Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

Студент: Молчанов Владислав Дмитриевич

Группа: М8О-208Б-20

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/molch4nov/OS>

**Постановка задачи**

Необходимо написать 3 программы. Далее будем обозначать эти программы A, B, C. Программа  
A принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка  
строк должна производится построчно. Программа C печатает в стандартный вывод, полученную  
строку от программы A. После получения программа C отправляет программе А сообщение о том,  
что строка получена. До тех пор, пока программа А не примет «сообщение о получение строки» от  
программы С, она не может отправлять следующую строку программе С. Программа B пишет в  
стандартный вывод количество отправленных символов программой А и количество принятых  
символов программой С. Данную информацию программа B получает от программ A и C  
соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

**Общие сведения о программе:** программа состоит из трех файлов: A.cpp, B.cpp, C.cpp

**Общий метод и алгоритм решения:** В начале создаются два дочерних процесса для B и C, сначала A с помощью getline считывает строку, передаёт в B количество считанных символов, а в C — количество считанных символов и саму строку посимвольно, затем B выводит количество введённых символов, C выводит строку и передаёт B количество выведенных символов, после чего B выводит количество выведенных символов и цикл начинается заново. Межпроцессорное взаимодействие основано на семафорах и pipe.

**Исходный код:**

A.cpp

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

using namespace std;

int quick\_get(sem\_t \*semaphore){

int s;

sem\_getvalue(semaphore, &s);

return s;

}

void quick\_set(sem\_t \*semaphore, int n){ //переделать{

while (quick\_get(semaphore) < n){

sem\_post(semaphore);

}

while (quick\_get(semaphore) > n){

sem\_wait(semaphore);

}

}

int main(){

int fdAC[2];

int fdAB[2];

int fdBC[2];

pipe(fdAC);

pipe(fdAB);

pipe(fdBC);

sem\_unlink("\_semA");

sem\_unlink("\_semB");

sem\_unlink("\_semC");

sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1); //0777 разобраться

sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);

if ((semA == SEM\_FAILED)||(semB == SEM\_FAILED)||(semC == SEM\_FAILED)){

perror("sem\_open");

return -1;

}

cout << "Enter some strings:\n";

pid\_t C = fork();

if (C == -1){

perror("fork");

return -1;

}

if (C == 0){

pid\_t B = fork();

if (B == -1){

perror("fork");

return -1;

}

if (B == 0){

execl("B", to\_string(fdAB[0]).c\_str(), to\_string(fdAB[1]).c\_str(), to\_string(fdBC[0]).c\_str(), to\_string(fdBC[1]).c\_str(), NULL); //разобраться

}

else{

execl("C", to\_string(fdAC[0]).c\_str(), to\_string(fdAC[1]).c\_str(), to\_string(fdBC[0]).c\_str(), to\_string(fdBC[1]).c\_str(), NULL);

}

}

else{

while(1){

string str;

getline(cin, str);

if (str == "EXIT"){

quick\_set(semA, 2);

quick\_set(semB, 2);

quick\_set(semC, 2);

break;

}

int size = str.length();

write(fdAC[1], &size, sizeof(int));

write(fdAB[1], &size, sizeof(int));

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

write(fdAC[1], &str[i], sizeof(char));

}

quick\_set(semB, 1);

quick\_set(semA, 0);

while (quick\_get(semA) == 0){

continue;

}

}

}

sem\_close(semA);

sem\_destroy(semA);

sem\_close(semB);

sem\_destroy(semB);

sem\_close(semC);

sem\_destroy(semC);

close(fdAC[0]);

close(fdAC[1]);

close(fdAB[0]);

close(fdAB[1]);

return 0;

}

B.cpp

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

using namespace std;

int quick\_get(sem\_t \*semaphore){

int s;

sem\_getvalue(semaphore, &s);

return s;

}

void quick\_set(sem\_t \*semaphore, int n){

while (quick\_get(semaphore) < n)

{

sem\_post(semaphore);

}

while (quick\_get(semaphore) > n)

{

sem\_wait(semaphore);

}

}

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

int fdAB[2];

fdAB[0] = atoi(argv[0]);

fdAB[1] = atoi(argv[1]);

int fdBC[2];

fdBC[0] = atoi(argv[2]);

fdBC[1] = atoi(argv[3]);

sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);

while (1){

while(quick\_get(semB) == 0){

continue;

}

if (quick\_get(semB) == 2){

break;

}

int size;

read(fdAB[0], &size, sizeof(int));

cout << "Number of input symbols is " << size << endl;

quick\_set(semC, 1);

quick\_set(semB, 0);

while (quick\_get(semB) == 0){

continue;

}

if (quick\_get(semB) == 2){

break;

}

read(fdBC[0], &size, sizeof(int));

cout << "Number of output symbols is " << size << endl;

quick\_set(semA, 1);

quick\_set(semB, 0);

while(quick\_get(semB) == 0){

continue;

}

if (quick\_get(semB) == 2){

break;

}

}

sem\_close(semA);

sem\_close(semB);

sem\_close(semC);

close(fdAB[0]);

close(fdAB[1]);

close(fdBC[0]);

close(fdBC[1]);

return 0;

}

C.cpp

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

using namespace std;

int quick\_get(sem\_t \*semaphore){

int s;

sem\_getvalue(semaphore, &s);

return s;

}

void quick\_set(sem\_t \*semaphore, int n){

while (quick\_get(semaphore) < n){

sem\_post(semaphore);

}

while (quick\_get(semaphore) > n){

sem\_wait(semaphore);

}

}

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

int fdAC[2];

fdAC[0] = atoi(argv[0]);

fdAC[1] = atoi(argv[1]);

int fdBC[2];

fdBC[0] = atoi(argv[2]);

fdBC[1] = atoi(argv[3]);

sem\_t\* semA = sem\_open("\_semA", O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open("\_semB", O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open("\_semC", O\_CREAT, 0777, 0);

while(1){

while(quick\_get(semC) == 0){

continue;

}

if (quick\_get(semC) == 2){

break;

}

int size;

string str;

read(fdAC[0], &size, sizeof(int));

int t = 0;

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

char c;

read(fdAC[0], &c, sizeof(char));

str.push\_back(c);

t = i;

}

++t;

cout << str << endl;

write(fdBC[1], &t, sizeof(int));

quick\_set(semB, 1);

quick\_set(semC, 0);

}

sem\_close(semA);

sem\_close(semB);

sem\_close(semC);

close(fdAC[0]);

close(fdAC[1]);

close(fdBC[0]);

close(fdBC[1]);

return 0;

}

**Демонстрация работы программы**

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/учеба/os\_cp$ make

g++ -pthread A.cpp -o A

g++ -pthread B.cpp -o B

g++ -pthread C.cpp -o C

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/учеба/os\_cp$ ./A

Enter some strings:

haha

Number of input symbols is 4

haha

Number of output symbols is 4

pupupu

Number of input symbols is 6

pupupu

Number of output symbols is 6

EXIT

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/учеба/os\_cp$ ./A

Enter some strings:

brawl stast

Number of input symbols is 11

brawl stast

Number of output symbols is 11

dawd

Number of input symbols is 4

dawd

Number of output symbols is 4

^C

**Выводы**

При написании курсового проекта я укрепил знания и навыки, полученные мной во время прохождения курса операционных систем.