МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №2 по курсу
«Разработка ПО ОС UNIX»

Выполнила студентка группы Б17-501: Баранова Д.Д.

Проверил: Ктитров С.В.

Москва, 2020

Задание (Вариант B–2)

Разработать программу, запускающую две задаваемые программы на выполнение, соединяя их в конвейер. Программы могут иметь аргументы. Программу оформить как утилиту командной строки.

Выполнение

Использовались библиотеки:

- unistd.h для подключения функции fork
- sys/types.h для подключения типа данных pid_t
- unistd.h для работы с потоками ввода/вывода

Код программы .sh

```
#!/usr/bin/env bash
cc main.c -o main
chmod +x main
flag=0
n=$#
myArray=()
while [ $n -gt 0 ]
do
    let n=$n-1
    case "$1" in
    --program)
        if [ $flag -eq 1 ]
        then
            flag=2
            myArray+=( "###" )
        fi
        if [ $flag -eq 0 ]
        then
            flag=1
        fi
        shift ;;

    *)  myArray+=( "$1" )
        shift ;;
    esac

done
./main ${myArray[*]}
```

Скрипт парсит поступившие на вход аргументы. Вызов программы должен происходить с указанием параметра “- -program” перед именем каждой из программ, которые нужно запустить. Это сделано для того, чтобы корректно определить какие параметры поступают для запуска первой программы, а какие для второй. Таким образом, примеры запуска программы могут быть:

*./B.sh - -program PROGRAM1 –a –b –c - -program PROGRAM2 –d –e hello world **

В этом случае будет запущена программа **PROGRAM1** с аргументами **–a –b –c** и вторая программа **PROGRAM2** с аргументами **–d –e hello world ***.

./B.sh - -program PROGRAM1 - -program PROGRAM2 –h

В этом случае будет запущена программа **PROGRAM1** без аргументов и вторая программа **PROGRAM2** с аргументом **-h**.

Код программы .c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>
#include <string.h>

int main(int argc, char* argv[]) {
    /* Parse arguments: "program1 -a -b -c ### program2 -d -e -f" */
    if (argc <= 3) {
        char *msg = "Недостаточно аргументов";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(EXIT_FAILURE);
    }

    int first_size = 0;
    char border = 0;
    int second_size = 0;
    for (int i = 2; i < argc; i++) {
        if (strcmp(argv[i], "###") == 0) {
            border = 1;
        }
        if (border) {
            second_size++;
        } else {
            first_size++;
        }
    }
    second_size--;
    char* argv1[first_size + 2];
    char* argv2[second_size + 2];

    argv1[0] = argv[1];
    for (int i = 0; i < first_size; i++) {
        argv1[i + 1] = argv[i + 2];
    }
    argv1[first_size + 1] = NULL;

    argv2[0] = argv[first_size + 3];
    for (int i = 0; i < second_size; i++) {
        argv2[i + 1] = argv[first_size + 4 + i];
    }
    argv2[second_size + 1] = NULL;

    pid_t pid;
    int pipefd[2];
    pipe(pipefd);
    switch(pid=fork()) {
      case -1:
        perror("Ошибка при создании дочернего процесса");
        exit(EXIT_FAILURE);
      case 0:
            /* child process */
        dup2(pipefd[1], STDOUT_FILENO); // перенаправить стандартный вывод, путем дублирования его дескриптора
        execv(argv1[0], argv1);
        fprintf(stderr, "Ошибка запуска '%s'\n", argv1[0]);
        exit(EXIT_FAILURE);
      default:
        /* parent process */
        waitpid(pid,0,0); // ждем пока завершится дочерний процесс

        switch(pid=fork()) {
          case -1:
            perror("Ошибка при создании дочернего процесса");
            exit(EXIT_FAILURE);
          case 0:
            dup2(pipefd[0], STDIN_FILENO);
            dup2(pipefd[1], STDOUT_FILENO);
            close(pipefd[1]);
            execv(argv2[0], argv2);
            fprintf(stderr, "Ошибка запуска '%s'\n", argv2[0]);
            exit(EXIT_FAILURE);
```

```
74          default:
75            close(pipefd[0]);
76            close(pipefd[1]);
77            waitpid(pid,0,0);
78        }
79    }
80 }
81
```

Пример работы:

В примере использовались две дополнительно реализованные программы program1.c и program2.c:

Program1.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char* argv[]) {
5     fprintf(stdout, "Some data from program1, args number: '%d'\n", argc - 1);
6     return 0;
7 }
8
```

Программа выводит в поток stdout сообщение «Some data from program1» с указанием количества аргументов, которые поступили на вход программе.

Program2.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char* argv[]) {
5     FILE* out = freopen("output.txt", "w", stdout);
6
7     const int BUFFER_SIZE = 1 << 10;
8     char BUFFER[BUFFER_SIZE];
9     while (fgets(BUFFER, sizeof(BUFFER), stdin)) {
10        fprintf(out, "%s", BUFFER);
11    }
12    fprintf(out, "Some data from program2, args number: '%d'\n", argc - 1);
13    fflush(out);
14    fclose(out);
15
16    return 0;
17 }
```

Программа читает из потока stdin поступающую информацию и выводит в файл output.txt поступившие на вход данные, присоединяя к ним свое сообщение «Some data from program2» с указанием количества аргументов, которые поступили на вход программе.

Пример 1

После выполнения команды:

```
super@super-Inspiron-5558:~/Desktop/unik/B$ ./B2.sh --program program1 -a -b --program program2 -c -d -e -s
super@super-Inspiron-5558:~/Desktop/unik/B$
```

Файл output.txt :

```
Some data from program1, args number: '2'

Some data from program2, args number: '4'
```

Пример 2

После выполнения команды:

```
super@super-Inspiron-5558:~/Desktop/unik/B$ ./B2.sh --program /bin/echo hello world --program program2 -c -d -e
super@super-Inspiron-5558:~/Desktop/unik/B$
```

Файл output.txt :

```
hello world

Some data from program2, args number: '3'
```