

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной
математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»

Студент: В. А. Амурский
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 А, В, С.

Необходимо создать программу А, которая читает строки из стандартного ввода и посылает их по одной программе С. После отправки каждой строки, программа С отправляет сообщение об успешном получении строки обратно программе А. Программа А не должна отправлять следующую строку до получения подтверждения от программы С.

Программа В отслеживает количество отправленных символов программой А и количество принятых символов программой С. Для этого программа В получает информацию от программ А и С соответственно, и выводит эту информацию в стандартный вывод.

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 А создает два дочерних процесса В и С, затем считывает строки из стандартного потока ввода, после чего передает строки процессу С

Процесс С пишет полученную строку, подтвердив получение строки процессу А

А отправляет размер отправленной строки программе В

С отправляет размер полученной строки программе В

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

```

```

107         kill(id1, SIGUSR1);
108         kill(id2, SIGUSR1);
109     }
110 }
111
112
113     close(pipeAC[0]);
114     close(pipeAC[1]);
115     close(pipeCA[0]);
116     close(pipeCA[1]);
117     close(pipeAB[0]);
118     close(pipeAB[1]);
119     close(pipeCB[0]);
120     close(pipeCB[1]);
121
122
123 }
```

B.cpp

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <signal.h>
5
6  int pipeAB;
7  int pipeCB;
8
9
10 void sig_handler(int signal) {
11     close(pipeAB);
12     close(pipeCB);
13     exit(0);
14 }
15
16 int main(int argc, char *argv[]){
17     if (signal(SIGUSR1, sig_handler) == SIG_ERR) {
18         printf("[%d] ", getpid());
19         perror("Error signal ");
20         return -1;
21     }
22     pipeAB = atoi(argv[1]);
23     pipeCB = atoi(argv[2]);
24
25     int sizeA;
26     int sizeB;
27     while(read(pipeAB, &sizeA, sizeof(int)) > 0 && read(pipeCB, &sizeB, sizeof(int)) >
28         0){
29         printf("[B] Get A: %d; Get C: %d\n", sizeA, sizeB);
30     }
```

```
30 ||
31 || }
```

C.cpp

```
1 ||
2 || #include <stdio.h>
3 || #include <unistd.h>
4 || #include <stdlib.h>
5 || #include <signal.h>
6 ||
7 || int pipeAC;
8 || int pipeCA;
9 || int pipeCB;
10 ||
11 ||
12 || void sig_handler(int signal) {
13 ||     close(pipeAC);
14 ||     close(pipeCA);
15 ||     close(pipeCB);
16 ||
17 ||     exit(0);
18 ||
19 || }
20 ||
21 || int main(int argc, char *argv[]){
22 ||     if (signal(SIGUSR1, sig_handler) == SIG_ERR) {
23 ||         perror("[C] Error signal ");
24 ||         return -1;
25 ||     }
26 ||     pipeAC = atoi(argv[1]);
27 ||     pipeCA = atoi(argv[2]);
28 ||     pipeCB = atoi(argv[3]);
29 ||
30 ||     int sizeA;
31 ||     while(read(pipeAC, &sizeA, sizeof(int))> 0){
32 ||         char line[sizeA+1];
33 ||         line[sizeA] = '\0';
34 ||         read(pipeAC, line, sizeA * sizeof(char));
35 ||         printf("[C] Get from A: %s\n", line);
36 ||
37 ||         int ok = 1;
38 ||         write(pipeCA, &ok, sizeof(int));
39 ||
40 ||         write(pipeCB, &sizeA, sizeof(int));
41 ||
42 ||     }
43 ||
44 || }
```

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