МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский ядерный университет «МИФИ»

(НИЯУ МИФИ)

Институт интеллектуальных кибернетических систем

Кафедра Кибернетики

**Лабораторная работа №3 по курсу**

**«Разработка ПО ОС UNIX»**

**Выполнил студент группы Б15-501:** Огнянович Павел

**Проверил:**  Ктитров С.В.

Москва, 2018

**Задание**

Разработать программу для Solaris, реализующую центральную доску объявлений. С помощью первой программы в разделяемой памяти публикуется объявление, а программы-клиенты отображают его, причем при изменении клиенты обновляют текст. Программа должна собираться из нескольких файлов с использованием make.

**Код программы**

Lab-3-server.c

#include <stdlib.h>

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <sys/mman.h>

#include <pthread.h>

#include <string.h>

#define SH\_MEM\_FILE\_NAME "lab-3-server.c"

#define SH\_MEM\_SIZE 1120

#define PROJ\_ID 12

#define BUFF\_SIZE 1024

int main(int argc, char \*argv[]) {

int smesch = sizeof(pthread\_mutex\_t)

+sizeof(pthread\_cond\_t)

+sizeof(int)

+sizeof(int);

pthread\_mutex\_t mutex;

pthread\_mutex\_init(&mutex, NULL);

pthread\_cond\_t cond;

pthread\_cond\_init(&cond, NULL);

int has\_new = 0; //flag for new messages

int message\_size = 0; //current size of message

int message\_number = -1; //number of message

char buffer[BUFF\_SIZE]; //buffer for message

char \*sh\_mem; //pointer to shared memory

int shmid;

key\_t key;

// generating key

//printf(">>>key = %d\n", key);

if((key = ftok(SH\_MEM\_FILE\_NAME, 12)) < 0) {

//printf(">>>key = %d\n",key);

printf(">>>FTOK\_ERROR!\n");

return 1;

}

// creating shared memory

if((shmid = shmget(key, SH\_MEM\_SIZE\*sizeof(char), 0666 | IPC\_CREAT)) < 0) {

//printf(">>>shmid = %d\n", shmid);

printf(">>>SHMGET\_ERROR!\n");

return 1;

}

// getting pointer to shared memory

if((sh\_mem = (int\*)shmat(shmid, NULL, 0)) == (int\*)(-1)) {

//printf(">>>sh\_mem = %d\n", sh\_mem);

printf(">>>SHMAT\_ERROR!\n");

return 1;

}

//TEST\_OF\_SH\_MEM #1

// int test = 100500;

// memcpy(sh\_mem, &test, sizeof(int));

// int res = 0;

// memcpy(&res, sh\_mem, sizeof(int));

// printf(">>>test = %d\n>>>res = %d\n", test, res);

// copying mutex to shared memory

printf(">>>COPYING\_<MUTEX>\_TO: sh\_mem[0]\n");

memcpy(sh\_mem, &mutex, sizeof(pthread\_mutex\_t));

// copying cond to shared memory

printf(">>>COPYING\_<COND>\_TO: sh\_mem[%ld]\n", sizeof(pthread\_mutex\_t));

memcpy(sh\_mem+sizeof(pthread\_mutex\_t), &cond, sizeof(pthread\_cond\_t));

// copying message\_size to shared memory

printf(">>>COPYING\_<MESSAGE\_SIZE>\_TO: sh\_mem[%ld]\n", sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t));

memcpy(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t), &message\_size, sizeof(int));

// copying message\_number to shared memory

printf(">>>COPYING\_<MESSAGE\_NUMBER>\_TO: sh\_mem[%ld]\n", sizeof(pthread\_mutex\_t));

memcpy(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t)+sizeof(int), &message\_number, sizeof(pthread\_cond\_t));

//TEST\_OF\_SH\_MEM #2

// char \*messag = "hello world!";

// memcpy(sh\_mem+smesch, messag, 12\*sizeof(char));

// char \*res;

// memcpy(res, sh\_mem+smesch, 12\*sizeof(char));

// printf(">>>RESULT:\n>>>");

// for(int i=0;i<12;i++)

// printf("%c",res[i]);

// printf("\n");

for(;;) {

printf(">>>1 - new message;\n>>>2 - exit\n");

char symb;

switch(symb = getchar()) {

case '1':

getchar();

pthread\_mutex\_lock((pthread\_mutex\_t\*)sh\_mem);

//-------------CREATE\_MESSAGE-------

message\_size = 0;

message\_number++;

printf(">>>MESSAGE\_NUMBER: %d\n", message\_number);

printf(">>>ENTER\_MESSAGE:\n");

// memset(buffer, 0, BUFF\_SIZE\*sizeof(char));

// for(int i=0;i<BUFF\_SIZE;i++) {

// message\_size++;

// if((buffer[i]=getchar())=='\n')

// break;

// }

create\_message(buffer, (int)BUFF\_SIZE, &message\_size);

printf(">>>MESSAGE\_SIZE: %d\n", message\_size);

//----------------------------------

//copying message

memcpy(sh\_mem+smesch,

buffer,

BUFF\_SIZE\*sizeof(char));

//copying message\_size

memcpy(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t),

&message\_size,

sizeof(int));

//copying message\_number

memcpy(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t)+sizeof(int),

&message\_number,

sizeof(int));

pthread\_cond\_broadcast((pthread\_cond\_t\*)(sh\_mem+sizeof(pthread\_mutex\_t)));

pthread\_mutex\_unlock((pthread\_mutex\_t\*)sh\_mem);

break;

case '2':

pthread\_mutex\_lock((pthread\_mutex\_t\*)sh\_mem);

message\_number = -1;

//setting buffer in sh\_mem to 0

memset(sh\_mem+smesch, 0, BUFF\_SIZE\*sizeof(char));

//setting message\_size to 0

memset(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t), 0, sizeof(int));

memcpy(sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t)+sizeof(int),

&message\_number,

sizeof(int));

pthread\_cond\_broadcast((pthread\_cond\_t\*)(sh\_mem+sizeof(pthread\_mutex\_t)));

pthread\_mutex\_unlock((pthread\_mutex\_t\*)sh\_mem);

if(shmdt(sh\_mem)<0) {

printf(">>>SHMDT\_ERROR!\n");

return 1;

}

return 0;

break;

default:

printf(">>>UNKNOWN\_COMMAND!\n");

if(shmdt(sh\_mem)<0)

printf(">>>SHMDT\_ERROR!\n");

return 1;

}

}

if(shmdt(sh\_mem)<0) {

printf(">>>SHMDT\_ERROR!\n");

return 1;

}

return 0;

}

Lab-3-client.c

#include <stdlib.h>

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <sys/mman.h>

#include <pthread.h>

#include <string.h>

#define SH\_MEM\_FILE\_NAME "lab-3-server.c"

#define SH\_MEM\_SIZE 1120

#define PROJ\_ID 12

#define BUFF\_SIZE 1024

int main(int argc, char \*argv[]) {

int has\_new = 0;

int message\_size = 0;

int message\_number = -1;

int message\_number\_old = -1;

char \*sh\_mem;

int shmid;

key\_t key;

// generating key

// printf(">>>key = %d\n", key);

if((key = ftok(SH\_MEM\_FILE\_NAME, 12)) < 0) {

// printf(">>>key = %d\n",key);

printf(">>>FTOK\_ERROR!\n");

return 1;

}

// creating shared memory

if((shmid = shmget(key, SH\_MEM\_SIZE\*sizeof(char), 0666)) < 0) {

// printf(">>>shmid = %d\n", shmid);

printf(">>>SHMGET\_ERROR!\n");

return 1;

}

// getting pointer to shared memory

if((sh\_mem = (int\*)shmat(shmid, NULL, 0)) == (int\*)(-1)) {

// printf(">>>sh\_mem = %d\n", sh\_mem);

printf(">>>SHMAT\_ERROR!\n");

return 1;

}

for(;;) {

pthread\_mutex\_lock((pthread\_mutex\_t\*)sh\_mem);

memcpy(&message\_number, sh\_mem+92, sizeof(int));

while(message\_number == sh\_mem[92])

pthread\_cond\_wait((pthread\_cond\_t\*)(sh\_mem+sizeof(pthread\_mutex\_t)), (pthread\_mutex\_t\*)sh\_mem);

memcpy(&message\_number,

sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t)+sizeof(int),

sizeof(int));

if(message\_number == -1) {

printf(">>>SERVER\_DISCONNECTED!\n");

printf(">>>MESSAGE\_NUMBER: -1\n");

} else {

printf(">>>GETTING\_MESSAGE:\n");

printf(">>>MESSAGE\_NUMBER: %d\n", message\_number);

memcpy(&message\_size,

sh\_mem+sizeof(pthread\_mutex\_t)+sizeof(pthread\_cond\_t),

sizeof(int));

printf(">>>MESSAGE\_SIZE = %d\n", message\_size);

for(int i=0;i<message\_size;i++) {

printf("%c", (sh\_mem+96)[i]);

}

}

pthread\_mutex\_unlock((pthread\_mutex\_t\*)sh\_mem);

}

return 0;

}

function.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

void create\_message(char \*buff, int buffer\_size, int \*msg\_size) {

memset(buff, 0, buffer\_size\*sizeof(char));

//printf(">%d\n", buffer\_size);

for(int i=0;i<buffer\_size;i++) {

msg\_size[0]++;

//printf("CR\_M>>>msg\_size=%d\n", msg\_size[0]);

if((buff[i]=getchar())=='\n')

break;

}

}

makefile

all: lab-3-server.exe lab-3-client.exe

lab-3-server.exe: lab-3-server.o function.o

gcc lab-3-server.o function.o -o lab-3-server.exe

functoin.o: function.c

gcc -c finction.c

lab-3-server.o: lab-3-server.c

gcc -c lab-3-server.c

lab-3-client.exe:

gcc lab-3-client.o -o lab-3-client.exe

lab-3-client.o: lab-3-client.c

gcc -c lab-3-client.c

clean:

rm -rf \*.o