Московский авиационный институт
(национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №1
по курсу «Операционные системы»

ПРОЦЕССЫ В ОС

Студент: Лагуткина Мария Сергеевна
Группа: М8О-206Б-19
Преподаватель: Соколов А. А.
Дата: 11.12.2020
Оценка:

Москва, 2020

1 Постановка задачи

Цель работы
Целью является приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

Задание
Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

В файле записаны команды вида: «число число число endline». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int.

2 Общий метод и алгоритм решения

Аргументом к программе-родителю подается имя файла, в котором лежат данные в формате «число число число endline».Этот файл открывается на чтение. Родительский процесс создает дочерний. Через pipe в дочерний процесс передаются данные из файла. Дочерний процесс считывает данные из своего стандартного потока ввода и считает сумму чисел в каждой строке, затем результат выводит в свой стандартный поток вывода. Результат выводится из родительского потока.

3 Основные файлы программы

child.c

```
1 #include <unistd.h>
2 #include <stdbool.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <ctype.h>
6
7 #define MAX_DIGIT_IN_STIRING  100
8
9 int main(int argc, const char* argv[]) {
```

```
  char c;
  char buf[MAX_DIGIT_IN_STIRING];
  int count_digit = 0;
  int number = 0;
  int result = 0;
  int f_not_left_minus = 0;
  int f_ch = 1;

  memset(buf, '\0', MAX_DIGIT_IN_STIRING);

  while (true) {
    f_ch = read(STDIN_FILENO, &c, sizeof(c));
    if (f_ch == 0){
      return 0;
    }
    if (isdigit(c) || c == '-') {
      if (c == '-' && count_digit  > 0){  //обработка ситуаций типа "1-1"
        f_not_left_minus = 1;
      }
      buf[count_digit] = c;
      count_digit++;
    }
    if (c == ' ' || c == '\n') {
      if (f_not_left_minus == 1){
        memset(buf, '\0', count_digit);
        count_digit = 0;
        f_not_left_minus = 0;
      }else
        if (count_digit > 0 || count_digit >= MAX_DIGIT_IN_STIRING) {
          number = atoi(buf);
          result += number;
          memset(buf, '\0', count_digit);
          count_digit = 0;
        }
    }
    if(c == '\n') {
                write(STDOUT_FILENO, &result, sizeof(result));
      result = 0;
    }
  }
}
```

parent.c

```
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#include <stdio.h>
```

```
7
8  int main(int argc, const char* argv[]){
9    int fd[2];
10   int file;
11   char file_name[50];
12
13   if (argc == 1) {
14     char buff[] = "Enter file name: ";
15     write(STDOUT_FILENO, &buff, sizeof(buff) - 1);
16     read(STDIN_FILENO, &file_name, sizeof(file_name));
17     int i;
18     for (i = 0; i < 50; i++) {
19           if (file_name[i] == '\n' || file_name[i] == '\0') {
20           break;
21       }
22     }
23       if(file_name[i] == '\n') {
24                 file_name[i] = '\0';
25           }
26   } else {
27     strcpy(file_name, argv[1]);
28   }
29
30     if (pipe(fd) < 0) {
31         char message[] = "pipe error\n";
32         write(STDERR_FILENO, &message, sizeof(message) - 1);
33         return 1;
34     }
35
36   pid_t pid;
37
38   switch(pid=fork()) {
39     case -1 : //ошибка при вызове fork()
40     ;char message[] = "fork error\n";
41           write(STDERR_FILENO, &message, sizeof(message) - 1);
42           return 2;
43
44   case 0 : //это код потомка
45           file = open(file_name, O_RDONLY);
46           if (file < 0) {
47                 char message[] = "can't open file\n";
48                 write(STDERR_FILENO, &message, sizeof(message) - 1);
49                 return 3;
50           }
51     dup2(file, STDIN_FILENO);
52           dup2(fd[1], STDOUT_FILENO);
53           dup2(fd[1], STDERR_FILENO);
54           close(file);
55           close(fd[0]);
```

```
56            close(fd[1]);
57
58          if (execl("child", "child", (char *) NULL) == -1) {
59              char message[] = "exec error\n";
60              write(STDERR_FILENO, &message, sizeof(message) - 1);
61              return 4;
62      }
63
64    default : //код родительского процесса
65      close(fd[1]);
66            waitpid(pid, (int *)NULL, 0);
67            int result;
68            while (read(fd[0], &result, sizeof(result))) {
69                  char buff[50];
70                  sprintf(buff, "%d\n", result);
71                  write(STDOUT_FILENO, &buff, strlen(buff));
72            }
73            close(fd[0]);
74      }
75      return 0;
76  }
```

4 Пример работы

```
test1.txt
1 2 4 6 3

0 10
1
maria@maria-MS-7817:~/project/os2$ ./parent test1.txt
16
0
10
1
test2.txt

maria@maria-MS-7817:~/project/os2$ ./parent test2.txt
0
test3.txt

0 -2 5 6 34 -1
9 0 -23456 23456
-23
```

```
4 -6 0 -2
maria@maria-MS-7817:~/project/os2$ ./parent test3.txt
0
42
9
-23
-4
test4.txt
2 -4-2
1-1
maria@maria-MS-7817:~/project/os2$ ./parent test4.txt
2
0
```

5 Вывод

В этой лабораторной я узнала как работать с процессами, узнала о том, как можно организовать передачу данных в дочерний процесс. Поработала с системными вызовами такими как, fork, exec, pipe. Создала свой первые процессы.