Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

Лабораторная работа №2 по курсу “Операционные системы”

Студент: Живалев Е.А.

Группа: М8О-206Б

Преподаватель: Соколов А.А.

Вариант: 27

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2019

1 Задание

Программе на вход поступает число n - количество процессов, которое будет создано в ходе выполнения и программы, и n названий файлов, в которые эти процессы будут записывать данные. Родительский процесс создает n дочерних процессов, которым поочередно передает символы с входной строки. Дочерний процесс записывает полученные символ в переданный ему файл.

В ходе выполнения лабораторной работы были использованы следующие системные вызовы:

- read - использовался для чтения данных с входной строки
- write - использовался для записи данных в файлы
- fork - использовался для создания дочерних процессов
- pipe - использовался для создания однонаправленных, использованных для передачи родительским процессом символов дочерним процессам

2 Описание работы программы

С помощью функции readLine, которая, используя read считывает одну строку с входной строки, программа получает количество процессов(число n), которые необходимо создать и имена файлов, в которые должна производиться запись. Затем создаётся n пайпов и n процессов. Затем в родительском процессе в цикле с условием, что возвращенное функцией read число больше 0 происходит запись считанного символа в определенный пайп. В то же время дочерний процесс считывает из пайпа символ и записывает его в свой файл.

3 Исходный код

main.c

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

#define FILE_NAME_LENGTH 129

void readLine(char* result) {
    char c;
    int byteCounter = 0;
    while(byteCounter < FILE_NAME_LENGTH) {
        if(read(STDIN_FILENO, &c, 1) == -1) {
            perror("Reading failed");
        }
        if(c == '\n') {
            break;
        } else {
            result[byteCounter++] = c;
        }
    }
    result[byteCounter] = '\0';
}

int main() {
    pid_t pid;
    char c;
    int numberOfProcesses = 1;
    char buf[FILE_NAME_LENGTH];
    readLine(buf);
    if(atoi(buf) > 0) {
        numberOfProcesses = atoi(buf);
    } else {
        printf("Invalid argment - must be a positive number\n");
        exit(-1);
    }
    int pipes[numberOfProcesses][2];
    char fileNames[numberOfProcesses][FILE_NAME_LENGTH];
    int processNumber = 0;
    for(int i = 0; i < numberOfProcesses; ++i) {
        readLine(buf);
        strcpy(fileNames[i], buf);
    }
    for(int i = 0; i < numberOfProcesses; ++i) {
        if(pipe(pipes[i]) == -1) {
            perror("Pipe failed");
        }
    }
    for(int i = 0; i < numberOfProcesses; ++i) {
       pid = fork();
       if(pid == 0) {
            processNumber = i;
            break;
```

```
57          } else if (pid == -1) {
58              perror("Fork failed");
59              return -1;
60          }
61      }
62      if(pid > 0) {
63          for(int i = 0; i < numberOfProcesses; ++i) {
64             close(pipes[i][0]);
65          }
66          while(read(STDIN_FILENO, &c, 1) > 0) {
67              if(write(pipes[processNumber][1], &c, 1) == -1) {
68                  perror("Writing failed");
69              }
70              processNumber++;
71              processNumber %= numberOfProcesses;
72          }
73          for(int i = 0; i < numberOfProcesses; ++i) {
74              char terminator = '\0';
75              if(write(pipes[i][1], &terminator, 1) == -1) {
76                  perror("Writing failed");
77              }
78              close(pipes[i][1]);
79          }
80      } else if(pid == 0) {
81          int fd = open(fileNames[processNumber], O_WRONLY, O_CREAT,
    S_IRWXU);
82          if(ftruncate(fd, 0) == -1) {
83              printf("errno %i\n", errno);
84              perror("ftruncate failed");
85          }
86          while(1) {
87              char c;
88              close(pipes[processNumber][1]);
89              if(read(pipes[processNumber][0], &c, 1) == -1) {
90                  perror("Reading failed");
91              }
92              if(c == '\0') {
93                  close(pipes[processNumber][0]);
94                  close(fd);
95                  exit(0);
96              } else {
97                  if(write(fd, &c, 1) == -1) {
98                      perror("Writing failed");
99                  }
100             }
101         }
102     }
103     return 0;
104 }
```

4 Консоль

```
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_2/src$ ./a.out
3
1.txt
2.txt
3.txt
ttthhhiiisss ssshhhooouuullldddd bbbeee iiinnn fffiiillleee123
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_2/src$ cat 1.txt
this should be in file1
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_2/src$ cat 2.txt
this should be in file2
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_2/src$ cat 3.txt
this should be in file3
```

5 Выводы

В ходе выполнения лабораторной работы я познакомился с некоторым системными вызовами, например, fork, pipe, а также увидел как их можно применять на практике. Также я более подробно изучил как устроены процессы и пайпы в ОС Linux.