

**Индивидуальное домашнее задание по дисциплине операционные
системы №1**

Вариант 4

Мерочкин Илья БПИ218

4. Разработать программу, находящую в заданной ASCII-строке последнюю при перемещении слева направо последовательность **N** символов, каждый элемент которой определяется по условию «больше предшествующего» (***N** вводится как отдельный параметр*).

4 балла

Разработана программа, осуществляющая взаимодействие между тремя родственными процессами с использованием неименованных каналов.

Код программы лежит в mark4/main.c.

В программе используется три родственных процесса - процесс main (будем говорить, что это первый процесс) создает новый процесс посредством вызова fork (будем говорить, что это второй процесс) и этот новый процесс также создает новый процесс посредством вызова fork (будем говорить, что это третий процесс). Третий процесс считывает данные из входного файла и записывает их в канал pipe_to_read. Второй процесс считывает данные из канала pipe_to_read, обрабатывает их, находит искомую подстроку и записывает ее в канал pipe_to_write (если данной строки не существует, то в канал записывается "Substring not found". И первый процесс считывает данные из канала pipe_to_write и выводит их в выходной файл.

Для задания имен входного и выходного файла, а также для задания длины искомой подстроки используются аргументы командной строки.

Ввод и вывод данных при работе с файлами осуществляется через системные вызовы *read* и *write*.

Размер буферов для хранения данных равен 8192 байта.

В файле приведены 5 тестов, запускать и тестировать стоит следующим образом:

```
gcc main.c -o ./main
```

Тест 1:

```
./main input_tests/input_test1.txt ouput_tests/output_test1.txt 2
```

Тест 2:

```
./main input_tests/input_test2.txt ouput_tests/output_test2.txt 4
```

Тест 3:

```
./main input_tests/input_test3.txt ouput_tests/output_test3.txt 3
```

Тест 4:

```
./main input_tests/input_test4.txt ouput_tests/output_test4.txt 1
```

Тест 5:

```
./main input_tests/input_test5.txt ouput_tests/output_test5.txt 2
```

5 баллов

Разработана программа, осуществляющая взаимодействие между тремя родственными процессами с использованием именованных каналов.

Код программы лежит в mark5/main.c

В программе используется три родственных процесса - процесс main (будем говорить, что это первый процесс) создает новый процесс посредством вызова fork (будем говорить, что это второй процесс) и этот новый процесс также создает новый процесс посредством вызова fork (будем говорить, что это третий процесс). Третий процесс считывает данные из входного файла и записывает их в именованный канал `fifo_to_read`. Второй процесс считывает данные из именованного канала `fifo_to_read`, обрабатывает их, находит искомую подстроку и записывает ее в канал `fifo_to_write` (если данной строки не существует, то в именованный

канал записывается “Substring not found”. И первый процесс считывает данные из именованного канала `fifo_to_write` и выводит их в выходной файл.

Именованные каналы создаются посредством вызова функции `mknod`.

Для задания имен входного и выходного файла, а также для задания длины искомой подстроки используются аргументы командной строки.

Ввод и вывод данных при работе с файлами осуществляется через системные вызовы *read* и *write*.

Размер буферов для хранения данных равен 8192 байта.

В файле приведены 5 тестов, запускать и тестировать стоит следующим образом:

```
gcc main.c -o ./main
```

Тест 1:

```
./main input_tests/input_test1.txt ouput_tests/output_test1.txt 2
```

Тест 2:

```
./main input_tests/input_test2.txt ouput_tests/output_test2.txt 4
```

Тест 3:

```
./main input_tests/input_test3.txt ouput_tests/output_test3.txt 3
```

Тест 4:

```
./main input_tests/input_test4.txt ouput_tests/output_test4.txt 1
```

Тест 5:

```
./main input_tests/input_test5.txt ouput_tests/output_test5.txt 2
```

6 баллов

Разработана программа, которая осуществляет взаимодействие между двумя родственными процессами с использованием неименованных каналов.

Код программы лежит в `mark6/main.c`

В программе используется два процесса - процесс `main` (будем говорить, что это первый процесс) и порожденный с помощью вызова `fork` второй процесс. Взаимодействие между ними осуществляется с помощью неименованного канала, созданного вызовом функции `pipe`. Также для синхронизации двух процессов в программе используются 2 именованных семафора с именами `"/read-semaphore"` и `"/write-semaphore"`. Изначально первый процесс начинает считывать данные из файла, а второй процесс ждет, пока семафор `"/read-semaphore"` разблокируется. Когда первый процесс записывает все данные в пайп, то семафор `"/read-semaphore"` разблокируется. Второй процесс считывает данные из канала, а первый ждет, когда семафор `"/write-semaphore"` разблокируется. Второй процесс находит ответ, записывает его в канал и разблокирует семафор `"/write-semaphore"`. Тогда первый семафор считывает данные из канала и выводит ответ в файл.

Для задания имен входного и выходного файла, а также для задания длины искомой подстроки используются аргументы командной строки.

Ввод и вывод данных при работе с файлами осуществляется через системные вызовы `read` и `write`.

Размер буферов для хранения данных равен 8192 байта.

В файле приведены 5 тестов, запускать и тестировать стоит следующим образом:

```
gcc main.c -o ./main
```

Тест 1:

```
./main input_tests/input_test1.txt ouput_tests/output_test1.txt 2
```

Тест 2:

```
./main input_tests/input_test2.txt ouput_tests/output_test2.txt 4
```

Тест 3:

```
./main input_tests/input_test3.txt ouput_tests/output_test3.txt 3
```

Тест 4:

`./main input_tests/input_test4.txt ouput_tests/output_test4.txt 1`

Тест 5:

`./main input_tests/input_test5.txt ouput_tests/output_test5.txt 2`

7 баллов

Разработана программа, которая осуществляет взаимодействие между двумя родственными процессами с использованием именованных каналов.

Общая схема решения идентична схеме на 6 баллов, за исключением того, что здесь используются именованный канал с именем “fifo”.

Тесты и остальные разделы задания остались идентичны.

8 баллов

Разработана программа, которая осуществляет взаимодействие между двумя неродственными процессами, с использованием именованных каналов.

Общая схема решаемой задачи схожа с схемой решения на 7 баллов. В файле `read_writer.c` описан ввод и вывод данных, в файле `main.c` обработка данных в соответствии с заданием. Процессы обмениваются данными через именованный канал `fifo`, а синхронизация осуществляется с помощью двух семафоров (более подробно расписано в заданиях на 4 - 6 баллов).

Есть три способа запуска программы:

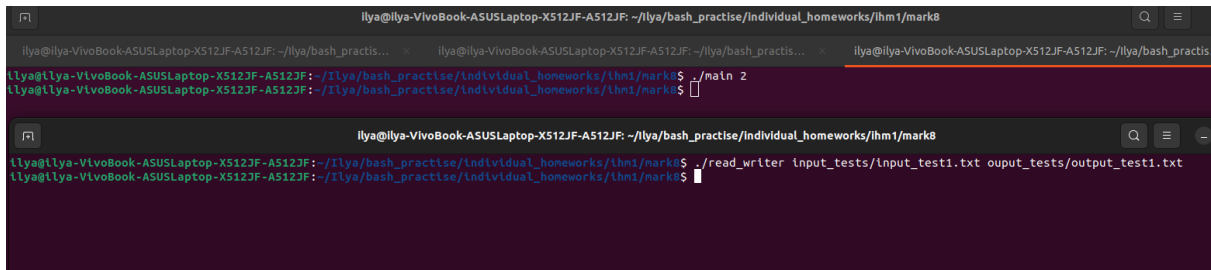
Для начала скомпилируем программы командами:

```
gcc main.c -o ./main
```

```
gcc read_writer.c -o ./read_write
```

Способ 1

Просто откроем два окна терминала, и запустим каждую программу в отдельном окне. Например:



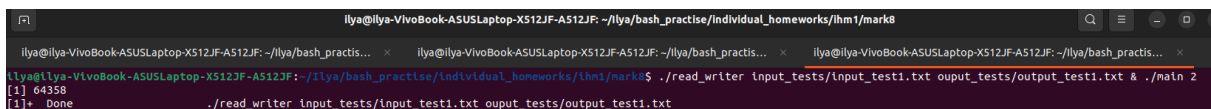
```
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8$ ./main 2
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8$

ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8$ ./read_writer input_tests/input_test1.txt ouput_tests/output_test1.txt
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8$
```

(В задании на данный балл в программу `./read_writer` аргументом командной строки мы передаем имена файлов для ввода и вывода, а в программу `./main` длину искомой подстроки).

Способ 2:

Ввести исполняемые файлы одной командной, разделив `&`. Например:



```
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8
ilya@ilya-VivoBook-ASUSLaptop-X512JF-A512JF: ~/ilya/bash_practise/individual_homeworks/ihm1/mark8$ ./read_writer input_tests/input_test1.txt ouput_tests/output_test1.txt & ./main 2
[1] 64358
[1]+ Done
./read_writer input_tests/input_test1.txt ouput_tests/output_test1.txt
```

Способ 3 (рекомендуемый):

В директории `mark8` есть `bash`-скрипт `run.sh`, который сам запускает исполняемые файлы способом номер 2. Поэтому, чтобы не вводить долгие команды, можно просто запустить `./run.sh input.txt output.txt 2` (не забудьте дать `run.sh` файлу нужные права). Пример:

```
./run.sh input_tests/input_test1.txt ouput_tests/output_test1.txt 2
```

Тестовые файлы также находятся в директориях `input_test` и `output_tests` соответственно. И запустить их можно следующим образом:

Тест 1:

```
./run.sh input_tests/input_test1.txt ouput_tests/output_test1.txt 2
```

Тест 2:

```
./run.sh input_tests/input_test2.txt ouput_tests/output_test2.txt 4
```

Тест 3:

```
./run.sh input_tests/input_test3.txt ouput_tests/output_test3.txt 3
```

Тест 4:

```
./run.sh input_tests/input_test4.txt ouput_tests/output_test4.txt 1
```

Тест 5:

```
./run.sh input_tests/input_test5.txt ouput_tests/output_test5.txt 2
```