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

### Лабораторная работа №2 по курсу "Операционные системы"

$\Gamma pynn$	na: M8O-206B
Преподаватель:	Соколов А.А.
	Вариант: 27
Оценка:	
Дата:	·
$\Pi o \partial nuc$ ь:	·

Студент: Живалев Е.А.

#### 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>
2 #include <string.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <errno.h>
7 #include <stdlib.h>
8 #include <fcntl.h>
10 #define FILE_NAME_LENGTH 129
12 void readLine(char* result) {
      char c;
      int byteCounter = 0;
14
      while(byteCounter < FILE_NAME_LENGTH) {</pre>
           if(read(STDIN_FILENO, &c, 1) == -1) {
               perror("Reading failed");
17
18
           if(c == '\n') {
19
               break;
           } else {
21
               result[byteCounter++] = c;
22
      }
      result[byteCounter] = '\0';
25
26 }
27
28 int main() {
      pid_t pid;
29
      char c;
30
      int numberOfProcesses = 1;
31
32
      char buf[FILE_NAME_LENGTH];
      readLine(buf);
33
      if(atoi(buf) > 0) {
34
           numberOfProcesses = atoi(buf);
      } else {
36
           printf("Invalid argment - must be a positive number\n");
37
           exit(-1);
38
      }
39
      int pipes[numberOfProcesses][2];
      char fileNames[numberOfProcesses][FILE_NAME_LENGTH];
41
      int processNumber = 0;
      for(int i = 0; i < numberOfProcesses; ++i) {</pre>
43
           readLine(buf);
44
           strcpy(fileNames[i], buf);
45
46
      for(int i = 0; i < numberOfProcesses; ++i) {</pre>
           if(pipe(pipes[i]) == -1) {
48
               perror("Pipe failed");
49
50
      for(int i = 0; i < numberOfProcesses; ++i) {</pre>
52
          pid = fork();
          if(pid == 0) {
               processNumber = i;
56
               break;
```

```
} else if (pid == -1) {
                perror("Fork failed");
58
                return -1;
60
       }
61
       if(pid > 0) {
           for(int i = 0; i < numberOfProcesses; ++i) {</pre>
63
              close(pipes[i][0]);
64
           while(read(STDIN_FILENO, &c, 1) > 0) {
66
                if(write(pipes[processNumber][1], &c, 1) == -1) {
67
                    perror("Writing failed");
69
                processNumber++;
                processNumber %= numberOfProcesses;
           for(int i = 0; i < numberOfProcesses; ++i) {</pre>
                char terminator = '\0';
                if(write(pipes[i][1], &terminator, 1) == -1) {
                    perror("Writing failed");
                }
77
                close(pipes[i][1]);
           }
79
       } else if(pid == 0) {
80
           int fd = open(fileNames[processNumber], O_WRONLY, O_CREAT,
       S_IRWXU);
           if(ftruncate(fd, 0) == -1) {
82
                printf("errno %i\n", errno);
83
                perror("ftruncate failed");
           }
85
           while(1) {
86
                char c;
                close(pipes[processNumber][1]);
                if(read(pipes[processNumber][0], &c, 1) == -1) {
89
                    perror("Reading failed");
90
                }
91
                if(c == '\0') {
                    close(pipes[processNumber][0]);
93
                    close(fd);
94
                    exit(0);
95
                } else {
96
                    if(write(fd, &c, 1) == -1) {
97
                        perror("Writing failed");
98
                    }
99
                }
           }
       }
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 ssshhhooouuulllddd 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.