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

Курсовая работа по курсу

«Операционные системы»

Вариант на удовлетворительно

Студент: Андреев Александр Олегович
Группа: М8О-206Б-20
Вариант:
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Постановка задачи

Требуется создать три программы A, B, C. Программа A принимает из стандартного ввода строки, а далее их отправляет программе B. Отправка строк должна производиться построчно. Программа B печатает в стандартный вывод, полученную строку от программы A. После получения программа B отправляет программе A сообщение о том что строка получена. До тех пор пока программа A не получит сообщение о получении строки от программы B, она не может отправлять следующую строку программе B. Программа C пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой B. Данную информацию программа C получает от программ A и B соответственно.

Алгоритм решения

Создаем 4 pipe`a, через которые программы смогут взаимодействовать друг с другом. Один для отправке строк из A в C, другой для отправки длины строки из A в B, следующий для отправки результата C в A и последний для отправки длины строки из C в B.

Программа содержит функцию получения строки со стандартного ввода.

Программа создает два дочерних процесса. В первом мы выполняем команду B, во втором C

Исходный код

A_prog.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <ctype.h>


#include "myString.h"
```

```
#define MIN_CAP 10


// int reading_string(string* str){
//         s_create(str);
//         char cur;
//         while(read(STDIN_FILENO, &cur, sizeof(char)) > 0){
//                 if(cur == '\n'){
//                         break;
//                 }
//                 s_push(str, cur);
//         }
//         return s_size(str);
// }
int read_string(char **str_, int fd) {
    free(*str_);
    int str_size = 0;
    int cap = MIN_CAP;
    char *str = (char*) malloc(sizeof(char) * cap);
    if (str == NULL) {
        perror("Malloc error");
        exit(-1);
    }
    char c;
    while (read(fd, &c, sizeof(char)) == 1) {
```

```
        if (c == '\n') {
            break;
        }
        str[(str_size)++] = c;
        if (str_size == cap) {
            cap = cap * 3 / 2;
            str = (char*) realloc(str, sizeof(char) * cap);
            if (str == NULL) {
                perror("Realloc error");
                exit(-2);
            }
        }
    }
    str[str_size] = '\0';

    *str_ = str;
    return str_size;
}
int str_length(char *str) {
    int length = 0;
    for (int i = 0; str[i] != '\0'; ++i) {
        ++length;
    }
    return length;
```

```
}

int main(){
    int a2b[2];
    int a2c[2];
    int c2a[2];
    int c2b[2];
    if (pipe(a2b) == -1) {
        perror("Cannot create pipe\n");
        return -1;
    }if (pipe(a2c) == -1) {
        perror("Cannot create pipe\n");
        return -2;
    }if (pipe(c2a) == -1) {
        perror("Cannot create pipe\n");
        return -3;
    }if (pipe(c2b) == -1) {
        perror("Cannot create pipe\n");
        return -4;
    }
    int pid1 = fork();
    if (pid1 < 0){
        perror("Problem with fork\n");
        return -5;
```

```
}
else if(pid1 == 0){
    close(a2c[1]);
    close(c2a[0]);
    close(c2b[0]);
    close(a2b[0]);
    close(a2b[1]);
    char pa2c[3];
    char pc2a[3];
    char pc2b[3];
    sprintf(pa2c, "%d", a2c[0]);
    sprintf(pc2a, "%d", c2a[1]);
    sprintf(pc2b, "%d", c2b[1]);
    execl("C_prog", "C_prog", pa2c, pc2a, pc2b, NULL);
}
else {
    int pid2 = fork();
    if (pid2 < 0){
        perror("Problem with fork\n");
        return -6;
    }
    else if (pid2 == 0){
        close(a2c[0]);
        close(a2c[1]);
```

```
    close(c2a[0]);
    close(c2a[1]);
    close(c2b[1]);
    close(a2b[1]);
    char pc2b[2];
    char pa2b[2];
    sprintf(pc2b, "%d", c2a[0]);
    sprintf(pa2b, "%d", c2b[0]);
    execl("B_prog", "B_prog",  pc2b, pa2b, NULL);
}
else {
    close(a2c[0]);
    close(c2a[1]);
    close(a2b[0]);
    close(c2b[1]);
    close(c2b[0]);
    // string str;
    // while(reading_string(&str)){
    //     int size = s_size(&str);
    //     write(a2c[1], &size, sizeof(int));
    //     write(a2c[1], &str, size);
    //     write(a2b[1], &size, sizeof(int));
    //     int ok;
    //     read(c2a[0], &ok, sizeof(ok));
```

```
            // }
            char *str = NULL;
            while ((read_string(&str, 0))) {
                int size = str_length(str);
                write(a2c[1], &size, sizeof(int));
                write(a2c[1], str, size);
                write(a2b[1], &size, sizeof(int));
                int ok;
                read(c2a[0], &ok, sizeof(ok));
            }
            close(c2a[0]);
            close(a2c[1]);
            close(a2b[1]);
        }
    }
    return 0;
}
```

B_prog.c

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

```
int main(int argc, char *argv[]) {
    int pc2b = atoi(argv[1]);
    int pa2b = atoi(argv[2]);

    int size;

    while (read(pa2b, &size, sizeof(int))) {
        printf("[B] - Get A: %d\n", size);
        read(pc2b, &size, sizeof(int));
        printf("[B] - Get C: %d\n", size);
    }

    close(pc2b);
    close(pa2b);

    return 0;
}
```

C_prog.c

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

```
int main(int argc, char *argv[]) {
    int pa2c = atoi(argv[1]);
    int pc2a = atoi(argv[2]);
    int pc2b = atoi(argv[3]);

    int size;

    while (read(pa2c, &size, sizeof(int))) {
        char* str = (char*)malloc(size);
        if (str == NULL){
            printf("NO MEMORY\n");
        }
        read(pa2c, str, size);
        printf("(C)Delivered from A: %s\n", str);
        write(pc2b, &size, sizeof(int));
        int ok = 1;
        write(pc2a, &ok, sizeof(int));
        free(str);
    }

    close(pa2c);
    close(pc2a);
    close(pc2b);
```

```
    return 0;
}
```

Демонстрация работы программы

missclick3@missclick3:~/Desktop/OSLabs/Capstone project$ ./a.out
test
(C)Delivered from A: test
(B) Recieved from A: 4
(B) Recieved from A: 4
qwerty
(C)Delivered from A: qwerty
(B) Recieved from A: 6
(B) Recieved from A: 6
hello
(C)Delivered from A: hello
(B) Recieved from A: 5
(B) Recieved from A: 5

Выводы

В ходе выполнения данной лабораторной работы я опирался на знания, полученные во время выполнения 2 лабораторной работы. Я использовал создание дочерних процессов, а также вспомнил, как работать в pipe`ами.