Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Опреационные системы»
Задание на «удовлетворительно»

Студент: Е. А. Айрапетова
Преподаватель: Е. С. Миронов
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2021

Постановка задачи

Цель работы:

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

Задание:

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

Вариант:

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 производиться 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.

Взаимодействие между программами будет происходить с помощью однонаправленных каналов.

Общие сведения о программе

Программы компилируются из файлов kp_a.c, kp_b.c и kp_c.c. Используются заголовочные файлы <stdlib.h>, <stdio.h>, <unistd.h>, <fcntl.h>, <ctype.h>. В программе используются следующие системные вызовы:

- pipe - создает канал и возвращает два файловых дескриптора, для взаимодействия по каналу;
- close – закрывает файловый дескриптор;
- fork – создает дочерний процесс и продолжает выполнение текущей программы в нем;
- execl – загружает в текущий процесс другую програму;

Метод и алгоритм решения

Программа A создаёт 4 канала: ab направлен из процесса A в процесс B, ac - из процесса A в процесс C, ca - из процесса C в процесс A, cb - из C в B.

Далее програма A считывает строку и отправлят её размер и саму строку программе C. Программе B отправляется только размер строки.

Программа C считывает данные из канала. Сначала считывается размер, выделяется память, считывается соответствующее сообщение. Отправляется информация о длине считанной строки программе B. Далее выводится строка.

Программа B считывает сначала информацию по каналу из A, затем из C. Затем выводит эту информацию.

Основные файлы программы

kp_a.c:

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

#define MIN_CAP 4
#define STDIN 0

size_t read_string(char **str_, int fd) {
    free(*str_);
    size_t str_size = 0;
    size_t 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) {
            str = (char*) realloc(str, sizeof(char) * cap * 3 / 2);
            cap = cap * 3 / 2;
            if (str == NULL) {
                perror("Realloc error");
                exit(-2);
            }
        }
    }
    str[str_size] = '\0';

    *str_ = str;
    return str_size;
}

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

```
}

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

    pipe(ab);
    pipe(ac);
    pipe(ca);
    pipe(cb);

    int id1 = fork();
    if (id1 < 0) {
        perror("Fork error");
        exit(1);
    }
    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]);

        execl("./c", "./c", pac, pca, pcb, NULL);
    }
    else {
        int id2 = fork();
        if (id2 < 0) {
            perror("Fork error");
            exit(1);
        }
        else if (id2 == 0) {
            close(ac[0]);
            close(ac[1]);
            close(ca[0]);
            close(ca[1]);
            close(cb[1]);
            close(ab[1]);

            char pcb[2];
```

```
            sprintf(pcb, "%d", ca[0]);
            char pab[2];
            sprintf(pab, "%d", cb[0]);

            execl("./b", "./b", pcb, pab, NULL);
        }
        else {
            close(ac[0]);
            close(ca[1]);
            close(ab[0]);
            close(cb[1]);
            close(cb[0]);

            char *str = NULL;
            while ((read_string(&str, STDIN)) > 0) {
                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;
}
```

kp_b.c:

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

    while (read(pab, &size, sizeof(size_t)) > 0) { //
        //     ,   while
        printf("B - From a: %zu\n", size);
```

```
        read(pcb, &size, sizeof(size_t));
        printf("B - From c: %zu\n", size);
    }

    close(pcb);
    close(pab);

    return 0;
}
```

kp_c.c:

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.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;
    while (read(pac, &size, sizeof(size_t)) > 0) {
        char *str = (char*) malloc(size); //
        if (str == NULL) {
            printf("MALLOC from C\n");
            exit(-1);
        }
        read(pac, str, size);
        printf("C - From a: %s\n", str);
        write(pcb, &size, sizeof(size_t));
        int ok = 1;
        write(pca, &ok, sizeof(int));
        free(str);
    }
    close(pac);
    close(pca);
    close(pcb);
    return 0;
}
```

Тестирование программы

```
jane@Evgenia:/mnt/c/Files/ОСи/КП$ ./a
hello world
C -From a: hello world
B -From a: 11
B -From c: 11
looooooooooooooooooooooooooong string
C -From a: looooooooooooooooooooooooooong string
B -From a: 39
B -From c: 39
a
C -From a: a
B -From a: 1
B -From c: 1
jane@Evgenia:/mnt/c/Files/ОСи/КП$
```

Выводы:

Выполнив курсовую работу по курсу «Операционные системы», я реализовала общение между 3 независимыми программами, применила передачу дескриптора не всего конвейера, а лишь ввода или вывода. Также я освежила свои знания о командах exec и fork.