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

Курсовая работа по курсу
«Операционные системы»

Студент: Рокотянский А.Е.
Группа: М8О-201Б-21
Вариант: 27
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/Mikhail-cWc/OS_mai/tree/main/kp

Постановка задачи

Цель работы

1. Приобретение практических навыков в использовании знаний, полученных в течении курса
2. Проведение исследования в выбранной предметной области

Задание

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 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

Межпроцессорное взаимодействие осуществляется с помощью пайпов и семафоров

Исходный код

Main.cpp

```
#include "utils.h"

int main()
{
    int fdAC[2];
    int fdAB[2];
    int fdBC[2];
    pipe(fdAC);
    pipe(fdAB);
    pipe(fdBC);
        // Удалить именованный семафор
    sem_unlink("_semA");
    sem_unlink("_semB");
    sem_unlink("_semC");
    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);
    if ((semA == SEM_FAILED)||(semB == SEM_FAILED)||(semC == SEM_FAILED))
    {
        perror("sem_open");
        return -1;
    }
    std::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", std::to_string(fdAB[0]).c_str(), std::to_string(fdAB[1]).c_str(),
std::to_string(fdBC[0]).c_str(), std::to_string(fdBC[1]).c_str(), NULL);
        }
        else
        {
            execl("C", std::to_string(fdAC[0]).c_str(), std::to_string(fdAC[1]).c_str(),
std::to_string(fdBC[0]).c_str(), std::to_string(fdBC[1]).c_str(), NULL);
        }
    }
    else
    {
        execl("A", std::to_string(fdAC[0]).c_str(), std::to_string(fdAC[1]).c_str(),
std::to_string(fdAB[0]).c_str(), std::to_string(fdAB[1]).c_str(), NULL);
    }
    return 0;
}
```

A.cpp

```
#include "utils.h"

int main(int args, char* argv[])
{
    int fdAC[2];
    fdAC[0] = atoi(argv[0]);
    fdAC[1] = atoi(argv[1]);
    int fdAB[2];
    fdAB[0] = atoi(argv[2]);
    fdAB[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)
    {
        std::string str;
        getline(std::cin, str);

        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));
        }
        sem_post(semB);
        sem_wait(semA);
    }
    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 "utils.h"

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)
    {
        int size;
        read(fdAB[0], &size, sizeof(int));
        std::cout << "Number of input symbols is " << size << std::endl;

        sem_post(semC);
        sem_wait(semB);

        read(fdBC[0], &size, sizeof(int));
        std::cout << "Number of output symbols is " << size << std::endl;

        sem_post(semA);
        sem_wait(semB);

    }
    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 "utils.h"

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)
    {
```

```
        int size;
        std::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;
        std::cout << str << std::endl;
        write(fdBC[1], &t, sizeof(int));

        sem_post(semB);
        sem_wait(semC);
    }
    sem_close(semA);
    sem_close(semB);
    sem_close(semC);
    close(fdAC[0]);
    close(fdAC[1]);
    close(fdBC[0]);
    close(fdBC[1]);
    return 0;
}
```

Выводы

Составлена и отлажена программа на языке C++, в соответствии с заданием.