

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

Общие сведения о программе: программа состоит из трех файлов: А.cpp, В.cpp, С.cpp

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

Исходный код:

А.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.