Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по курсу
«Операционные системы»

Студент: Шандрюк Пётр Николаевич
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/Peter1811/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 C. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе A сообщение о том, что строка получена. До тех пор, пока программа A не примет «сообщение о получение строки» от программы C, она не может отправлять следующую строку программе C. Программа B пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой C. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

Общие сведения о программе: программа состоит из четырёх файлов: A.cpp, B.cpp, C.cpp и main.cpp, который объединяет в себе три предыдущих файла.

Общий метод и алгоритм решения: В начале работы в main.cpp создаются два дочерних процесса для B и C, а родительский процесс замещается программой A с помощью execl, 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:

main.cpp

```
#include <iostream>
#include <unistd.h>
#include <semaphore.h>
#include <string.h>
#include <sstream>
#include <vector>

using namespace std;

int main() {

    int fdAC[2];
    int fdCA[2];
    int fdCB[2];
    int fdAB[2];
    if (pipe(fdAC) < 0 || pipe(fdCA) < 0 || pipe(fdCB) < 0 || pipe(fdAB) < 0) {
```

```
        cout << "Error with pipe" << endl;
        exit(-1);
    }

    int pid = fork();

    if (pid < 0) {
        cout << "Error with fork" << endl;
        exit(-1);
    } else if (pid == 0) { // Child process (C)
        execl("C", to_string(fdAC[0]).c_str(),
                    to_string(fdCA[1]).c_str(),
                    to_string(fdCB[1]).c_str(),
                    NULL);


    } else {

        int pid2 = fork();

        if (pid2 < 0) {
            cout << "Error eith fork" << endl;
            exit(-1);
        }
        else if (pid2 == 0) { // Child process (B)
            execl("B", to_string(fdCB[0]).c_str(),
                        to_string(fdAB[0]).c_str(), NULL);

        } else { // Parent process (A)
            execl("A", to_string(fdCA[0]).c_str(),
                        to_string(fdAC[1]).c_str(),
                        to_string(fdAB[1]).c_str(), NULL);
        }
    }


    return 0;
}
```

A.cpp

```
#include <iostream>
#include <unistd.h>
```

```cpp
using namespace std;

int main(int argc, char* argv []) {
    int fdAC[2];
    int fdCA[2];
    int fdAB[2];
    fdAC[1] = atoi(argv[1]);
    fdCA[0] = atoi(argv[0]);
    fdAB[1] = atoi(argv[2]);
    char c;
    int res;
    int len = 0;
    while ((c = getchar()) != EOF) {
        len ++;
        if (write(fdAC[1], &c, sizeof(char)) == -1) {
            cout << "Error with writing in parent" << endl;
            return -1;
        }
        if (c == '\n') {
            if (read(fdCA[0], &res, sizeof(int)) == -1) {
                cout << "Error with reading in " << endl;
                return -1;
            }
            len --;
            if (write(fdAB[1], &len, sizeof(int)) == -1) {
                cout << "Error with writing in parent" << endl;
                return -1;
            }
            len = 0;
        }
    }
    if (c == EOF) {
        int ex = -1;
        if (write(fdAB[1], &ex, sizeof(int)) == -1) {
            cout << "Error with writing in parent" << endl;
            return -1;
        }
    }


    return 0;
}
```

B.cpp

```cpp
#include <iostream>
#include <unistd.h>

using namespace std;

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

    int fdCB[2];
    int fdAB[2];
    fdAB[0] = atoi(argv[1]);
    fdCB[0] = atoi(argv[0]);

    int incoming;
    int outcoming;
    while (true) {
        if (read(fdAB[0], &outcoming, sizeof(int)) == -1) {
            cout << "Error with reading in child" << endl;
            return -1;
        }
        if (outcoming == -1) break;
        if (read(fdCB[0], &incoming, sizeof(int)) == -1) {
            cout << "Error with reading in child" << endl;
            return -1;
        }
        cout << "Outcomming symbols: " << outcoming << endl;
        cout << "Incomming symbols: " << incoming << endl;
        cout << endl;
    }


    return 0;
}
```

C.cpp

```cpp
#include <iostream>
#include <vector>
#include <unistd.h>

using namespace std;

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

```
    int fdAC[2];
    int fdCA[2];
    int fdCB[2];
    fdAC[0] = atoi(argv[0]);
    fdCA[1] = atoi(argv[1]);
    fdCB[1] = atoi(argv[2]);
    int ans = 0;
    char c;
    int len2 = 0;
    vector <char> my_vec;
    while(read(fdAC[0], &c, sizeof(char))) {
        len2 ++;
        my_vec.push_back(c);
        if (c == '\n') {
            if (write(fdCA[1], &ans, sizeof(int)) == -1) {
                cout << "Error with writing in " << endl;
                return -1;
            }
            for (int i = 0; i < my_vec.size(); i++) {
                cout << my_vec[i];
            }
            my_vec.clear();
            len2 --;
            // cout << endl;
            if (write(fdCB[1], &len2, sizeof(int)) == -1) {
                cout << "Error with writing in child" << endl;
                return -1;
            }
            len2 = 0;
        }
        }


    return 0;
}
```

Makefile

```
files: main A B C

main: main.cpp
    g++ main.cpp -o main

A: A.cpp
```

```
    g++ A.cpp -o A

B: B.cpp
    g++ B.cpp -o B

C: C.cpp
    g++  C.cpp -o C
```

Демонстрация работы программы

```
peter@DESKTOP-V53N291:/mnt/c/Users/Peter/Desktop/Repositories/OS/KP/src$ ./main
hello world
hello world
Outcomming symbols: 11
Incomming symbols: 11

hello, my name is Peter
hello, my name is Peter
Outcomming symbols: 23
Incomming symbols: 23



Outcomming symbols: 0
Incomming symbols: 0

peter@DESKTOP-V53N291:/mnt/c/Users/Peter/Desktop/Repositories/OS/KP/src$
```

Выводы

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