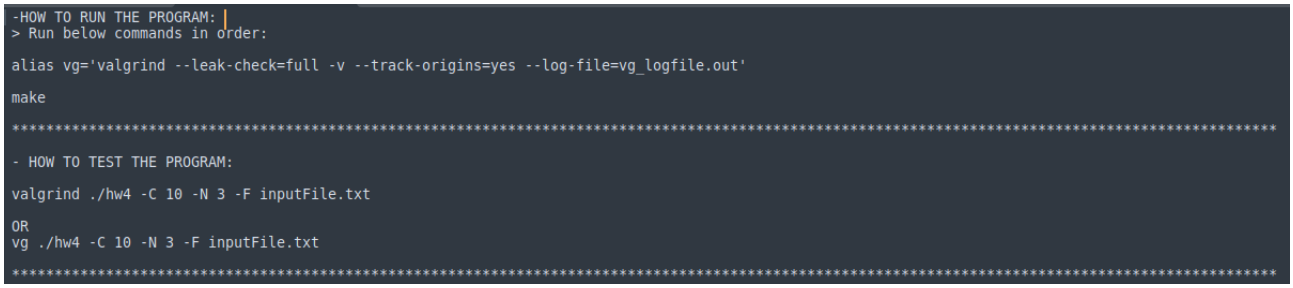
# Systems Programming HW4 Report
# Mohammad Ashraf Yawar 161044123

**- HOW TO RUN AND TEST THE PROGRAM ?**

- You can find the instructions in README.txt in order to run and test the program.



```
-HOW TO RUN THE PROGRAM:
> Run below commands in order:

alias vg='valgrind --leak-check=full -v --track-origins=yes --log-file=vg_logfile.out'

make

*****************************************************************************************************************************************************

- HOW TO TEST THE PROGRAM:

valgrind ./hw4 -C 10 -N 3 -F inputFile.txt

OR
vg ./hw4 -C 10 -N 3 -F inputFile.txt

*****************************************************************************************************************************************************
```

## Implemented Concepts:

- File read, write,System-Calls.
- Signal handling, threads, detached threads, join-able threads.
- Make files.
- Waiting for the threads to finish.
- System V semaphores.
- Producer and consumer paradigm.

## Working Cases:

- This program works for cases all the cases.

## Note Working Cases:

- NONE

## Design Explanation:

- All the System-Calls and their possible return error values are checked with detailed errno checks.

- I have my global and constant variable as:

```c
#define MESSAGE_LEN 1000
#define BUFFER_LIMIT 1// buffer size used to read from file.
#define ACCESS_PERMISSION_FLAG_INPUT  O_RDONLY
#define MODE S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IWGRP | S_IXGRP | S_IROTH | S_IWOTH | S_IXOTH


sig_atomic_t sigintcaught = 0;
union semun{
    int val;
    struct semid_ds * buf;
    unsigned short* array;
    struct seminfo* __buf;
}sem_attr;
int C = 0,N = 0,semid;
pthread_t *consumerThreadsArray=NULL;
```

- In my main program I start the program by controlling and getting some inputs from the argv pointers as:

```c
int main(int argc, char **argv){
    setbuf(stdout, NULL);
    char *input_path = NULL,*message;
    int inpfd = 0,s;
    mode_t mode = MODE;
    pthread_t supplierThreadPointer;
    pthread_attr_t attr;
    struct sigaction newact;
    time_t t;time(&t);

    newact.sa_handler = &siginthandler; /* set the new handler */
    newact.sa_flags = 0;
    if ((sigemptyset(&newact.sa_mask) == -1) || (sigaction(SIGINT,&newact, NULL) == -1)){
        perror("Failed to install SIGINT signal handler");
        exit(EXIT_FAILURE);
    }

    if (sigintcaught == 1){// if sigint has recieved.
        exit(EXIT_FAILURE);
    }

    // check if the user has entered sufficient arguments.
    if (argc < 7){
        perror("No Sufficient Parameters !!!\n");
        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Usage: vg ./hw4 -C 10 -N 5 -F inputfilePath\n",ctime(&t));
        printMessage(message);
        exit(EXIT_FAILURE);
    }else if (argc > 7){
        perror("Too Much Parameters !!!\n");
        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Usage: vg ./hw4 -C 10 -N 5 -F inputfilePath\n",ctime(&t));
        printMessage(message);
        exit(EXIT_FAILURE);
    }

    //file paths read from terminal
    C = atoi(argv[2]);
    N = atoi(argv[4]);
    input_path = argv[6];// input file path
    if (C < 4 && N < 1){
        perror("error:");
        exit(EXIT_FAILURE);
    }
    //input file settings:
    inpfd = open(input_path,ACCESS_PERMISSION_FLAG_INPUT,mode);
    if (inpfd == -1){// if file not found then print error on sdterr
        perror("Coudln't Open The Input File !!!\n");
        exit(EXIT_FAILURE);
    }
    //*****************************************************************************************
```

- In my main thread I first create the the system v semaphore and initialize it to 0 so that consumers can’t consume in the first time or to say before the first item is available via supplier thread as:

```
//system V semaphore settings:

semid = semget(/*semaphore key*/ IPC_PRIVATE,/*semaphore count*/ 2,/*semaphore permissions*/  0777 | IPC_CREAT);
if (semid == -1){
    perror("semid semget() error");
    exit(EXIT_FAILURE);
}

sem_attr.val = 0;
if (semctl(/*semaphore set id*/ semid,/*semaphore set element index*/ 0,/*semaphore operations*/ SETVAL,/*semaphore union object*/ sem_attr) == -1){
    perror("semctl()");
    exit(EXIT_FAILURE);
}
if (semctl(/*semaphore set id*/ semid,/*semaphore set element index*/ 1,/*semaphore operations*/ SETVAL,/*semaphore union object*/ sem_attr) == -1){
    perror("semctl()");
    exit(EXIT_FAILURE);
}

if (sigintcaught == 1){// if sigint has recieved.
    if (semctl(/*semaphore set id*/ semid,/*semaphore set element index*/ 0,/*semaphore operations*/ IPC_RMID,sem_attr) == -1){
        perror("semctl() destroy semaphore");
        exit(EXIT_FAILURE);
    }
    close(inpfd);
    exit(EXIT_FAILURE);
}
//*****************************************************************************************
```

- Later I create the supplier thread making it detached and C consumer threads making them join-able as:

```
    // create supplier thread:
    s = pthread_attr_init(&attr);
    if (s != 0){
        perror("pthread_attr_init\n");
        exit(EXIT_FAILURE);
    }
    s = pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
    if (s != 0){
        perror("pthread_attr_setdetachstate");
        exit(EXIT_FAILURE);
    }

    int *arg = malloc(sizeof(*arg));
    if ( arg == NULL ) {
        perror("Couldn't allocate memory for arg.\n");
        exit(EXIT_FAILURE);
    }
    *arg = inpfd;
    s = pthread_create( &supplierThreadPointer, &attr, supplierThread,arg);
    if( s != 0){
        perror("pthread_create() error");
        exit(EXIT_FAILURE);
    }
    s = pthread_attr_destroy(&attr);
    if (s != 0){
        perror("pthread_attr_destroy");
        exit(EXIT_FAILURE);
    }

    if (sigintcaught == 1){// if sigint has recieved.
        if (semctl(/*semaphore set id*/ semid,/*semaphore set element index*/ 0,/*semaphore operations*/ IPC_RMID,sem_attr) == -1){
            perror("semctl() destroy semaphore");
            exit(EXIT_FAILURE);
        }
        close(inpfd);
        exit(EXIT_FAILURE);
    }
    //*****************************************************************************************

    // create consumer threads:
    pthread_t consumerThreadsPointersList[C];
    for (int i = 0; i < C; ++i){

        int *arg = malloc(sizeof(*arg));
        if ( arg == NULL ) {
            perror("Couldn't allocate memory for arg.\n");
            exit(EXIT_FAILURE);
        }
        *arg = i;
        s = pthread_create(&consumerThreadsPointersList[i],NULL, consumerThread,arg);
        if( s != 0){
            perror("pthread_create() error");
            exit(EXIT_FAILURE);
        }
    }
```

- Later I wait for the threads to terminate and exit the program:

```
// wait for the threads to terminate:
for (int i = 0; i < C; ++i){
    s = pthread_join(consumerThreadsPointersList[i],NULL);
    if (s != 0) {
        perror("pthread_join() error");
        exit(EXIT_FAILURE);
    }
}
//*************************************************************************************

close(inpfd);
if (semctl(/*semaphore set id*/ semid,/*semaphore set element index*/ 0,/*semaphore operations*/ IPC_RMID) == -1){
    perror("semctl() destroy semaphore");
    exit(EXIT_FAILURE);
}

pthread_exit(0);
```

**- Inside supplier thread:**

- I read the 1s and 2s from the file and when ever I read 1 I post the corresponding semaphore and do the same for 2 and keep doing that until all the items are read in the file as:

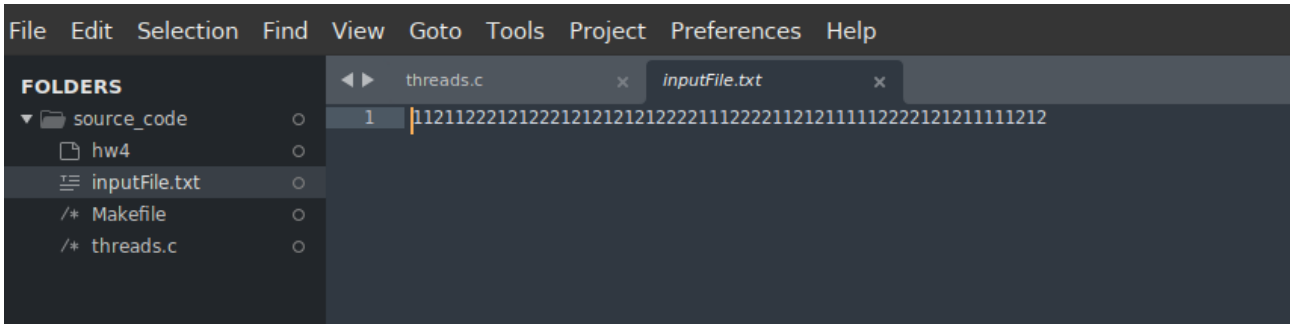
```
void *supplierThread(void *ptr){

    char buf[BUFFER_LIMIT],*message;
    int fd = *((int*) ptr), bytesread;
    time_t t;time(&t);

    for (;;){

        while((bytesread = read(fd,buf,BUFFER_LIMIT)) == -1 && errno == EINTR);// read bytes from the input file
        if (bytesread <= 0 ){// if the end of file is reached.
            break;
        }
        if (buf[0] == '1'){// supplier has read 1 from the file

            struct sembuf semOp;
            semOp.sem_num = 0;
            semOp.sem_op = 1;
            semOp.sem_flg = 0;

            message = (char*) malloc(MESSAGE_LEN * sizeof(char));
            sprintf(message,"[%.19s] Supplier: read from input a '%s'. Current amounts: %d x '1', %d x '2'\n",ctime(&t),buf,semctl(semid,0,GETVAL),semctl(semid,1,GETVAL));
            printMessage(message);

            if (semop(semid, &semOp, 1) == -1){// perfrom operation on semaphore set
                perror("semop on semid");
                exit(EXIT_FAILURE);
            }
        }else if (buf[0] == '2'){// supplier has read 2 from the file

            struct sembuf semOp;
            semOp.sem_num = 1;
            semOp.sem_op = 1;
            semOp.sem_flg = 0;
            message = (char*) malloc(MESSAGE_LEN * sizeof(char));
            sprintf(message,"[%.19s] Supplier: read from input a '%s'. Current amounts: %d x '1', %d x '2'\n",ctime(&t),buf,semctl(semid,0,GETVAL),semctl(semid,1,GETVAL));
            printMessage(message);

            if (semop(semid, &semOp, 1) == -1){// perfrom operation on semaphore set
                perror("semop on semid");
                exit(EXIT_FAILURE);
            }
        }
        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Supplier: delivered a '%s'. Post-delivery amounts: %d x '1', %d x '2'\n",ctime(&t),buf,semctl(semid,0,GETVAL),semctl(semid,1,GETVAL));
        printMessage(message);

    }

    message = (char*) malloc(MESSAGE_LEN * sizeof(char));
    sprintf(message,"[%.19s] The Supplier has left.\n",ctime(&t));
    printMessage(message);

    free(ptr);
    return(NULL);
}
```

**-Inside Consumer threads:**

- I wait for both the 1 and 2 to be available via supplier and if both are available then I consume them and continue waiting as:

```c
void *consumerThread(void *ptr){

    int ptrLocal = *((int *) ptr);
    char *message;
    time_t t;time(&t);

    struct sembuf semOp[2];
    semOp[0].sem_num = 0;
    semOp[0].sem_op = -1;
    semOp[0].sem_flg = 0;

    semOp[1].sem_num = 1;
    semOp[1].sem_op = -1;
    semOp[1].sem_flg = 0;

    for (int i = 0; i < N; ++i){


        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Consumer-%d at iteration %d (waiting). Current amounts: %d x '1', %d x '2'\n",ctime(&t),ptrLocal,i,semctl(semid,0,GETVAL),semctl(semid,1,GETVAL));
        printMessage(message);

        if (semop(semid, semOp, 2) == -1){// perfrom operation on semaphore set
            perror("semop on semid");
            exit(EXIT_FAILURE);
        }

        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Consumer-%d at iteration %d (consumed). Post-consumption amounts: %d x '1', %d x '2'\n",ctime(&t),ptrLocal,i,semctl(semid,0,GETVAL),semctl(semid,1,GETVAL));
        printMessage(message);

    }

    free(ptr);

    message = (char*) malloc(MESSAGE_LEN * sizeof(char));
    sprintf(message,"[%.19s] Consumer-%d has left\n",ctime(&t),ptrLocal);
    printMessage(message);

    return(NULL);
}
```

## INPUT FILE AS:




## SCREEN SHOTS FROM THE PROGRAMS:

```
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 6 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 6 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Consumer-4 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Consumer-4 at iteration 2 (waiting). Current amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-6 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-6 at iteration 2 (waiting). Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-0 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-0 at iteration 2 (waiting). Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 5 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-3 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-3 at iteration 2 (waiting). Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-1 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-1 at iteration 2 (waiting). Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Consumer-5 at iteration 1 (consumed). Post-consumption amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Consumer-5 at iteration 2 (waiting). Current amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Consumer-2 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Consumer-2 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Consumer-9 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-8 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-8 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Consumer-7 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-7 has left
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 4 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-4 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-4 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 3 x '2'
[Sat May 14 05:08:11] Consumer-6 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Consumer-6 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 2 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Consumer-0 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 1 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Consumer-0 has left
[Sat May 14 05:08:11] Consumer-3 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Consumer-3 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Consumer-1 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Consumer-1 has left
[Sat May 14 05:08:11] Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
[Sat May 14 05:08:11] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] The Supplier has left.
[Sat May 14 05:08:11] Consumer-5 at iteration 2 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:08:11] Consumer-5 has left
==186705==
==186705== HEAP SUMMARY:
==186705==     in use at exit: 0 bytes in 0 blocks
==186705==   total heap usage: 418 allocs, 418 frees, 204,474 bytes allocated
==186705==
==186705== All heap blocks were freed -- no leaks are possible
==186705==
==186705== For lists of detected and suppressed errors, rerun with: -s
==186705== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw4/source_code$
```

## TESTING SIGINT CASE:

```
[Sat May 14 05:10:17] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 1 x '2'
[Sat May 14 05:10:17] Supplier: read from input a '1'. Current amounts: 0 x '1', 1 x '2'
[Sat May 14 05:10:17] Supplier: delivered a '1'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-43 at iteration 0 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-44 at iteration 0 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-44 at iteration 1 (waiting). Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-43 at iteration 1 (waiting). Current amounts: 1 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-40 at iteration 0 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-40 at iteration 1 (waiting). Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: read from input a '1'. Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: delivered a '1'. Post-delivery amounts: 1 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: read from input a '2'. Current amounts: 1 x '1', 0 x '2'
[Sat May 14 05:10:17] Supplier: delivered a '2'. Post-delivery amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-41 at iteration 0 (consumed). Post-consumption amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:18] Consumer-41 at iteration 1 (waiting). Current amounts: 0 x '1', 0 x '2'
[Sat May 14 05:10:17] The Supplier has left.
^Csemop on semid: Interrupted system call
```