Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7

По дисциплине: «ОСиСП»

Тема: «Семафоры»

Выполнил:

Студент 2 курса

Группы ПО-7

Комиссаров А.Е.

Проверила:

Давидюк Ю.И.

Брест 2022

**Цель работы:** изучить основные принципы работы, ознакомиться с семафорами.

**Задание :**

Ознакомиться с руководством, теоретическими сведениями и лекционным материалом по использованию и функционированию средств синхронизации - семафоров Дейкстры, и их реализацией в Linux - System V IPC семафоры и POSIX-семафоры.

Написать две (или более) программы, которые, работая параллельно зацикленно, обмениваются информацией согласно варианту. Передачу и получение информации каждым

из процессов сопровождать выводом на экран информации типа "процесс такой-то

передал/получил такую-то информацию". Синхронизацию работы процессов реализовать с

помощью семафоров. Учтите, что при организации совместного доступа к разделяемому

ресурсу (например, файлу) вам понадобится применять мьютексы.

Для наглядности запускайте свои процессы в разных окнах терминала. Запустите

программы в нескольких экземплярах (одну первую и две/три вторых, две первых и две

вторых...).

**Вариант 6.** Первый процесс в цикле ожидает ввода символа с потока stdin, после чего

пишет в файл соответствующий символ случайное количество раз, каждый раз открывая и

закрывая за собой файл. Второй процесс забирает из файла символы и выводит на экран их

количество.

**Код программы:**

#include <unistd.h>

#include <iostream>

#include <sys/mman.h>

#include <sys/stat.h>

#include <bits/stdc++.h>

#include <string.h>

#include <pthread.h>

#include <sys/types.h>

#include <signal.h>

#include <fcntl.h>

const char \*filename = "strings.txt";

using namespace std;

void handler(int signal){

if (signal == SIGUSR1){cout << "<handler> received SIGUSR1." << endl;}

else if (signal == SIGUSR2){cout << "<handler> received SIGUSR2." << endl;}

else {cout << "<handler> received other signals." << endl;}

}

void parent\_write(int argc, char \*\*argv)

{

int oflags = O\_RDWR | O\_CREAT | O\_TRUNC;

//generating message to write in the file

string str1 = "mystring1 is here";

string str2 = "this is string 2";

string str3 = "string 3 is over here";

string message = str1 + "\n" + str2 + "\n" + str3;

off\_t length = message.length();

//

cout << "[parent] accessing / creating shared file." << endl;

int fd = shm\_open(filename, oflags, 0666);

ftruncate(fd, length);

u\_char \*ptr = (u\_char \*)mmap(NULL, length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

//

cout << "[parent] generated message: " << endl << message << endl;

strcpy((char \*)ptr, &message[0]);

cout << "[parent] closing file." << endl;

close(fd);

}

void parent\_read()

{

int oflags = O\_RDWR;

int fd = shm\_open(filename, oflags, 0644);

struct stat state;

fstat(fd, &state);

off\_t length = state.st\_size;

cout << "[parent] opening shared file." << endl;

u\_char \*ptr = (u\_char \*)mmap(NULL, length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

cout << "[parent] reading shared file." << endl;

string received = "";

for (size\_t i = 0; i < length; i++)

{ if (ptr[i] == '\n')

{ break; }

received += ptr[i];

} cout << "[parent] received string : " << received << endl;

}

bool isSogl(char ch){ ch = toupper(ch);

return (ch == 'Q' || ch == 'W' || ch == 'R' ||

ch == 'T' || ch == 'P' || ch == 'S' ||

ch == 'D' || ch == 'F' || ch == 'G' ||

ch == 'H' || ch == 'J' || ch == 'K' ||

ch == 'L' || ch == 'Z' || ch == 'X' ||

ch == 'C' || ch == 'V' || ch == 'B' ||

ch == 'N' || ch == 'M' );

}

int countSogl(string str){

int count = 0;

for (int i = 0; i < str.length(); i++){

if (isSogl(str[i])) {++count;}

} return count;

}

void child\_wread()

{

string \*strings = new string[3];

int best[3];

int fd = shm\_open(filename, O\_RDWR, 0644);

struct stat state;

fstat(fd, &state);

off\_t length = state.st\_size;

u\_char \*ptr = (u\_char \*)mmap(NULL, length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

cout << "[[child]] reading shared file." << endl;

string fileString = "";

int currentStringsCount = 0;

for (size\_t i = 0; i < length; i++)

{ if (ptr[i] == '\n')

{

currentStringsCount++;

continue;

}

strings[currentStringsCount] += ptr[i];

}

cout << "[[child]] received string 1 : " << strings[0] << endl;

cout << "[[child]] received string 2 : " << strings[1] << endl;

cout << "[[child]] received string 3 : " << strings[2] << endl;

string message = strings[0];

string soglasniye = "qwrtpsdfghjklzxcvbnmQWRTPSDFGHJKLZXCVBNM";

for (int i = 0; i < 3; i++) {

best[i] = countSogl(strings[i]);

}

int maxSogl = \*max\_element(best, best+3);

for (int i = 0; i < 3; i++){

if (maxSogl == best[i]){message = strings[i];}

}

cout << "[[child]] sent string : " << endl << message << endl;

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

ptr[i] = 0;

}

strcpy((char \*)ptr, &message[0]);

cout << "[[child]] closing shared file." << endl;

close(fd);

}

int main(int argc, char \*\*argv)

{

srand(time(NULL)); //random number generator seed = current time.

struct sigaction action;

memset(&action, 0, sizeof(action));

action.sa\_handler = handler;

sigset\_t set;

sigemptyset(&set); //emptying the sigset

//adding signals to set

sigaddset(&set, SIGUSR1);

sigaddset(&set, SIGUSR2);

//adding masking with set, linking action to signals

action.sa\_mask = set;

sigaction(SIGUSR1, &action, 0);

sigaction(SIGUSR2, &action, 0);

signal(SIGUSR1, handler);

int signal;

pid\_t pid;

pid = fork();

if (pid > 0){

cout << "[parent] started, process PID : " << getpid() << "." << endl;

cout << "[parent] writing to shared file..." << endl;

parent\_write(argc, argv);

sleep(1);

cout << "[parent] done writing, sending SIGUSR1 to child." << endl;

kill(pid, SIGUSR1);

sigemptyset(&set);

sigaddset(&set, SIGUSR2);

cout << "[parent] waiting for signals..." << endl;

sigwait(&set, &signal);

cout << "[parent] received SIGUSR2." << endl;

cout << "[parent] reading from shared file..." << endl;

parent\_read();

cout << "[parent] done working, exiting." << endl;

}

else if (pid == 0)

{ sigemptyset(&set);

sigaddset(&set, SIGUSR1);

sigwait(&set, &signal);

cout << "[[child]] received SIGUSR1. Process PID : " << getpid() << endl;

cout << "[[child]] reading from file and writing to it..." << endl;

child\_wread();

cout << "[[child]] done working, sending SIGUSR2 to parent." << endl;

kill(getppid(), SIGUSR2);

cout << "[[child]] message sent, exiting." << endl;

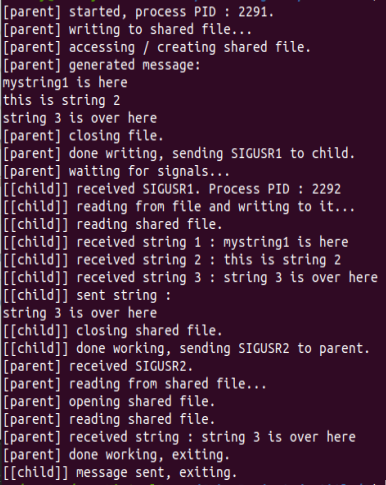
exit(0);

}

return 0;

}

**Результат работы программы:**



Вывод: В ходе данной лабораторной работы изучил основные принципы работы, ознакомился с семафорами.