МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт№8: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**КУРСОВОЙ ПРОЕКТ**

По курсу «Операционные системы»

III семестр

Тема:

«Межпроцессорное взаимодействие»

|  |  |
| --- | --- |
| Группа: | М8О-206Б-21 |
| Студент: | Орусский В.Р. |
| Преподаватель: | Миронов Е. С. |
| Оценка: |  |
| Дата: |  |

Москва, 2023

Содержание

[Постановка задачи 3](#_Toc130579388)

[Описание программы 4](#_Toc130579389)

[Листинг 4](#_Toc130579390)

[function.cpp 4](#_Toc130579391)

[main.cpp 5](#_Toc130579392)

[A.cpp 6](#_Toc130579393)

[B.cpp 7](#_Toc130579394)

[C.cpp 8](#_Toc130579395)

[Пример работы 9](#_Toc130579396)

[Вывод 9](#_Toc130579397)

[Источники 10](#_Toc130579398)

Постановка задачи

Необходимо написать 3-и 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.

Описание программы

В главной программе запускаются два процесса, которые заменяются программами B и C с помощью execl, а сама главная программа заменяется программой A.

Программа A считывает символы построчно и передает количество символов программам B и C, а саму строку передает только программе C. Затем программа A блокируется с помощью семафора A и ожидает разблокировки. После этого программа разблокирует семафор B.

Программа B ждет разблокировки семафора B и после разблокировки выводит размер, переданный от программы A. Затем она блокирует семафор B и разблокирует семафор C. После разблокировки семафора в программе C, программа B выводит размер полученный от программы C. Затем программа B снова блокирует семафор B и разблокирует семафор A.

Программа C ждет разблокировки семафора C и после разблокировки выводит строку, полученную от программы A. Затем она подсчитывает количество выведенных символов и отправляет это количество программе B. Затем программа C блокирует семафор C и разблокирует семафор B.

В библиотеке описаны две вспомогательные функции для получения и установки значений, а также настройки программ.

Листинг

function.cpp

const char\* SEMAPHORE\_A = "sem\_a";

const char\* SEMAPHORE\_B = "sem\_b";

const char\* SEMAPHORE\_C = "sem\_c";

const char\* INPUT\_END = "EOF";

int get\_semaphore\_value(sem\_t\* semaphore) {

int s;

sem\_getvalue(semaphore, &s);

return s;

}

void set\_semaphore\_value(sem\_t\* semaphore, int n) {

while (get\_semaphore\_value(semaphore) < n) {

sem\_post(semaphore);

}

while (get\_semaphore\_value(semaphore) > n) {

sem\_wait(semaphore);

}

}

main.cpp

#include <iostream>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

#include "functions.cpp"

using namespace std;

int main() {

int fdAC[2], fdAB[2], fdBC[2];

pipe(fdAC);

pipe(fdAB);

pipe(fdBC);

sem\_unlink(SEMAPHORE\_A);

sem\_unlink(SEMAPHORE\_B);

sem\_unlink(SEMAPHORE\_C);

sem\_t\* semA = sem\_open(SEMAPHORE\_A, O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open(SEMAPHORE\_B, O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open(SEMAPHORE\_C, O\_CREAT, 0777, 0);

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

perror("sem\_open");

return -1;

}

cout << "End of input " << INPUT\_END << endl;

pid\_t C = fork();

if (C == -1) {

perror("fork");

return -1;

}

if (C == 0) {

close(fdAB[0]);

close(fdAB[1]);

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 {

pid\_t B = fork();

if (B == -1){

perror("fork");

return -1;

}

if (B == 0){

close(fdAC[0]);

close(fdAC[1]);

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);

} else {

close(fdBC[0]);

close(fdBC[1]);

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

}

}

return 0;

}

A.cpp

#include <iostream>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

#include "functions.cpp"

using namespace std;

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

int fdAC[2], fdAB[2];

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

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

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

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

sem\_t\* semA = sem\_open(SEMAPHORE\_A, O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open(SEMAPHORE\_B, O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open(SEMAPHORE\_C, O\_CREAT, 0777, 0);

while (true) {

string input;

getline(cin, input);

if (input == INPUT\_END) {

set\_semaphore\_value(semA, 2);

set\_semaphore\_value(semB, 2);

set\_semaphore\_value(semC, 2);

break;

}

int size = input.length();

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

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

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

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

}

set\_semaphore\_value(semB, 1);

set\_semaphore\_value(semA, 0);

while (get\_semaphore\_value(semA) == 0) {

continue;

}

}

sem\_close(semA);

sem\_unlink(SEMAPHORE\_A);

sem\_close(semB);

sem\_unlink(SEMAPHORE\_B);

sem\_close(semC);

sem\_unlink(SEMAPHORE\_C);

close(fdAC[0]);

close(fdAC[1]);

close(fdAB[0]);

close(fdAB[1]);

return 0;

}

B.cpp

#include <iostream>

#include <iostream>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

#include "functions.cpp"

using namespace std;

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

int fdAB[2], fdBC[2];

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

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

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

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

sem\_t\* semA = sem\_open(SEMAPHORE\_A, O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open(SEMAPHORE\_B, O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open(SEMAPHORE\_C, O\_CREAT, 0777, 0);

while (true) {

while (get\_semaphore\_value(semB) == 0) {

continue;

}

if (get\_semaphore\_value(semB) == 2) {

break;

}

int size;

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

cout << "In count: " << size << endl;

set\_semaphore\_value(semC, 1);

set\_semaphore\_value(semB, 0);

while (get\_semaphore\_value(semB) == 0) {

continue;

}

if (get\_semaphore\_value(semB) == 2) {

break;

}

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

cout << "Out count: " << size << endl;

set\_semaphore\_value(semA, 1);

set\_semaphore\_value(semB, 0);

while (get\_semaphore\_value(semB) == 0) {

continue;

}

if (get\_semaphore\_value(semB) == 2) {

break;

}

}

sem\_close(semA);

sem\_unlink(SEMAPHORE\_A);

sem\_close(semB);

sem\_unlink(SEMAPHORE\_B);

sem\_close(semC);

sem\_unlink(SEMAPHORE\_C);

close(fdAB[0]);

close(fdAB[1]);

close(fdBC[0]);

close(fdBC[1]);

return 0;

}

C.cpp

#include <iostream>

#include <string>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

#include "functions.cpp"

using namespace std;

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

int fdAC[2], fdBC[2];

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

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

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

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

sem\_t\* semA = sem\_open(SEMAPHORE\_A, O\_CREAT, 0777, 1);

sem\_t\* semB = sem\_open(SEMAPHORE\_B, O\_CREAT, 0777, 0);

sem\_t\* semC = sem\_open(SEMAPHORE\_C, O\_CREAT, 0777, 0);

while (true) {

while (get\_semaphore\_value(semC) == 0) {

continue;

}

if (get\_semaphore\_value(semC) == 2) {

break;

}

int size;

string input;

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

int count = 0;

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

char c;

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

input.push\_back(c);

count+=1;

}

cout << input << endl;

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

set\_semaphore\_value(semB, 1);

set\_semaphore\_value(semC, 0);

}

sem\_close(semA);

sem\_unlink(SEMAPHORE\_A);

sem\_close(semB);

sem\_unlink(SEMAPHORE\_B);

sem\_close(semC);

sem\_unlink(SEMAPHORE\_C);

close(fdAC[0]);

close(fdAC[1]);

close(fdBC[0]);

close(fdBC[1]);

return 0;

}

Пример работы

End of input EOF string In count: 6 string Out count: 6 123 In count: 3 123 Out count: 3

In count: 0

Out count: 0 EOF

Вывод

С помощью данной курсовой я применил знания полученные во время выполнения предыдущих лаб. Самым простым способом для организации взаимодействия между программами оказалось использование pipe для передачи строк между программами и семафоров для передачи состояний. Во время написания кода были не удобны стандартные функции семафоров, поэтому пришлось написать свою.

Источники

1. Иртегов Д. В. Введение в операционные системы. — 2-е изд. — СПб.: BHV-СПб, 2007.
2. Таненбаум э., Вудхалл А. Операционные системы: разработка и реализация (+CD). Классика СS. - . СПб.: Питер, 2006.-. 576 с
3. Столлингс У. Операционные системы = Operating Systems: Internals and Design Principles. — М.: Вильямс, 2004. — 848 с.