Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»

Студент: А. Р. Боташев
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21

Дата:
Оценка:
Подпись:

Москва, 2023

1 Постановка задачи

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 стандартый 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 отправялять следующую строку программе C. Программа B пишет в стандартный вывод количество отправленных символов программой A и количество принятыхсимволов программой C. Данную информацию программа B получает от программ A и C соответственно

2 Сведения о программе

Программы написаны на языке C++ для Unix подобной операционной системы на базе ядра Linux. Для связи между процессами используется pipe

3 Общий метод и алгоритм решения

Программа A создает два дочерних процесса B и C, затем считывает строки из стандартного потока ввода, после чего передает строки процессу C

Процесс C пишет полученную строку, подтвердив получение строки процессу A

A отправляет размер отправленной строки программе B

C отправляет размер полученной строки программе B

4 Листинг программы

A.cpp

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <signal.h>
5 #include <string.h>
6
7 #include "../include/get_line.h"
8
```

```
9  int id1, id2;
10 int pipeAC[2];
11 int pipeAB[2];
12 int pipeCA[2];
13 int pipeCB[2];
14
15 void sig_handler(int signal) {
16     kill(id1, SIGUSR1);
17     kill(id2, SIGUSR1);
18
19     close(pipeAC[0]);
20     close(pipeAC[1]);
21     close(pipeCA[0]);
22     close(pipeCA[1]);
23     close(pipeAB[0]);
24     close(pipeAB[1]);
25     close(pipeCB[0]);
26     close(pipeCB[1]);
27     exit(0);
28 }
29
30 int main(){
31
32     if (signal(SIGINT, sig_handler) == SIG_ERR) {
33         printf("[%d] ", getpid());
34         perror("Error signal ");
35         return -1;
36     }
37     pipe(pipeAC);
38     pipe(pipeAB);
39     pipe(pipeCA);
40     pipe(pipeCB);
41
42
43     id1 = fork();
44     if (id1 == -1)
45     {
46         perror("fork1");
47         exit(-1);
48     }
49     else if(id1 == 0)
50     {
51         char name[] = "./B";
52         char pAB[3] = "";
53         sprintf(pAB, "%d", pipeAB[0]);
54         char pCB[3] = "";
55         sprintf(pCB, "%d", pipeCB[0]);
56
57         close(pipeAC[0]);
```

```
58          close(pipeAC[1]);
59          close(pipeCA[0]);
60          close(pipeCA[1]);
61          close(pipeAB[1]);
62          close(pipeCB[1]);
63
64          execl(name, name, pAB, pCB, NULL);
65      }
66      else
67      {
68          id2 = fork();
69          if (id2 == -1)
70          {
71              perror("fork2");
72              exit(-1);
73          }
74          else if(id2 == 0)
75          {
76              char name[] = "./C";
77              char pAC[3] = "";
78              sprintf(pAC, "%d", pipeAC[0]);
79              char pCA[3] = "";
80              sprintf(pCA, "%d", pipeCA[1]);
81              char pCB[3] = "";
82              sprintf(pCB, "%d", pipeCB[1]);
83
84              close(pipeAC[1]);
85              close(pipeCA[0]);
86              close(pipeAB[0]);
87              close(pipeAB[1]);
88              close(pipeCB[0]);
89              execl(name, name, pAC, pCA, pCB, NULL);
90          }
91          else
92          {
93              char* line = NULL;
94              int size;
95              while((size = get_line(&line, STDIN_FILENO)) != 0)
96              {
97                  write(pipeAC[1], &size, sizeof(int));
98                  write(pipeAC[1], line, size*sizeof(char));
99
100                 int ok;
101                 read(pipeCA[0], &ok, sizeof(int));
102
103                 write(pipeAB[1], &size, sizeof(int));
104             }
105             free(line);
106
```

```
            kill(id1, SIGUSR1);
            kill(id2, SIGUSR1);
        }
    }


    close(pipeAC[0]);
    close(pipeAC[1]);
    close(pipeCA[0]);
    close(pipeCA[1]);
    close(pipeAB[0]);
    close(pipeAB[1]);
    close(pipeCB[0]);
    close(pipeCB[1]);


}
```

B.cpp

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

int pipeAB;
int pipeCB;


void sig_handler(int signal) {
    close(pipeAB);
    close(pipeCB);
    exit(0);
}

int main(int argc, char *argv[]){
    if (signal(SIGUSR1, sig_handler) == SIG_ERR) {
        printf("[%d] ", getpid());
        perror("Error signal ");
        return -1;
    }
    pipeAB = atoi(argv[1]);
    pipeCB = atoi(argv[2]);

    int sizeA;
    int sizeB;
    while(read(pipeAB, &sizeA, sizeof(int)) > 0 && read(pipeCB, &sizeB, sizeof(int)) >
        0){
        printf("[B] Get A: %d; Get C: %d\n", sizeA, sizeB);
    }
```

```

}
```

C.cpp

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

int pipeAC;
int pipeCA;
int pipeCB;


void sig_handler(int signal) {
    close(pipeAC);
    close(pipeCA);
    close(pipeCB);

    exit(0);

}

int main(int argc, char *argv[]){
    if (signal(SIGUSR1, sig_handler) == SIG_ERR) {
        perror("[C] Error signal ");
        return -1;
    }
    pipeAC = atoi(argv[1]);
    pipeCA = atoi(argv[2]);
    pipeCB = atoi(argv[3]);

    int sizeA;
    while(read(pipeAC, &sizeA, sizeof(int))> 0){
        char line[sizeA+1];
        line[sizeA] = '\0';
        read(pipeAC, line, sizeA * sizeof(char));
        printf("[C] Get from A: %s\n", line);

        int ok = 1;
        write(pipeCA, &ok, sizeof(int));

        write(pipeCB, &sizeA, sizeof(int));

    }

}
```

5 Демонстрация работы программ

```
botashev@botashev-laptop:~/ClionProjects/os_labs/course_proj$ ./A
hello
[C] Get from A: hello
[B] Get A: 5; Get C: 5
world
[C] Get from A: world
[B] Get A: 5; Get C: 5
qwertyuioplkmasnfdkja
[C] Get from A: qwertyuioplkmasnfdkja
[B] Get A: 21; Get C: 21
                     f
[C] Get from A:                      f
[B] Get A: 22; Get C: 22
^C
```

6 Вывод

Данная крусовая работа основывается на знаниях полученных в ходе изучения курса. По итогу мы получили нейколько программ, которые взаимодействуют друг с другом с помощью pipe. Задача курсового проекта не сложна в реализации, но ее реализация обощает и закрепляет полученные в курсе знания.