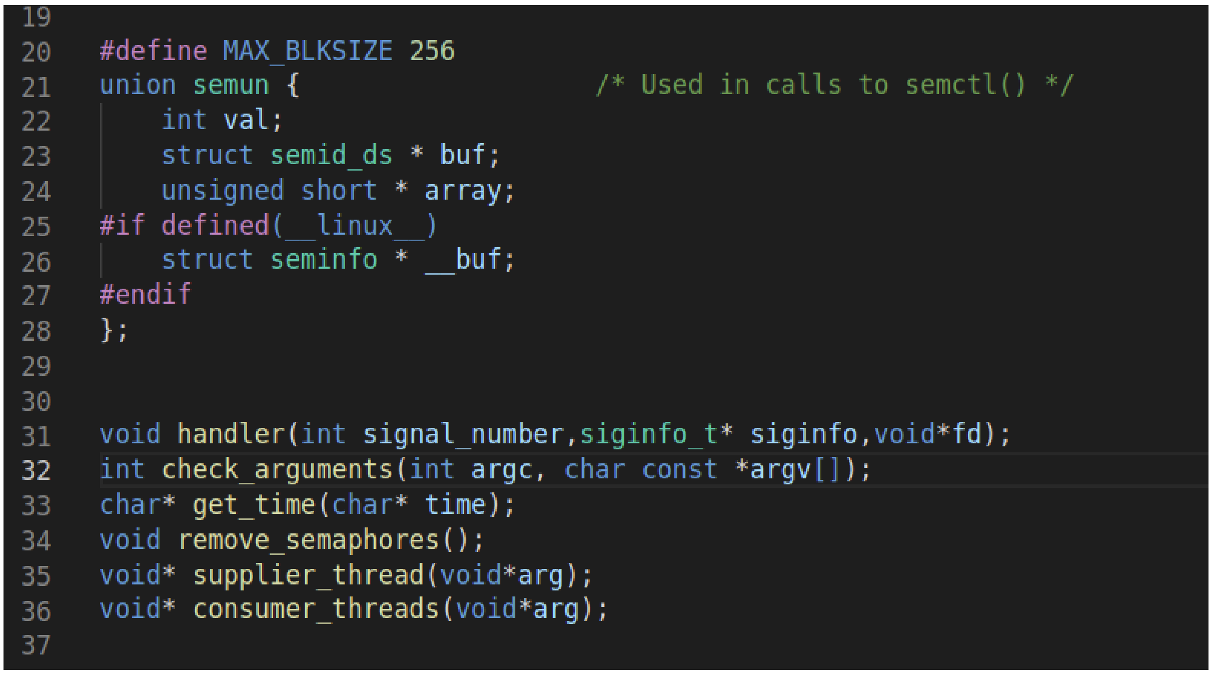
CSE 344 Systems Programming Homework 4 Report

Yakup Talha Yolcu
1801042609

header.h



```c
#define MAX_BLKSIZE 256
union semun {                        /* Used in calls to semctl() */
    int val;
    struct semid_ds * buf;
    unsigned short * array;
#if defined(__linux__)
    struct seminfo * __buf;
#endif
};


void handler(int signal_number,siginfo_t* siginfo,void*fd);
int check_arguments(int argc, char const *argv[]);
char* get_time(char* time);
void remove_semaphores();
void* supplier_thread(void*arg);
void* consumer_threads(void*arg);
```

This union is fixed for system V semaphores. semaphore values are stored in array as unsigned short variables.

I have signal handler to handle SIGINT.

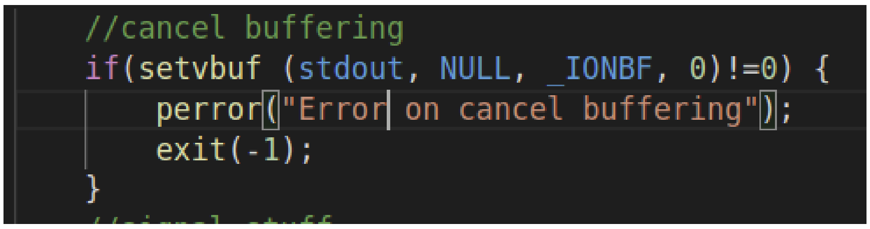
I have 1 function to represent all the consumer threads and 1 function to represent supplier thread.

I have check arguments function, takes directly argc and argv from main and looks for argument count and letters after - sign.

I have get_time function which takes ctime(&curtime)) string. My function erases \n of this string and returns it back.

remove semaphores function just calls semctl (IPC_RMID).

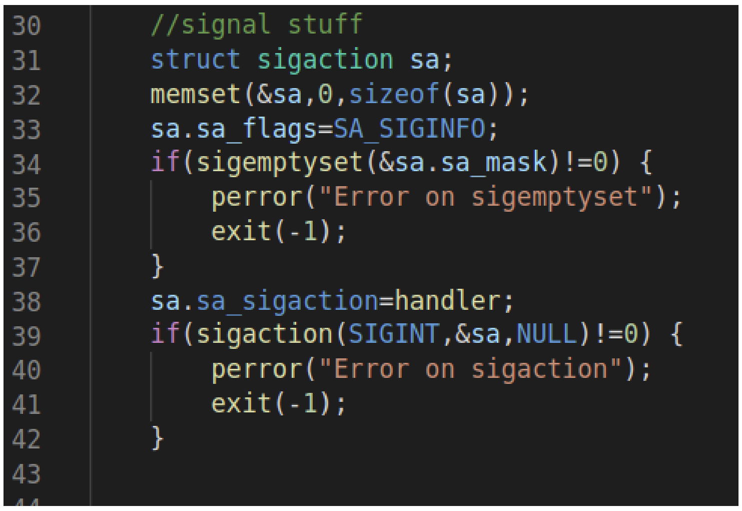
How I solved buffering?



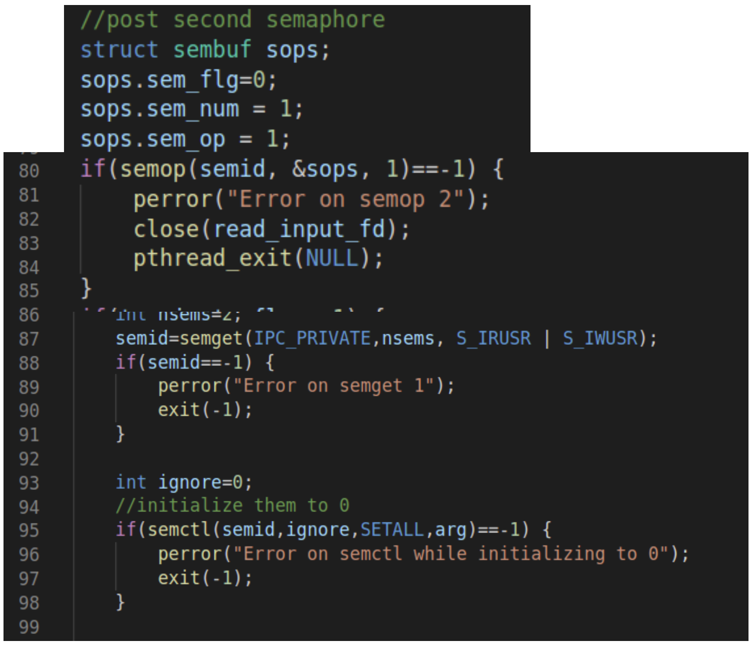
```c
    //cancel buffering
    if(setvbuf (stdout, NULL, _IONBF, 0)!=0) {
        perror("Error on cancel buffering");
        exit(-1);
    }
```

Signal handling

Creating and initializing semaphores



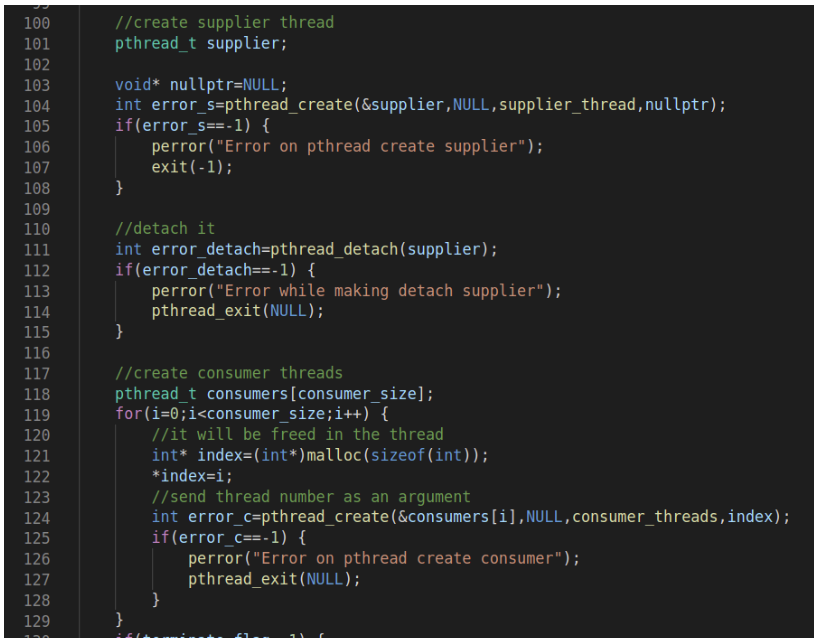
```c
//signal stuff
struct sigaction sa;
memset(&sa,0,sizeof(sa));
sa.sa_flags=SA_SIGINFO;
if(sigemptyset(&sa.sa_mask)!=0) {
    perror("Error on sigemptyset");
    exit(-1);
}
sa.sa_sigaction=handler;
if(sigaction(SIGINT,&sa,NULL)!=0) {
    perror("Error on sigaction");
    exit(-1);
}
```



```c
//post second semaphore
struct sembuf sops;
sops.sem_flg=0;
sops.sem_num = 1;
sops.sem_op = 1;
if(semop(semid, &sops, 1)==-1) {
    perror("Error on semop 2");
    close(read_input_fd);
    pthread_exit(NULL);
}
```

```c
    int nsems=2;
    semid=semget(IPC_PRIVATE,nsems, S_IRUSR | S_IWUSR);
    if(semid==-1) {
        perror("Error on semget 1");
        exit(-1);
    }

    int ignore=0;
    //initialize them to 0
    if(semctl(semid,ignore,SETALL,arg)==-1) {
        perror("Error on semctl while initializing to 0");
        exit(-1);
    }
```

Creating all threads and detach supplier thread

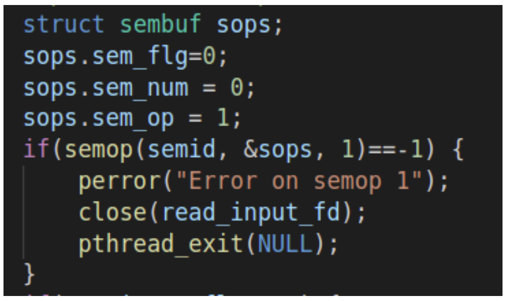


```c
    //create supplier thread
    pthread_t supplier;

    void* nullptr=NULL;
    int error_s=pthread_create(&supplier,NULL,supplier_thread,nullptr);
    if(error_s==-1) {
        perror("Error on pthread create supplier");
        exit(-1);
    }

    //detach it
    int error_detach=pthread_detach(supplier);
    if(error_detach==-1) {
        perror("Error while making detach supplier");
        pthread_exit(NULL);
    }

    //create consumer threads
    pthread_t consumers[consumer_size];
    for(i=0;i<consumer_size;i++) {
        //it will be freed in the thread
        int* index=(int*)malloc(sizeof(int));
        *index=i;
        //send thread number as an argument
        int error_c=pthread_create(&consumers[i],NULL,consumer_threads,index);
        if(error_c==-1) {
            perror("Error on pthread create consumer");
            pthread_exit(NULL);
        }
    }
```

After that I call pthread join for each consumer thread, then I removed semaphores and call pthread_exit function

Supplier thread

Supplier thread opens the file. File path is hold in global variable. After opening file it reads byte by byte the file. If it reads 1, it posts semaphore respect to 1. It is also true for 2.



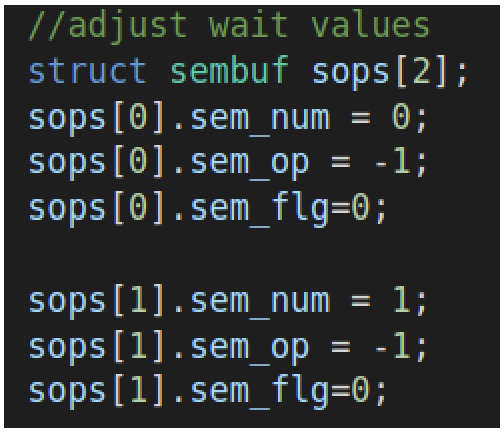
```c
struct sembuf sops;
sops.sem_flg=0;
sops.sem_num = 0;
sops.sem_op = 1;
if(semop(semid, &sops, 1)==-1) {
    perror("Error on semop 1");
    close(read_input_fd);
    pthread_exit(NULL);
}
```

Posting ‘1’ semaphore

Posting ‘2’ semaphore

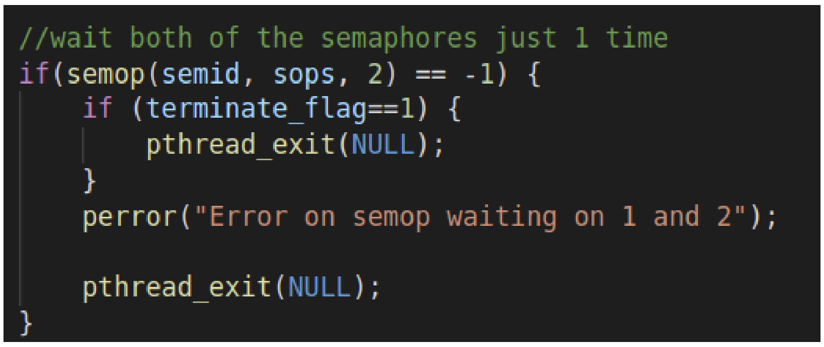
Consumer thread

It takes consumer index number as an argument. Then it starts the loop for n times. In the loop, firstly it adjusts the waiting values for semaphores.



```c
//adjust wait values
struct sembuf sops[2];
sops[0].sem_num = 0;
sops[0].sem_op = -1;
sops[0].sem_flg=0;

sops[1].sem_num = 1;
sops[1].sem_op = -1;
sops[1].sem_flg=0;
```

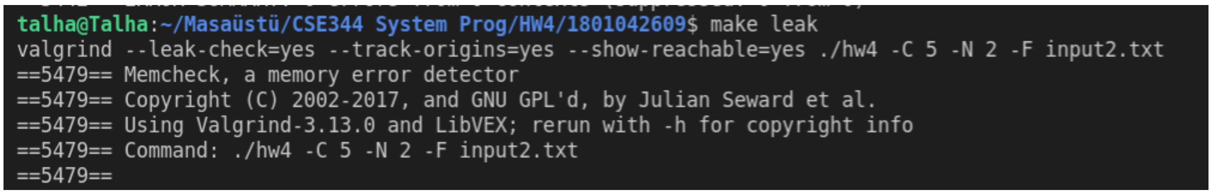
After doing it, it waits for both semaphores.



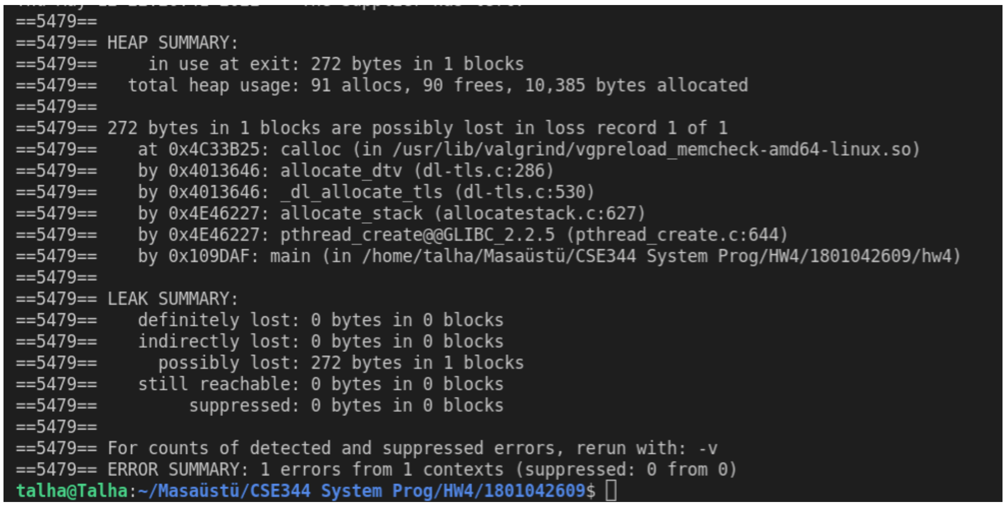
```c
//wait both of the semaphores just 1 time
if(semop(semid, sops, 2) == -1) {
    if (terminate_flag==1) {
        pthread_exit(NULL);
    }
    perror("Error on semop waiting on 1 and 2");

    pthread_exit(NULL);
}
```

Then loop continues until n.

Valgrind output



```
talha@Talha:~/Masaüstü/CSE344 System Prog/HW4/1801042609$ make leak
valgrind --leak-check=yes --track-origins=yes --show-reachable=yes ./hw4 -C 5 -N 2 -F input2.txt
==5479== Memcheck, a memory error detector
==5479== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5479== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==5479== Command: ./hw4 -C 5 -N 2 -F input2.txt
==5479==
```



```
==5479==
==5479== HEAP SUMMARY:
==5479==     in use at exit: 272 bytes in 1 blocks
==5479==   total heap usage: 91 allocs, 90 frees, 10,385 bytes allocated
==5479==
==5479== 272 bytes in 1 blocks are possibly lost in loss record 1 of 1
==5479==    at 0x4C33B25: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==5479==    by 0x4013646: allocate_dtv (dl-tls.c:286)
==5479==    by 0x4013646: _dl_allocate_tls (dl-tls.c:530)
==5479==    by 0x4E46227: allocate_stack (allocatestack.c:627)
==5479==    by 0x4E46227: pthread_create@@GLIBC_2.2.5 (pthread_create.c:644)
==5479==    by 0x109DAF: main (in /home/talha/Masaüstü/CSE344 System Prog/HW4/1801042609/hw4)
==5479==
==5479== LEAK SUMMARY:
==5479==    definitely lost: 0 bytes in 0 blocks
==5479==    indirectly lost: 0 bytes in 0 blocks
==5479==      possibly lost: 272 bytes in 1 blocks
==5479==    still reachable: 0 bytes in 0 blocks
==5479==         suppressed: 0 bytes in 0 blocks
==5479==
==5479== For counts of detected and suppressed errors, rerun with: -v
==5479== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
talha@Talha:~/Masaüstü/CSE344 System Prog/HW4/1801042609$
```