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

Курсовой проект по курсу

«Операционные системы»

Студент: Румынина Екатерина Александровна
Группа: М8О-201Б-21
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/KetRum0/mai_os_labs

Постановка задачи

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.
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

a.c – программа A

b.c – программа B

c.c – программа C

Общий метод и алгоритм решения

Создам четыре канала для взаимодействия процессов между собой.

Первый канал ab – для отправки длины строки из A в C.

Второй канал ac – для отправки строки из A в C.

Третий канал ca – для отправки сигнала из C в A, что строка была получена.

Четвёртый канал cb – для отправки длины полученной C из A в B.

Исходный код

a.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
```

```
#include <ctype.h>


char* ReadString(FILE* stream) {
    if(feof(stream)) {
        return NULL;
    }

    const int chunkSize = 256;
    char* buffer = (char*)malloc(chunkSize);
    int bufferSize = chunkSize;

    if(!buffer) {
        printf("Couldn't allocate buffer");
        exit(EXIT_FAILURE);
    }

    int readChar;
    int idx = 0;
    while((readChar = getc(stream)) != EOF) {
        buffer[idx++] = readChar;

        if(idx == bufferSize) {
            buffer = realloc(buffer, bufferSize + chunkSize);
            bufferSize += chunkSize;
        }

        if(readChar == '\n') {
            break;
        }
    }

    buffer[idx] = '\0';

    return buffer;
}

size_t str_length(char *str) {
```

```
    size_t length = 0;
    for (int i = 0; str[i] != '\n'; i++) {
        length++;
    }
    return length;
}

int main() {
    int ab[2];
    int ac[2];
    int ca[2];
    int cb[2];

    if (pipe(ab) == -1){
                                    perror("pipe error");
                                    exit(EXIT_FAILURE);
                                }

    if (pipe(ac) == -1){
                                    perror("pipe error");
                                    exit(EXIT_FAILURE);
                                }

    if (pipe(ca) == -1){
                                    perror("pipe error");
                                    exit(EXIT_FAILURE);
                                }

    if (pipe(cb) == -1){
                                    perror("pipe error");
                                    exit(EXIT_FAILURE);
                                }

    int id1 = fork();

    if (id1 < 0) {
        perror("fork error");
                                    exit(EXIT_FAILURE);
```

```
    }
    else if (id1 == 0) {
        close(ac[1]);
        close(ca[0]);
        close(cb[0]);
        close(ab[0]);
        close(ab[1]);

        char pac[3];
        sprintf(pac, "%d", ac[0]);

        char pca[3];
        sprintf(pca, "%d", ca[1]);

        char pcb[3];
        sprintf(pcb, "%d", cb[1]);

        if (execl("./c", "./c", pac, pca, pcb, NULL) == -1){
                                            perror("execl error");
                                            exit(EXIT_FAILURE);
                                    }

    }
    else {
        int id2 = fork();
        if (id2 < 0) {
            perror("fork error");
                                        exit(EXIT_FAILURE);
        }
        else if (id2 == 0) {
            close(ac[0]);
            close(ac[1]);
            close(ca[0]);
            close(ca[1]);
            close(cb[1]);
            close(ab[1]);

            char pcb[3];
```

```
            sprintf(pcb, "%d", cb[0]);

            char pab[3];
            sprintf(pab, "%d", ab[0]);

            if ( execl("./b", "./b",  pcb, pab, NULL) == -1){
                                        perror("execl error");
                                        exit(EXIT_FAILURE);
                                    }

        }
        else {
            close(ac[0]);
            close(ca[1]);
            close(ab[0]);
            close(cb[1]);
            close(cb[0]);

            char *str = NULL;
            while ((str = ReadString(stdin)) != NULL) {
                size_t size = str_length(str);
                write(ac[1], &size, sizeof(size_t));
                write(ac[1], str, size);
                write(ab[1], &size, sizeof(size_t));
                int ok;
                read(ca[0], &ok, sizeof(ok));
            }
            close(ca[0]);
            close(ac[1]);
            close(ab[1]);
         }
    }

    return 0;
}
```

b.c

```
#include <stdlib.h>
```

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>
#include <stdbool.h>

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

    int pcb = atoi(argv[1]);
    int pab = atoi(argv[2]);

    size_t size,size1;

    while (read(pab, &size, sizeof(size_t)) > 0) {
        read(pcb, &size1, sizeof(size_t));
        printf("b - from a: %zu\n", size);
        printf("b - from c: %zu\n", size1);
    }

    close(pcb);
    close(pab);

    return 0;
}
```

c.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>

int main(int argc, char *argv[]) {
    int pac = atoi(argv[1]);
    int pca = atoi(argv[2]);
    int pcb = atoi(argv[3]);
```

```
    size_t size,size1;
    while (read(pac, &size, sizeof(size_t)) > 0) {
        char *str = (char*) malloc(size);
        if (str == NULL) {
            perror("malloc error");
                                        exit(EXIT_FAILURE);
        }
        read(pac, str, size);
        printf("c - from a: %s\n", str);
        size1 = strlen(str);
        write(pcb, &size1, sizeof(size_t));
        int ok = 1;
        write(pca, &ok, sizeof(int));
        free(str);
    }

    close(pac);
    close(pca);
    close(pcb);

    return 0;
}
```

Демонстрация работы программы

```
ket@ket-laptop:~/Desktop/mai_os_labs/cp$ ./a
abcde
c - from a: abcde
b - from a: 5
b - from c: 5
123456789/// // //
c - from a: 123456789/// // //
b - from a: 18
b - from c: 18
!2$  ^ 7& (8* /.. !
c - from a: !2$  ^ 7& (8* /.. !
b - from a: 19
b - from c: 19

c - from a:
b - from a: 0
```

```
b - from c: 0
^C
```

Выводы

В результате выполнения данной лабораторной работы я приобрела практические навыки в использовании знаний, полученных в течении курса