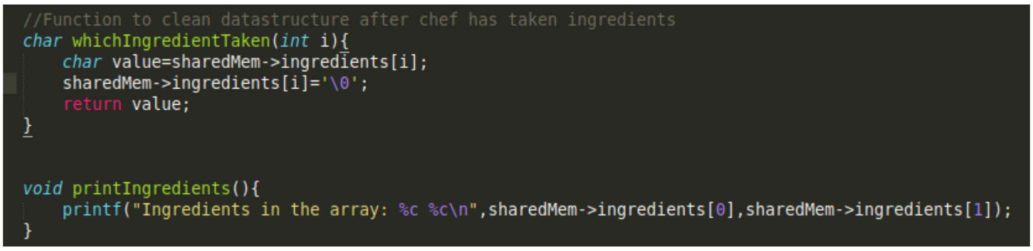
```c
int chefOperation0(){
    int result=0;
    printf("chef1 (pid %d) is waiting for walnuts and sugar. Ingredients in the array: %c %c\n"
                ,getpid(),sharedMem->ingredients[0],sharedMem->ingredients[1]);
    for(;;){
        if(sem_wait(&(sharedMem->walnutSugar))== -1){
            if(sharedMem->signalArrived != 0){
                printf("chef1 (pid %d) is exiting. Ingredients in the array: %c %c\n"
                        ,getpid(),sharedMem->ingredients[0],sharedMem->ingredients[1]);
                return result;
            }
            printf("error on sem_wait at chef0\n");
            perror("Error sem_wait at chef func\n");
            exit(EXIT_FAILURE);
        }
        result++;
        printf("chef1 (pid %d) has taken the %c.",getpid(),whichIngredientTaken(0));
        printIngredients();
        printf("chef1(pid %d) has taken the %c.",getpid(),whichIngredientTaken(1));
        printIngredients();
        printf("chef1 (pid %d) is preparing the dessert.Ingredients in the array: %c %c\n",getpid(),sharedMem->ingredients[0],sharedMem->i
        printf("chef1 (pid %d) has delivered the dessert.Ingredients in the array: %c %c\n",getpid(),sharedMem->ingredients[0],sharedMem->
        printf("the wholesaler (pid %d) has obtained the dessert and left.Ingredients in the array: %c %c\n",getppid(),sharedMem->ingredie

        if(sem_post(&(sharedMem->wholesalerSem))== -1){
            perror("Error on sem_post at chef func\n");
        }
    }
```

commonPusher(): Like the chef functions, the pusher function continues its operations in an endless loop. When the process will terminate, the function is terminated with the signal received. Other than that, instead of writing a pusher for each material, this function looks at the incoming material with if conditions and posts the semaphore of the relevant material if the isIngredient semaphore is post.



```c
//makes common pusher's operations
void commonPusher(){

    for(;;){
        if(sem_wait(&(sharedMem->ingredientsArrived))==-1){
            if(sharedMem->signalArrived != 0){

                exit(EXIT_FAILURE);
            }
                perror("Error on sem_wait at pusher!\n");
                exit(EXIT_FAILURE);
        }


        if(sharedMem->ingredients[0]=='W' && sharedMem->ingredients[1]=='S'){
            if(sem_post(&(sharedMem->walnutSugar))){
                perror("Error sem_post\n");
                exit(EXIT_FAILURE);
            }
        }
        if(sharedMem->ingredients[0]=='F' && sharedMem->ingredients[1]=='W'){
            if(sem_post(&(sharedMem->flourWalnut))){
                perror("Error sem_post\n");
                exit(EXIT_FAILURE);
            }
        }
        if(sharedMem->ingredients[0]=='S' && sharedMem->ingredients[1]=='F'){
            if(sem_post(&(sharedMem->sugarFlour))){
```

whichIngredientTaken(int i): The char array to be used in the homework pdf should to be used efficiently to input and output materials. With this function, if the necessary materials come to the array, I ensure that the chef takes those materials one by one and empty the array.



```
//Function to clean datastructure after chef has taken ingredients
char whichIngredientTaken(int i){
    char value=sharedMem->ingredients[i];
    sharedMem->ingredients[i]='\0';
    return value;
}


void printIngredients(){
    printf("Ingredients in the array: %c %c\n",sharedMem->ingredients[0],sharedMem->ingredients[1]);
}
```

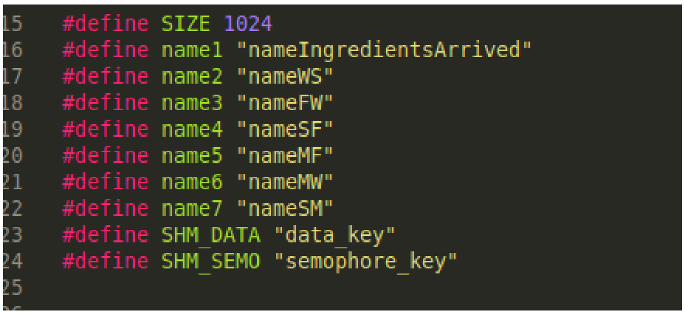
printIngredient(): It is a function that prints the materials in the array.

Named Semaphore Version

In the second version, named semaphores, the main function flow, written chef and pusher functions, and the methods of providing synchronization are exactly the same. Therefore, I will not explain them again in this part of the report.

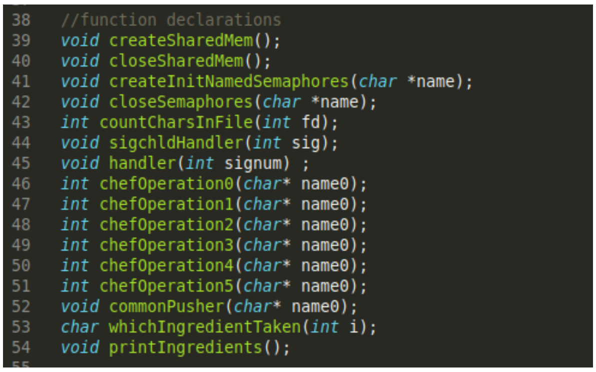
The only difference when using a named semaphore is that in this version it takes a name value from the user and creates the required semaphores with a name when creating them. Since it may take a long time to get names from users as many as the number of semaphores I will use, I only took a name for the semaphor that replaces the wholeSaler and determined the rest myself.

Shared memory was also used in this version, but there was no need to keep semaphores in shared memory. Only the array holding the materials and the flag used for signal control were kept in the shared memory. Apart from that, since we use named semaphores, I added new methods that do these operations because the creation and closing of these semaphores are different from unnamed semaphores. After each process (parent and all childrens) completes its operations, they close the semaphores before exiting.



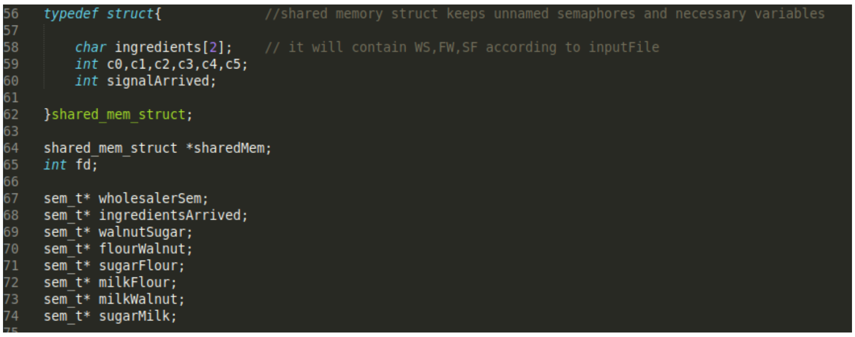
```
#define SIZE 1024
#define name1 "nameIngredientsArrived"
#define name2 "nameWS"
#define name3 "nameFW"
#define name4 "nameSF"
#define name5 "nameMF"
#define name6 "nameMW"
#define name7 "nameSM"
#define SHM_DATA "data_key"
#define SHM_SEMO "semophore_key"

```

Defined semaphore names except the one taken from commandline



```
//function declarations
void createSharedMem();
void closeSharedMem();
void createInitNamedSemaphores(char *name);
void closeSemaphores(char *name);
int countCharsInFile(int fd);
void sigchldHandler(int sig);
void handler(int signum) ;
int chefOperation0(char* name0);
int chefOperation1(char* name0);
int chefOperation2(char* name0);
int chefOperation3(char* name0);
int chefOperation4(char* name0);
int chefOperation5(char* name0);
void commonPusher(char* name0);
char whichIngredientTaken(int i);
void printIngredients();
```

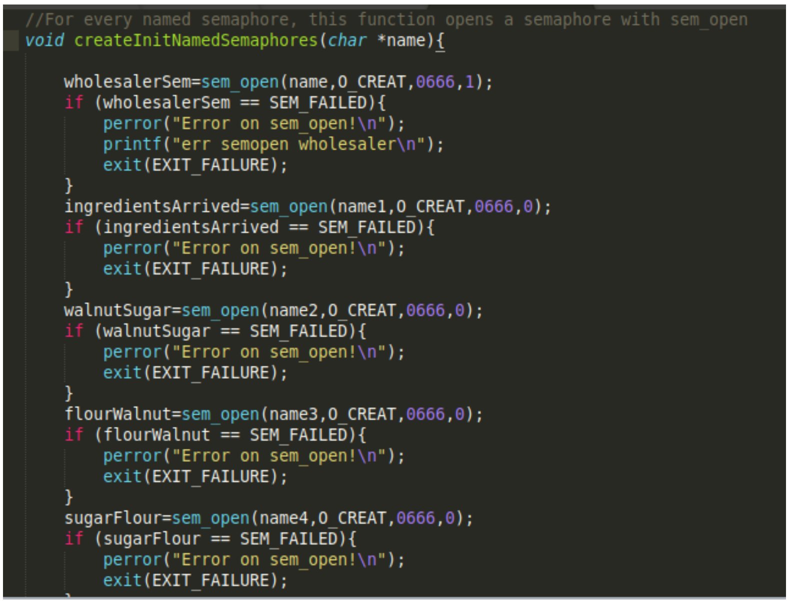
Functions used in named version



```
typedef struct{            //shared memory struct keeps unnamed semaphores and necessary variables

    char ingredients[2];    // it will contain WS,FW,SF according to inputFile
    int c0,c1,c2,c3,c4,c5;
    int signalArrived;

}shared_mem_struct;

shared_mem_struct *sharedMem;
int fd;

sem_t* wholesalerSem;
sem_t* ingredientsArrived;
sem_t* walnutSugar;
sem_t* flourWalnut;
sem_t* sugarFlour;
sem_t* milkFlour;
sem_t* milkWalnut;
sem_t* sugarMilk;
```

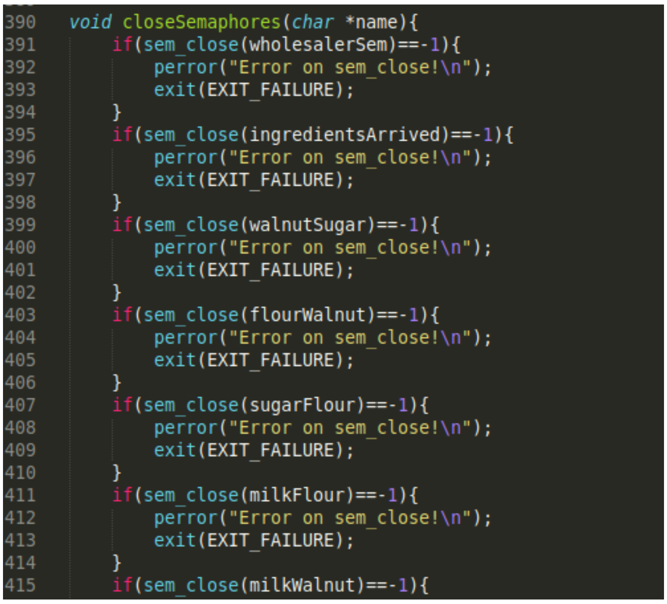
Shared memory and named semaphores

createInitNamedSemaphores(char *name): This function is written for named semaphores version. It opens all semaphores and makes error checks.

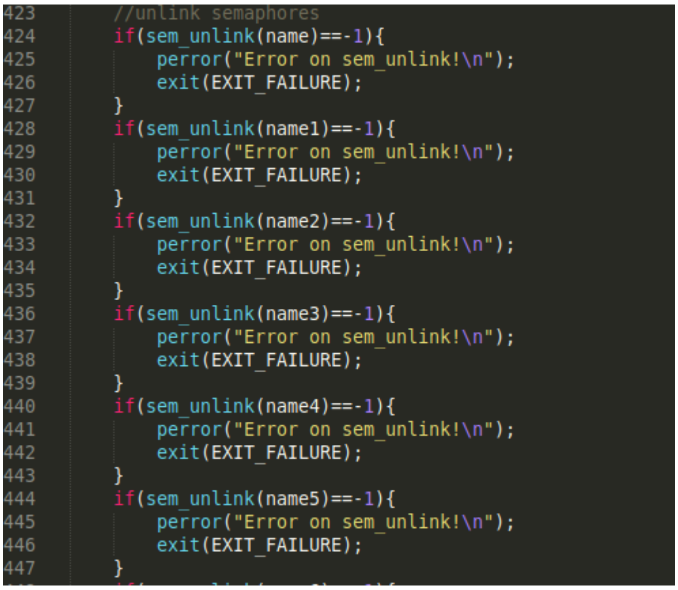


```
//For every named semaphore, this function opens a semaphore with sem_open
void createInitNamedSemaphores(char *name){

    wholesalerSem=sem_open(name,O_CREAT,0666,1);
    if (wholesalerSem == SEM_FAILED){
        perror("Error on sem_open!\n");
        printf("err semopen wholesaler\n");
        exit(EXIT_FAILURE);
    }
    ingredientsArrived=sem_open(name1,O_CREAT,0666,0);
    if (ingredientsArrived == SEM_FAILED){
        perror("Error on sem_open!\n");
        exit(EXIT_FAILURE);
    }
    walnutSugar=sem_open(name2,O_CREAT,0666,0);
    if (walnutSugar == SEM_FAILED){
        perror("Error on sem_open!\n");
        exit(EXIT_FAILURE);
    }
    flourWalnut=sem_open(name3,O_CREAT,0666,0);
    if (flourWalnut == SEM_FAILED){
        perror("Error on sem_open!\n");
        exit(EXIT_FAILURE);
    }
    sugarFlour=sem_open(name4,O_CREAT,0666,0);
    if (sugarFlour == SEM_FAILED){
        perror("Error on sem_open!\n");
        exit(EXIT_FAILURE);
```

closeSemaphores(char *name): This function is makes sem_close on every named semaphore. Since closing a named semaphore doesn't delete it, also after sem_close() , for every semaphore, this function makes sem_unlink() operation to destroy semaphores.

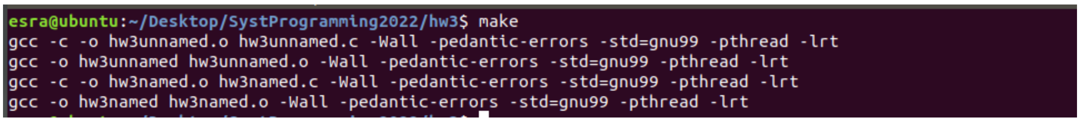


```c
void closeSemaphores(char *name){
    if(sem_close(wholesalerSem)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(ingredientsArrived)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(walnutSugar)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(flourWalnut)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(sugarFlour)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(milkFlour)==-1){
        perror("Error on sem_close!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_close(milkWalnut)==-1){
```



```c
    //unlink semaphores
    if(sem_unlink(name)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_unlink(name1)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_unlink(name2)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_unlink(name3)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_unlink(name4)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
    if(sem_unlink(name5)==-1){
        perror("Error on sem_unlink!\n");
        exit(EXIT_FAILURE);
    }
```

PROGRAM OUTPUTS



```
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$ make
gcc -c -o hw3unnamed.o hw3unnamed.c -Wall -pedantic-errors -std=gnu99 -pthread -lrt
gcc -o hw3unnamed hw3unnamed.o -Wall -pedantic-errors -std=gnu99 -pthread -lrt
gcc -c -o hw3named.o hw3named.c -Wall -pedantic-errors -std=gnu99 -pthread -lrt
gcc -o hw3named hw3named.o -Wall -pedantic-errors -std=gnu99 -pthread -lrt
```

Programs compiled with wall

Unnamed Semaphore

Input file consists: (I've keep it simple for achieve shorter outputs)

WS

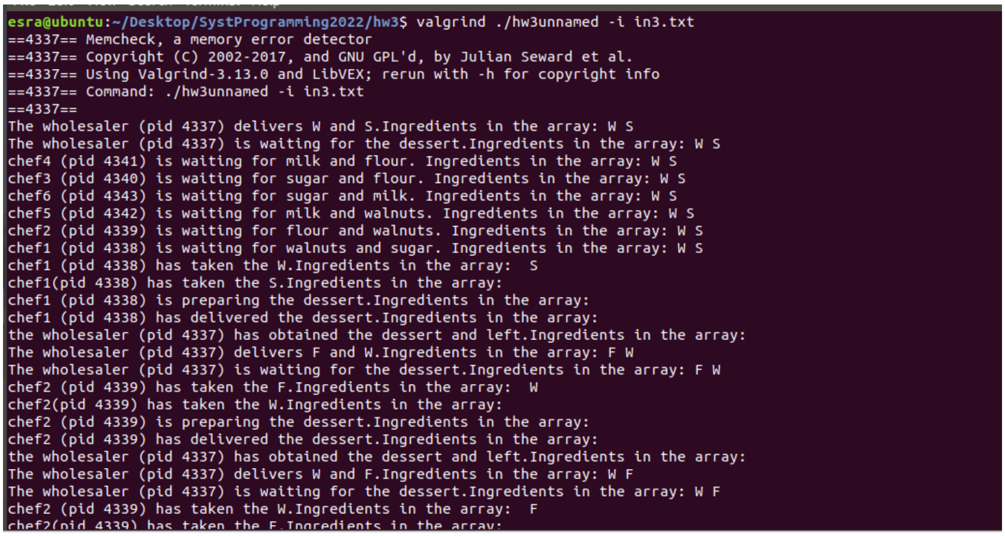
FW

WF

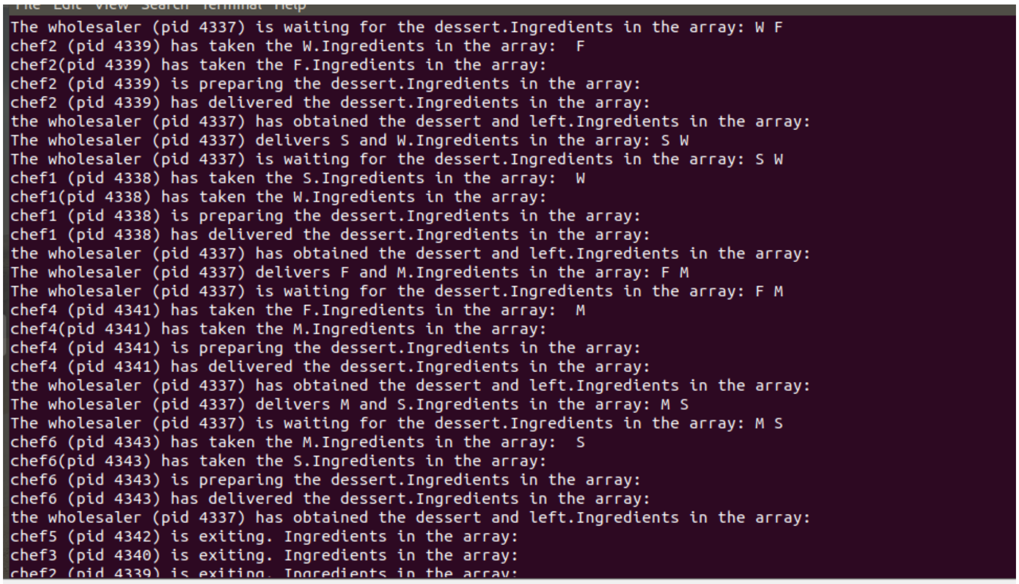
SW

FM

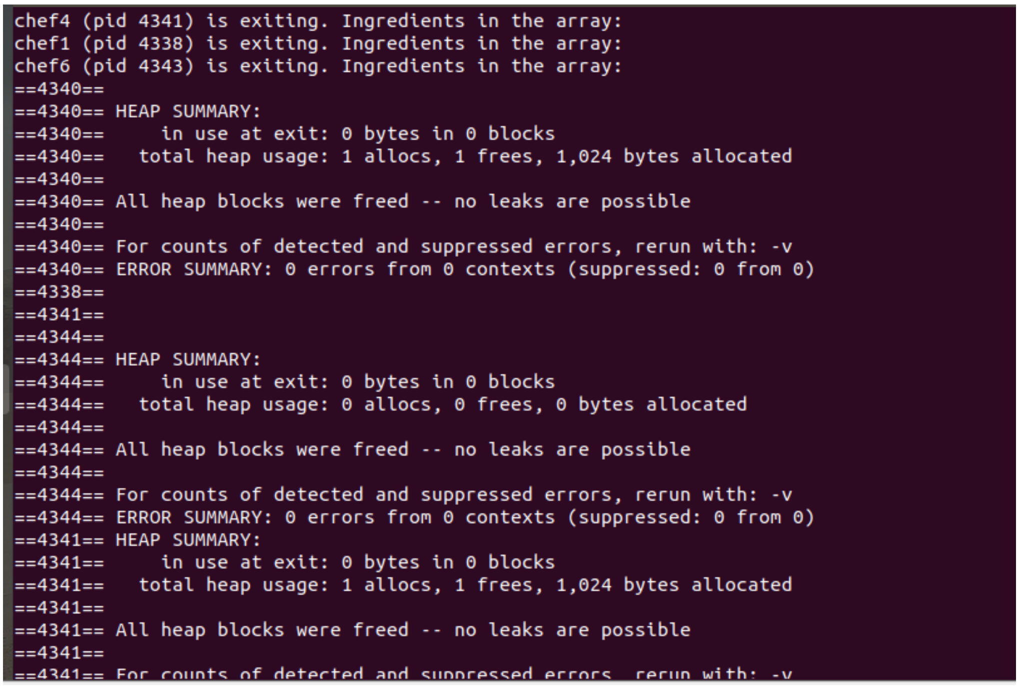
MS



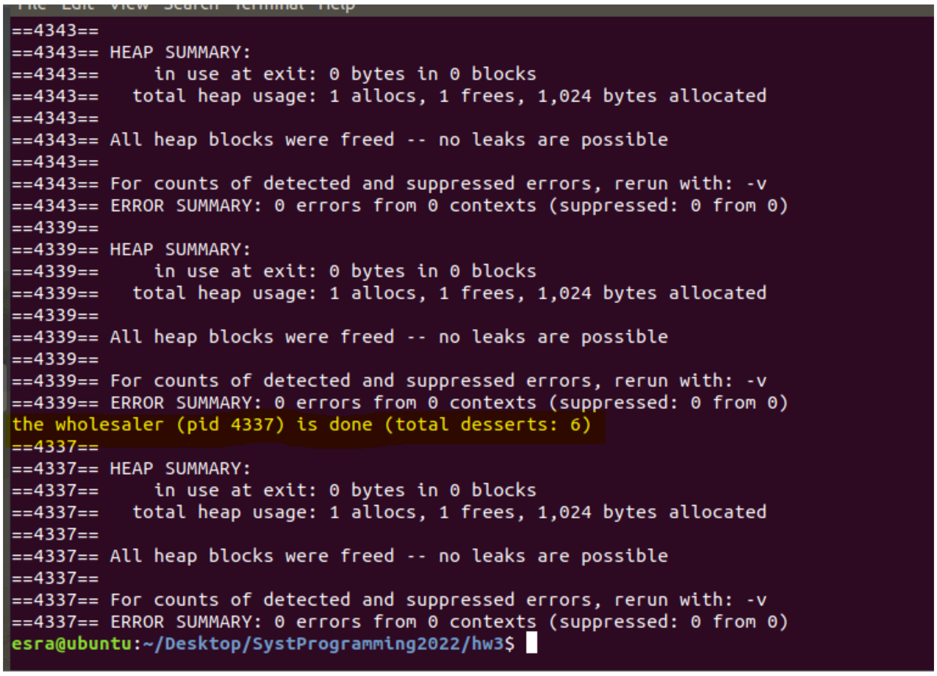
```
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$ valgrind ./hw3unnamed -i in3.txt
==4337== Memcheck, a memory error detector
==4337== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4337== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4337== Command: ./hw3unnamed -i in3.txt
==4337==
The wholesaler (pid 4337) delivers W and S.Ingredients in the array: W S
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: W S
chef4 (pid 4341) is waiting for milk and flour. Ingredients in the array: W S
chef3 (pid 4340) is waiting for sugar and flour. Ingredients in the array: W S
chef6 (pid 4343) is waiting for sugar and milk. Ingredients in the array: W S
chef5 (pid 4342) is waiting for milk and walnuts. Ingredients in the array: W S
chef2 (pid 4339) is waiting for flour and walnuts. Ingredients in the array: W S
chef1 (pid 4338) is waiting for walnuts and sugar. Ingredients in the array: W S
chef1 (pid 4338) has taken the W.Ingredients in the array:  S
chef1(pid 4338) has taken the S.Ingredients in the array:
chef1 (pid 4338) is preparing the dessert.Ingredients in the array:
chef1 (pid 4338) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4337) delivers F and W.Ingredients in the array: F W
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: F W
chef2 (pid 4339) has taken the F.Ingredients in the array:  W
chef2(pid 4339) has taken the W.Ingredients in the array:
chef2 (pid 4339) is preparing the dessert.Ingredients in the array:
chef2 (pid 4339) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4337) delivers W and F.Ingredients in the array: W F
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: W F
chef2 (pid 4339) has taken the W.Ingredients in the array:  F
chef2(pid 4339) has taken the F.Ingredients in the array:
```



```
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: W F
chef2 (pid 4339) has taken the W.Ingredients in the array:  F
chef2(pid 4339) has taken the F.Ingredients in the array:
chef2 (pid 4339) is preparing the dessert.Ingredients in the array:
chef2 (pid 4339) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4337) delivers S and W.Ingredients in the array: S W
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: S W
chef1 (pid 4338) has taken the S.Ingredients in the array:  W
chef1(pid 4338) has taken the W.Ingredients in the array:
chef1 (pid 4338) is preparing the dessert.Ingredients in the array:
chef1 (pid 4338) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4337) delivers F and M.Ingredients in the array: F M
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: F M
chef4 (pid 4341) has taken the F.Ingredients in the array:  M
chef4(pid 4341) has taken the M.Ingredients in the array:
chef4 (pid 4341) is preparing the dessert.Ingredients in the array:
chef4 (pid 4341) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4337) delivers M and S.Ingredients in the array: M S
The wholesaler (pid 4337) is waiting for the dessert.Ingredients in the array: M S
chef6 (pid 4343) has taken the M.Ingredients in the array:  S
chef6(pid 4343) has taken the S.Ingredients in the array:
chef6 (pid 4343) is preparing the dessert.Ingredients in the array:
chef6 (pid 4343) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4337) has obtained the dessert and left.Ingredients in the array:
chef5 (pid 4342) is exiting. Ingredients in the array:
chef3 (pid 4340) is exiting. Ingredients in the array:
chef2 (pid 4339) is exiting. Ingredients in the array:
```



```
chef4 (pid 4341) is exiting. Ingredients in the array:
chef1 (pid 4338) is exiting. Ingredients in the array:
chef6 (pid 4343) is exiting. Ingredients in the array:
==4340==
==4340== HEAP SUMMARY:
==4340==     in use at exit: 0 bytes in 0 blocks
==4340==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4340==
==4340== All heap blocks were freed -- no leaks are possible
==4340==
==4340== For counts of detected and suppressed errors, rerun with: -v
==4340== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==4338==
==4341==
==4344==
==4344== HEAP SUMMARY:
==4344==     in use at exit: 0 bytes in 0 blocks
==4344==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==4344==
==4344== All heap blocks were freed -- no leaks are possible
==4344==
==4344== For counts of detected and suppressed errors, rerun with: -v
==4344== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==4341== HEAP SUMMARY:
==4341==     in use at exit: 0 bytes in 0 blocks
==4341==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4341==
==4341== All heap blocks were freed -- no leaks are possible
==4341==
==4341== For counts of detected and suppressed errors, rerun with: -v
```

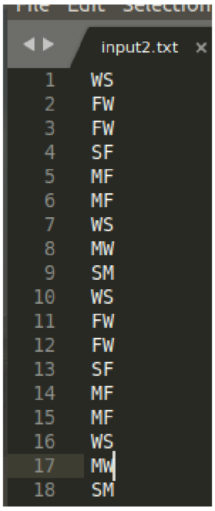


```
==4343==
==4343== HEAP SUMMARY:
==4343==     in use at exit: 0 bytes in 0 blocks
==4343==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4343==
==4343== All heap blocks were freed -- no leaks are possible
==4343==
==4343== For counts of detected and suppressed errors, rerun with: -v
==4343== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==4339==
==4339== HEAP SUMMARY:
==4339==     in use at exit: 0 bytes in 0 blocks
==4339==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4339==
==4339== All heap blocks were freed -- no leaks are possible
==4339==
==4339== For counts of detected and suppressed errors, rerun with: -v
==4339== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
the wholesaler (pid 4337) is done (total desserts: 6)
==4337==
==4337== HEAP SUMMARY:
==4337==     in use at exit: 0 bytes in 0 blocks
==4337==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==4337==
==4337== All heap blocks were freed -- no leaks are possible
==4337==
==4337== For counts of detected and suppressed errors, rerun with: -v
==4337== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$
```

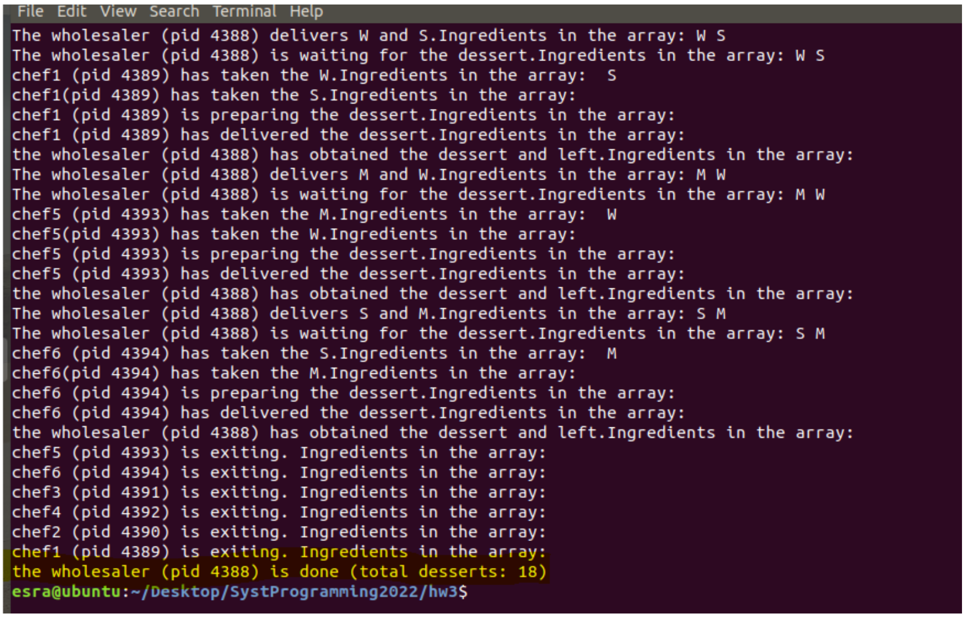
Note: After all print lines, the character array is also printed. We can see that ex: wholesaler brings F and W and array consist F and W. Then chef takes F array contains only W. Same chef takes W and array contains zero ingredients. After wholesaler brings new ingredients , printing array shows new ingredients.

Another output with a longer input file, I will only show total number of desserts to prove output result is correct:

Test file contains: 18 lines, we expect 18 desserts.



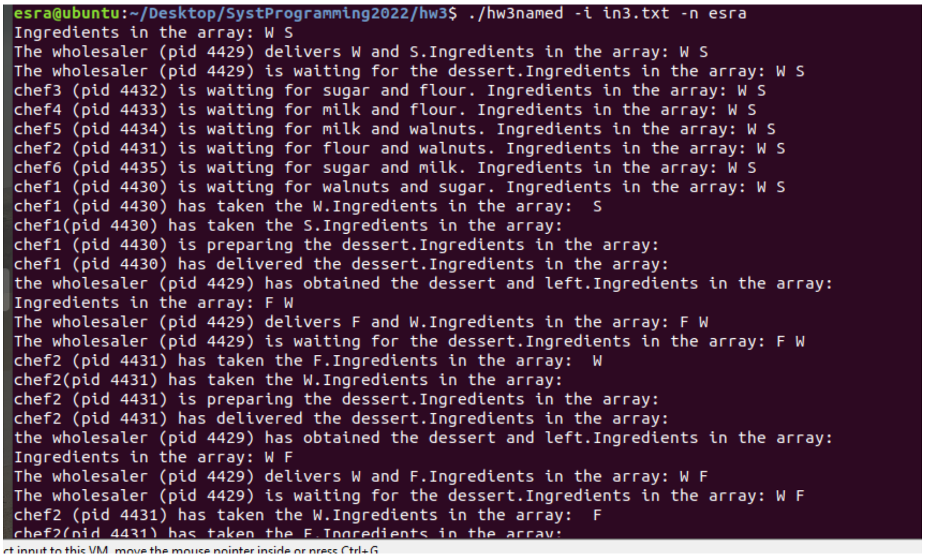
```
WS
FW
FW
SF
MF
MF
WS
MW
SM
WS
FW
FW
SF
MF
MF
WS
MW
SM
```



```
The wholesaler (pid 4388) delivers W and S.Ingredients in the array: W S
The wholesaler (pid 4388) is waiting for the dessert.Ingredients in the array: W S
chef1 (pid 4389) has taken the W.Ingredients in the array:  S
chef1(pid 4389) has taken the S.Ingredients in the array:
chef1 (pid 4389) is preparing the dessert.Ingredients in the array:
chef1 (pid 4389) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4388) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4388) delivers M and W.Ingredients in the array: M W
The wholesaler (pid 4388) is waiting for the dessert.Ingredients in the array: M W
chef5 (pid 4393) has taken the M.Ingredients in the array:  W
chef5(pid 4393) has taken the W.Ingredients in the array:
chef5 (pid 4393) is preparing the dessert.Ingredients in the array:
chef5 (pid 4393) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4388) has obtained the dessert and left.Ingredients in the array:
The wholesaler (pid 4388) delivers S and M.Ingredients in the array: S M
The wholesaler (pid 4388) is waiting for the dessert.Ingredients in the array: S M
chef6 (pid 4394) has taken the S.Ingredients in the array:  M
chef6(pid 4394) has taken the M.Ingredients in the array:
chef6 (pid 4394) is preparing the dessert.Ingredients in the array:
chef6 (pid 4394) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4388) has obtained the dessert and left.Ingredients in the array:
chef5 (pid 4393) is exiting. Ingredients in the array:
chef6 (pid 4394) is exiting. Ingredients in the array:
chef3 (pid 4391) is exiting. Ingredients in the array:
chef4 (pid 4392) is exiting. Ingredients in the array:
chef2 (pid 4390) is exiting. Ingredients in the array:
chef1 (pid 4389) is exiting. Ingredients in the array:
the wholesaler (pid 4388) is done (total desserts: 18)
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$
```

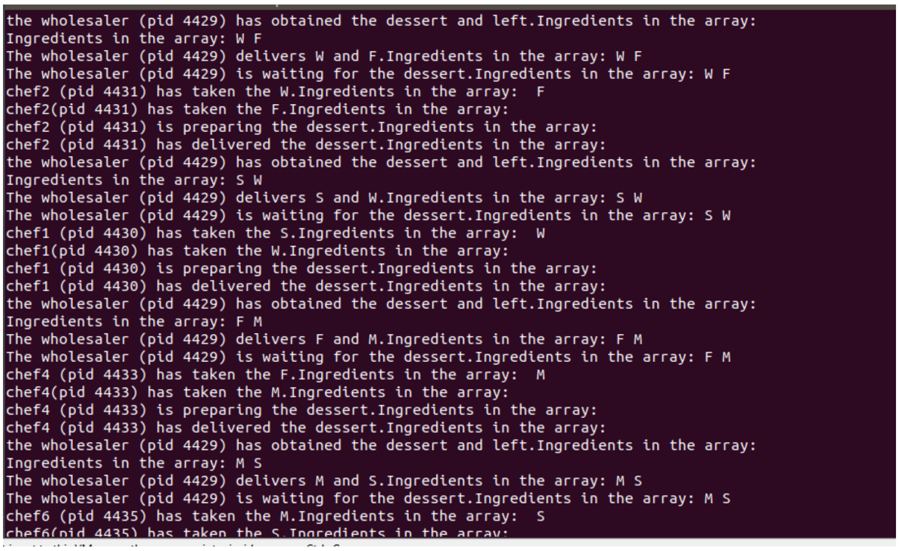
Named Semaphore

Named semaphore input files are same with unnamed semaphore version.

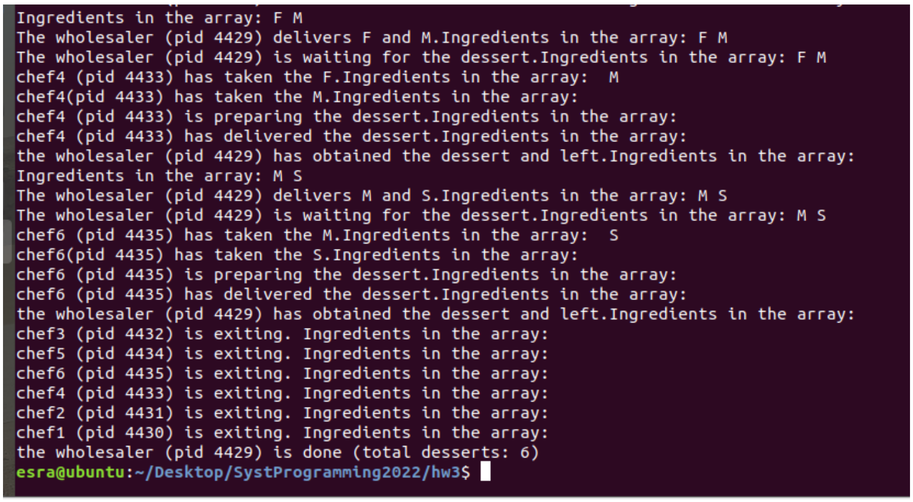


```
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$ ./hw3named -i in3.txt -n esra
Ingredients in the array: W S
The wholesaler (pid 4429) delivers W and S.Ingredients in the array: W S
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: W S
chef3 (pid 4432) is waiting for sugar and flour. Ingredients in the array: W S
chef4 (pid 4433) is waiting for milk and flour. Ingredients in the array: W S
chef5 (pid 4434) is waiting for milk and walnuts. Ingredients in the array: W S
chef2 (pid 4431) is waiting for flour and walnuts. Ingredients in the array: W S
chef6 (pid 4435) is waiting for sugar and milk. Ingredients in the array: W S
chef1 (pid 4430) is waiting for walnuts and sugar. Ingredients in the array: W S
chef1 (pid 4430) has taken the W.Ingredients in the array:  S
chef1(pid 4430) has taken the S.Ingredients in the array:
chef1 (pid 4430) is preparing the dessert.Ingredients in the array:
chef1 (pid 4430) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: F W
The wholesaler (pid 4429) delivers F and W.Ingredients in the array: F W
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: F W
chef2 (pid 4431) has taken the F.Ingredients in the array:  W
chef2(pid 4431) has taken the W.Ingredients in the array:
chef2 (pid 4431) is preparing the dessert.Ingredients in the array:
chef2 (pid 4431) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: W F
The wholesaler (pid 4429) delivers W and F.Ingredients in the array: W F
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: W F
chef2 (pid 4431) has taken the W.Ingredients in the array:  F
chef2(pid 4431) has taken the F.Ingredients in the array:
```

ct input to this VM, move the mouse pointer inside or press Ctrl+G

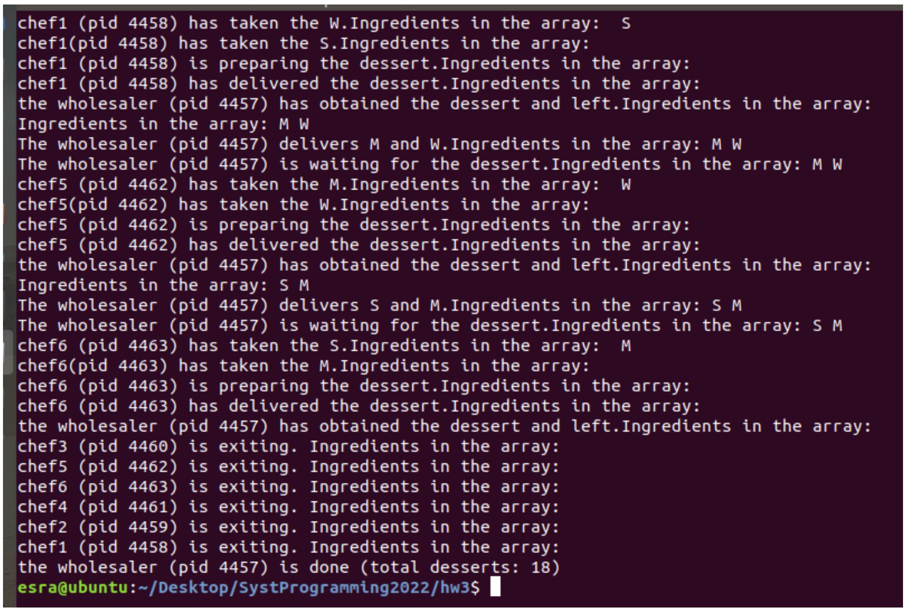


```
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: W F
The wholesaler (pid 4429) delivers W and F.Ingredients in the array: W F
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: W F
chef2 (pid 4431) has taken the W.Ingredients in the array:  F
chef2(pid 4431) has taken the F.Ingredients in the array:
chef2 (pid 4431) is preparing the dessert.Ingredients in the array:
chef2 (pid 4431) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: S W
The wholesaler (pid 4429) delivers S and W.Ingredients in the array: S W
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: S W
chef1 (pid 4430) has taken the S.Ingredients in the array:  W
chef1(pid 4430) has taken the W.Ingredients in the array:
chef1 (pid 4430) is preparing the dessert.Ingredients in the array:
chef1 (pid 4430) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: F M
The wholesaler (pid 4429) delivers F and M.Ingredients in the array: F M
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: F M
chef4 (pid 4433) has taken the F.Ingredients in the array:  M
chef4(pid 4433) has taken the M.Ingredients in the array:
chef4 (pid 4433) is preparing the dessert.Ingredients in the array:
chef4 (pid 4433) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: M S
The wholesaler (pid 4429) delivers M and S.Ingredients in the array: M S
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: M S
chef6 (pid 4435) has taken the M.Ingredients in the array:  S
chef6(pid 4435) has taken the S.Ingredients in the array:
```



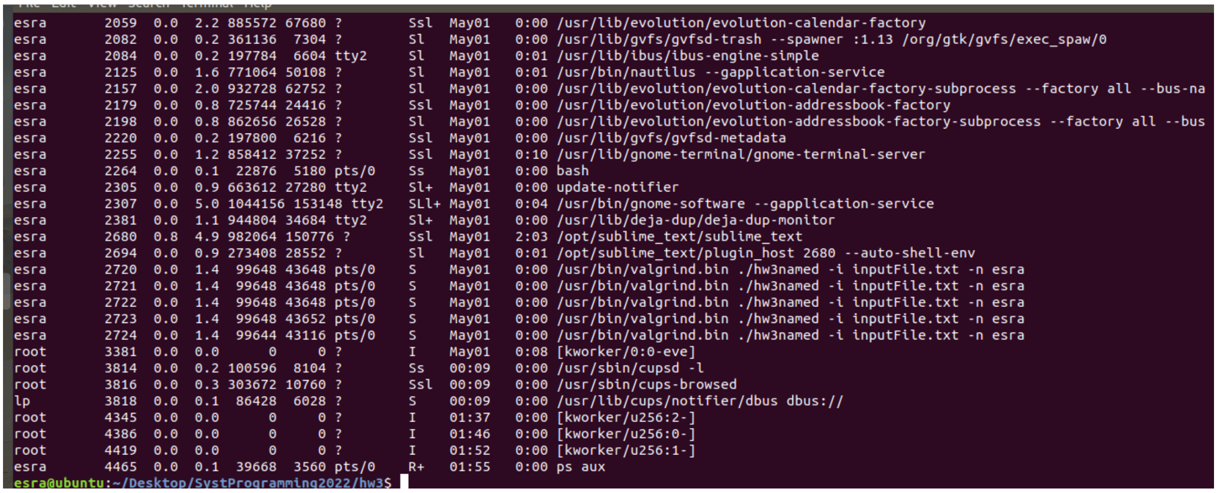
```
Ingredients in the array: F M
The wholesaler (pid 4429) delivers F and M.Ingredients in the array: F M
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: F M
chef4 (pid 4433) has taken the F.Ingredients in the array:  M
chef4(pid 4433) has taken the M.Ingredients in the array:
chef4 (pid 4433) is preparing the dessert.Ingredients in the array:
chef4 (pid 4433) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: M S
The wholesaler (pid 4429) delivers M and S.Ingredients in the array: M S
The wholesaler (pid 4429) is waiting for the dessert.Ingredients in the array: M S
chef6 (pid 4435) has taken the M.Ingredients in the array:  S
chef6(pid 4435) has taken the S.Ingredients in the array:
chef6 (pid 4435) is preparing the dessert.Ingredients in the array:
chef6 (pid 4435) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4429) has obtained the dessert and left.Ingredients in the array:
chef3 (pid 4432) is exiting. Ingredients in the array:
chef5 (pid 4434) is exiting. Ingredients in the array:
chef6 (pid 4435) is exiting. Ingredients in the array:
chef4 (pid 4433) is exiting. Ingredients in the array:
chef2 (pid 4431) is exiting. Ingredients in the array:
chef1 (pid 4430) is exiting. Ingredients in the array:
the wholesaler (pid 4429) is done (total desserts: 6)
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$
```

For longer test file outputs:



```
chef1 (pid 4458) has taken the W.Ingredients in the array:  S
chef1(pid 4458) has taken the S.Ingredients in the array:
chef1 (pid 4458) is preparing the dessert.Ingredients in the array:
chef1 (pid 4458) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4457) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: M W
The wholesaler (pid 4457) delivers M and W.Ingredients in the array: M W
The wholesaler (pid 4457) is waiting for the dessert.Ingredients in the array: M W
chef5 (pid 4462) has taken the M.Ingredients in the array:  W
chef5(pid 4462) has taken the W.Ingredients in the array:
chef5 (pid 4462) is preparing the dessert.Ingredients in the array:
chef5 (pid 4462) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4457) has obtained the dessert and left.Ingredients in the array:
Ingredients in the array: S M
The wholesaler (pid 4457) delivers S and M.Ingredients in the array: S M
The wholesaler (pid 4457) is waiting for the dessert.Ingredients in the array: S M
chef6 (pid 4463) has taken the S.Ingredients in the array:  M
chef6(pid 4463) has taken the M.Ingredients in the array:
chef6 (pid 4463) is preparing the dessert.Ingredients in the array:
chef6 (pid 4463) has delivered the dessert.Ingredients in the array:
the wholesaler (pid 4457) has obtained the dessert and left.Ingredients in the array:
chef3 (pid 4460) is exiting. Ingredients in the array:
chef5 (pid 4462) is exiting. Ingredients in the array:
chef6 (pid 4463) is exiting. Ingredients in the array:
chef4 (pid 4461) is exiting. Ingredients in the array:
chef2 (pid 4459) is exiting. Ingredients in the array:
chef1 (pid 4458) is exiting. Ingredients in the array:
the wholesaler (pid 4457) is done (total desserts: 18)
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$
```

No zombie processes:



```
esra       2059  0.0  2.2 885572 67680 ?        Ssl  May01   0:00 /usr/lib/evolution/evolution-calendar-factory
esra       2082  0.0  0.2 361136  7304 ?        Sl   May01   0:00 /usr/lib/gvfs/gvfsd-trash --spawner :1.13 /org/gtk/gvfs/exec_spaw/0
esra       2084  0.0  0.2 197784  6604 tty2     Sl   May01   0:01 /usr/lib/ibus/ibus-engine-simple
esra       2125  0.0  1.6 771064 50108 ?        Sl   May01   0:01 /usr/bin/nautilus --gapplication-service
esra       2157  0.0  2.0 932728 62752 ?        Sl   May01   0:00 /usr/lib/evolution/evolution-calendar-factory-subprocess --factory all --bus-na
esra       2179  0.0  0.8 725744 24416 ?        Ssl  May01   0:00 /usr/lib/evolution/evolution-addressbook-factory
esra       2198  0.0  0.8 862656 26528 ?        Sl   May01   0:00 /usr/lib/evolution/evolution-addressbook-factory-subprocess --factory all --bus
esra       2220  0.0  0.2 197800  6216 ?        Ssl  May01   0:00 /usr/lib/gvfs/gvfsd-metadata
esra       2255  0.0  1.2 858412 37252 ?        Ssl  May01   0:10 /usr/lib/gnome-terminal/gnome-terminal-server
esra       2264  0.0  0.1  22876  5180 pts/0    Ss   May01   0:00 bash
esra       2305  0.0  0.9 663612 27280 tty2     Sl+  May01   0:00 update-notifier
esra       2307  0.0  5.0 1044156 153148 tty2   SLl+ May01   0:04 /usr/bin/gnome-software --gapplication-service
esra       2381  0.0  1.1 944804 34684 tty2     Sl+  May01   0:00 /usr/lib/deja-dup/deja-dup-monitor
esra       2680  0.8  4.9 982064 150776 ?       Ssl  May01   2:03 /opt/sublime_text/sublime_text
esra       2694  0.0  0.9 273408 28552 ?        Sl   May01   0:01 /opt/sublime_text/plugin_host 2680 --auto-shell-env
esra       2720  0.0  1.4  99648 43648 pts/0    S    May01   0:00 /usr/bin/valgrind.bin ./hw3named -i inputFile.txt -n esra
esra       2721  0.0  1.4  99648 43648 pts/0    S    May01   0:00 /usr/bin/valgrind.bin ./hw3named -i inputFile.txt -n esra
esra       2722  0.0  1.4  99648 43648 pts/0    S    May01   0:00 /usr/bin/valgrind.bin ./hw3named -i inputFile.txt -n esra
esra       2723  0.0  1.4  99648 43652 pts/0    S    May01   0:00 /usr/bin/valgrind.bin ./hw3named -i inputFile.txt -n esra
esra       2724  0.0  1.4  99644 43116 pts/0    S    May01   0:00 /usr/bin/valgrind.bin ./hw3named -i inputFile.txt -n esra
root       3381  0.0  0.0      0     0 ?        I    May01   0:08 [kworker/0:0-eve]
root       3814  0.0  0.2 100596  8104 ?        Ss   00:09   0:00 /usr/sbin/cupsd -l
root       3816  0.0  0.3 303672 10760 ?        Ssl  00:09   0:00 /usr/sbin/cups-browsed
lp         3818  0.0  0.1  86428  6028 ?        S    00:09   0:00 /usr/lib/cups/notifier/dbus dbus://
root       4345  0.0  0.0      0     0 ?        I    01:37   0:00 [kworker/u256:2-]
root       4386  0.0  0.0      0     0 ?        I    01:46   0:00 [kworker/u256:0-]
root       4419  0.0  0.0      0     0 ?        I    01:52   0:00 [kworker/u256:1-]
esra       4465  0.0  0.1  39668  3560 pts/0    R+   01:55   0:00 ps aux
esra@ubuntu:~/Desktop/SystProgramming2022/hw3$
```