**OPERATING SYSTEM ASSIGNMENT # 3**

**REPORT**

**GROUP MEMBERS:**

* Muhibullah Sherwani
* Faraz Ahmed
* Faris Jamil

Class ID: 101348

Date: 27-April-2019

**CODE EXPLANATION:**

**ChatServer.c**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include<stdio.h>

#include<string.h>

#include<errno.h>

#include<stdlib.h>

#define BUF\_SIZE 1024

#define SHM\_KEY 0x1234

// Creation Of Shared Memory

int shmget(key\_t key, size\_t size, int shmflg)

// Creation Of Semaphores

int semget(key\_t key, int num\_sems, int sem\_flags);

// Deletion Of Shared Memory

if (shmdt(shared\_memory) == -1) {

fprintf(stderr, “shmdt failed\n”);

exit(EXIT\_FAILURE);

}

if (shmctl(shmid, IPC\_RMID, 0) == -1) {

fprintf(stderr, “shmctl(IPC\_RMID) failed\n”);

exit(EXIT\_FAILURE);

}

exit(EXIT\_SUCCESS);

}

// Deletion Of Semaphores

void remove\_semaphore() {

int semid;

int retval;

semid = semget(SEM\_KEY, 1, 0);

if (semid < 0) {

perror("Remove Semaphore: Semaphore GET: ");

return;

}

retval = semctl(semid, 0, IPC\_RMID);

if (retval == -1) {

perror("Remove Semaphore: Semaphore CTL: ");

return;

}

return;

}

// Reading Message from message board

struct shmseg {

int cnt;

int complete;

char buf[BUF\_SIZE];

};

int shmid;

struct shmseg \*shmp;

shmid = shmget(SHM\_KEY, sizeof(struct shmseg), 0644|IPC\_CREAT);

if (shmid == -1) {

perror("Shared memory");

return 1;

}

// Attach to the segment to get a pointer to it.

shmp = shmat(shmid, NULL, 0);

if (shmp == (void \*) -1) {

perror("Shared memory attach");

return 1;

}

/\* Transfer blocks of data from shared memory to stdout\*/

while (shmp->complete != 1) {

printf("segment contains : \n\"%s\"\n", shmp->buf);

if (shmp->cnt == -1) {

perror("read");

return 1;

}

printf("Reading Process: Shared Memory: Read %d bytes\n", shmp->cnt);

sleep(3);

}

printf("Reading Process: Reading Done, Detaching Shared Memory\n");

if (shmdt(shmp) == -1) {

perror("shmdt");

return 1;

}

printf("Reading Process: Complete\n");

return 0;

// Writing Message to Message board

int fill\_buffer(char \* bufptr, int size);

/\* Transfer blocks of data from buffer to shared memory \*/

bufptr = shmp->buf;

spaceavailable = BUF\_SIZE;

for (numtimes = 0; numtimes < 5; numtimes++) {

shmp->cnt = fill\_buffer(bufptr, spaceavailable);

shmp->complete = 0;

printf("Writing Process: Shared Memory Write: Wrote %d bytes\n", shmp->cnt);

bufptr = shmp->buf;

spaceavailable = BUF\_SIZE;

sleep(3);

}

printf("Writing Process: Wrote %d times\n", numtimes);

shmp->complete = 1;

if (shmdt(shmp) == -1) {

perror("shmdt");

return 1;

}

if (shmctl(shmid, IPC\_RMID, 0) == -1) {

perror("shmctl");

return 1;

}

printf("Writing Process: Complete\n");

return 0;

}

int fill\_buffer(char \* bufptr, int size) {

static char ch = 'A';

int filled\_count;

//printf("size is %d\n", size);

memset(bufptr, ch, size - 1);

bufptr[size-1] = '\0';

if (ch > 122)

ch = 65;

if ( (ch >= 65) && (ch <= 122) ) {

if ( (ch >= 91) && (ch <= 96) ) {

ch = 65;

}

}

filled\_count = strlen(bufptr);

printf("buffer count is: %d\n", filled\_count);

printf("buffer filled is:%s\n", bufptr);

ch++;

return filled\_count;

}

// Creating Client List

int running = 1;

**ClientChat.c**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include<stdio.h>

#include<string.h>

#include<errno.h>

#include<stdlib.h>

#define BUF\_SIZE 1024

#define SHM\_KEY 0x1234

// Creaton Of Shared Memory

int shmget(key\_t key, size\_t size, int shmflg)

// Creation Of Semaphore

int semget(key\_t key, int num\_sems, int sem\_flags);

// Changing Value of Semaphore

int semop(int sem\_id, struct sembuf \*sem\_ops, size\_t num\_sem\_ops);

// Deletion Of Shared Memory

if (shmdt(shared\_memory) == -1) {

fprintf(stderr, “shmdt failed\n”);

exit(EXIT\_FAILURE);

}

if (shmctl(shmid, IPC\_RMID, 0) == -1) {

fprintf(stderr, “shmctl(IPC\_RMID) failed\n”);

exit(EXIT\_FAILURE);

}

exit(EXIT\_SUCCESS);

}

// Deletion Of Semaphores

void remove\_semaphore() {

int semid;

int retval;

semid = semget(SEM\_KEY, 1, 0);

if (semid < 0) {

perror("Remove Semaphore: Semaphore GET: ");

return;

}

retval = semctl(semid, 0, IPC\_RMID);

if (retval == -1) {

perror("Remove Semaphore: Semaphore CTL: ");

return;

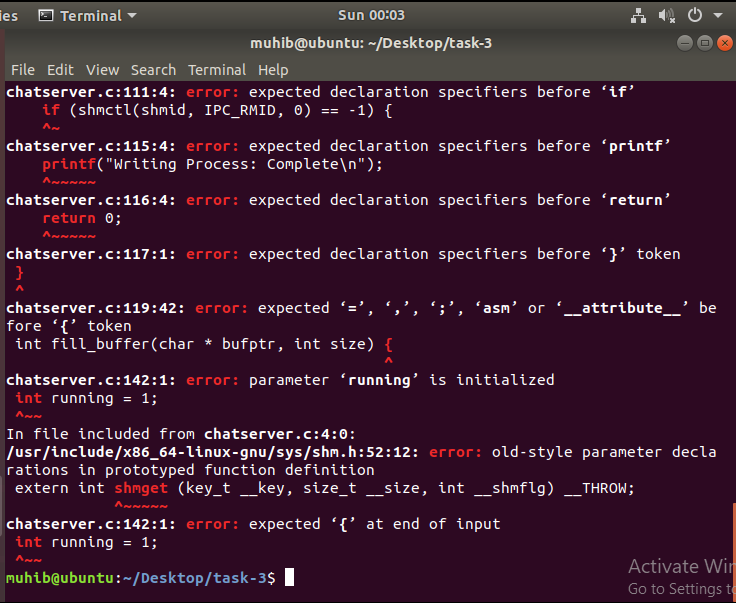
}

return;

}

**CODE EXCECUTION SCREENSHOT:**

Note: Errors in code execution but we done some code explanation



**MANUALLY CODE EXCECUTION SCREENSHOT:**

