МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Лабораторная работа №2
по курсу "Операционные системы"
3 семестр

Студент: Леухин М. В.
Группа: М8О-206Б-20
Преподаватель: Соколов А. А.
Дата: 09.10.21
Оценка: 5
Подпись: _____

Москва, 2021

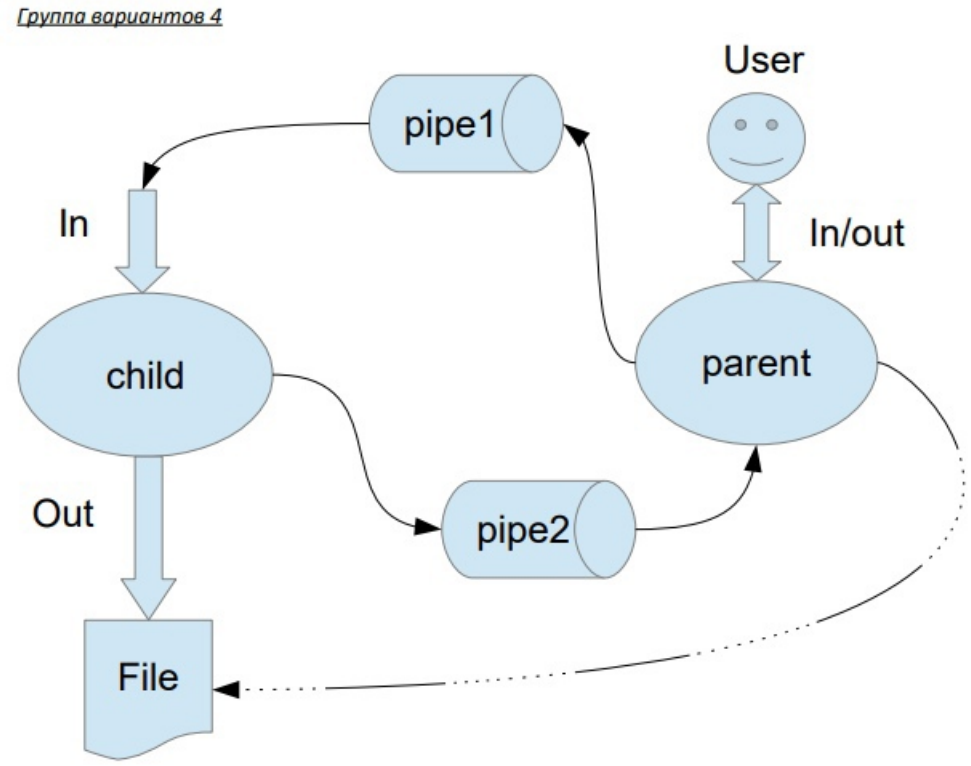
Содержание

1 Постановка задачи

Цель работы: приобретение практических навыков в управлении процессами в ОС и обеспечении обмена данных между процессами посредством каналов.

Задание: составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними. В результате работы программы основной процесс должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Вариант 15: родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода. Правило проверки: строка должна начинаться с заглавной буквы.



2 Основная часть

2.1 Листинг программы

Файл parent.c:

```
#define _GNU_SOURCE

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

void handle_error(bool expr, char* msg) {
    if (expr) {
        write(fileno(stdout), msg, strlen(msg) * sizeof(char));
        exit(-1);
    }
}

void print_t(char* str, char* msg){
    char* full_msg = malloc((strlen(str) + strlen(msg)) *
       sizeof(char));
    strcat(full_msg, str); strcat(full_msg, msg);
    write(fileno(stdout), full_msg, strlen(full_msg) *
       sizeof(char));
}

void clean(char* str){
    for (int i = 0 ; i < strlen(str); ++i){
        if (str[i] == '\n'){ str[i] = '\0'; }
    }
}

int main(){

    int src_fd[2];
    int pipe_response = pipe(src_fd);
    handle_error(pipe_response == -1, "pipe error");

    int err_fd[2];
    pipe_response = pipe(err_fd);
    handle_error(pipe_response == -1, "pipe error");

    pid_t id = fork();
    handle_error(id == -1, "fork error");

```

```
42      if (id == 0){
43
44          char name[64];
45          read(src_fd[0], &name, sizeof(name));
46
47          char *src_fd_0, *src_fd_1, *err_fd_0, *err_fd_1;
48          asprintf(&src_fd_0, "%d", src_fd[0]);
49          asprintf(&src_fd_1, "%d", src_fd[1]);
50          asprintf(&err_fd_0, "%d", err_fd[0]);
51          asprintf(&err_fd_1, "%d", err_fd[1]);
52
53          execl("child.out", name, src_fd_0, src_fd_1, err_fd_0,
                err_fd_1, NULL);
54
55      } else {
56
57          char* parent; int parent_pid = getpid();
58          asprintf(&parent, "[%d] PARENT. ", parent_pid);
59
60          print_t(parent, "Enter the name of file to write: ");
61          char name[256];
62          read(fileno(stdin), name, 256); clean(name);
63          write(src_fd[1], &name, sizeof(name));
64          bool file_error; read(err_fd[0], &file_error,
                sizeof(bool));
65          if (file_error){
66              close(src_fd[0]); close(src_fd[1]);
67              close(err_fd[0]); close(err_fd[1]);
68              handle_error(true, "file error\n");
69          }
70
71          char str[256];
72          print_t(parent, "Enter string: ");
73          while (read(fileno(stdin), str, 256) != 0){
74              clean(str);
75              write(src_fd[1], &str, sizeof(str));
76              bool err;
77              read(err_fd[0], &err, sizeof(bool));
78              if (err){
79                  char* err_msg;
80                  asprintf(&err_msg, "Error: \"%s\" is not
                        valid.\n", str);
81                  print_t(parent, err_msg);
82              }
83              print_t(parent, "Enter string: ");
84          }
85          write(src_fd[1], "_quit", sizeof(str));
86
87      }
88      write(fileno(stdout), "\n", sizeof "\n");
```

```
    close(src_fd[0]); close(src_fd[1]);
    close(err_fd[0]); close(err_fd[1]);
    return 0;

}
```

Файл child.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argv, char* argc[]){

    int src_fd[2], err_fd[2];
    src_fd[0] = atoi(argc[1]);
    src_fd[1] = atoi(argc[2]);
    err_fd[0] = atoi(argc[3]);
    err_fd[1] = atoi(argc[4]);

    char* name = argc[0];
    int output_fd = open(name, O_WRONLY | O_CREAT);
    bool file_error = false;
    if (output_fd < 0){ file_error = true; }
    write(err_fd[1], &file_error, sizeof(bool));
    if (file_error){
        close(src_fd[0]); close(src_fd[1]);
        close(err_fd[0]); close(err_fd[1]);
    }

    char str[256];
    read(src_fd[0], &str, sizeof(str));
    read(src_fd[0], &str, sizeof(str));
    while(strcmp(str, "_quit") != 0){
        bool err;
        if (str[0] >= 'A' && str[0] <= 'Z'){
            err = false;
            write(output_fd, str, strlen(str) * sizeof(char));
            write(output_fd, "\n", sizeof "\n");
        } else {
            err = true;
        }
        write(err_fd[1], &err, sizeof(bool));
        read(src_fd[0], &str, sizeof(str));
    }
    close(src_fd[0]); close(src_fd[1]);
```

```
44      close(err_fd[0]); close(err_fd[1]);
45  }
```

2.2 Результат работы программы

```
1  matvey@matvey-Lenovo-IdeaPad-S340-15API:~/labs/2os/2lab/src$
       ./a.out
2  [24647] PARENT. Enter the name of file to write: output.txt
3  [24647] PARENT. Enter string: This is valid string
4  [24647] PARENT. Enter string: invalid str
5  [24647] PARENT. Error: "invalid str" is not valid.
6  [24647] PARENT. Enter string:
7  matvey@matvey-Lenovo-IdeaPad-S340-15API:~/labs/2os/2lab/src$ cat
       output.txt
8  This is valid string
```

3 Вывод

В ходе выполнения лабораторной работы я познакомился с тем, что такое процессы в операционной системе и узнал о том, как можно их создавать. Я научился использовать инструменты для осуществления взаимодействия между процессами, а именно pipe — однонаправленный канал межпроцессного взаимодействия.