Отчет ИДЗ 1

Хамид Карим Мохамадович БПИ 214
Вариант 26
Работа выполнена на оценку 8

Задание:

Разработать программу, которая определяет количество целых чисел в ASCII-строке. Числа состоят из цифр от 0 до 9. Разделителями являются все другие символы.

Использовались буферы размера 10000 байт, поэтому удовлетворяет задаче использования минимум 5000 байтного буфера.

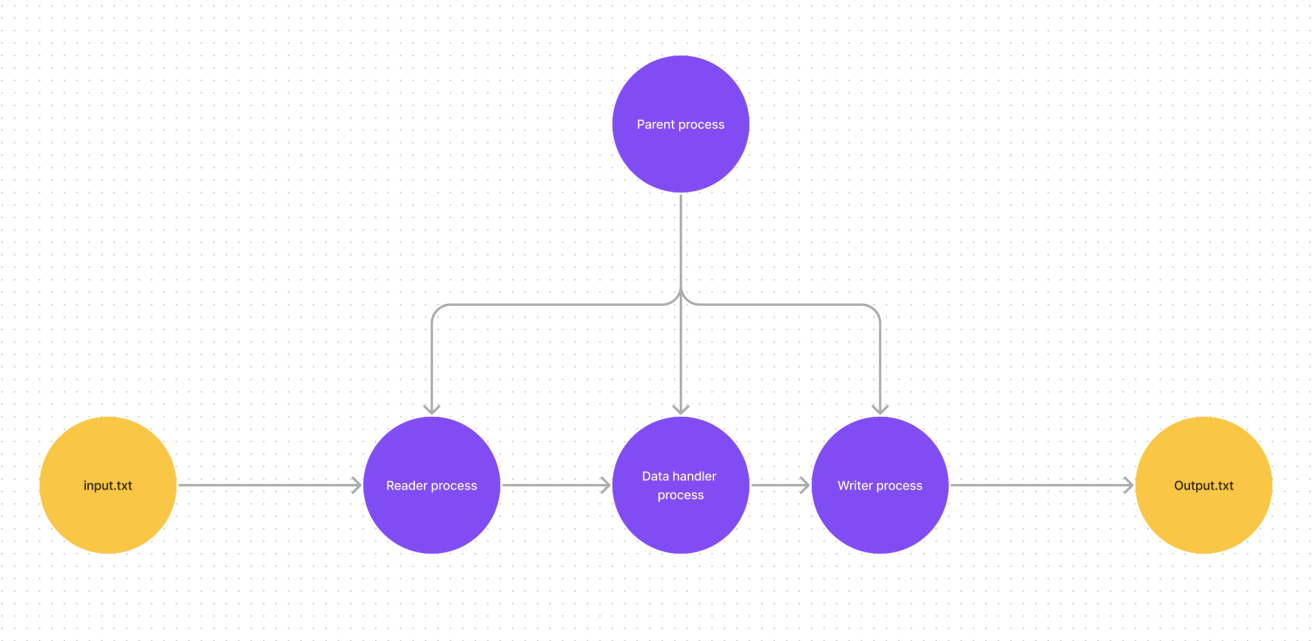
Для решения проблемы доступа к ресурсу чтения и записи(тут это каналы) использовал семафоры.

Тестовые файлы и результат их работы находятся в папке test. Файлы одинаковые для всех оценок, так как программа обработчик одна и та же и нет смысла делать разные файлы.

Все файлы на каждую оценку находятся в папках <оценка>_mark.

На оценку 4

Общая схема решаемой задачи:



Для запуска программы, необходимо ввести команду:

```
$ gcc main.c -o main -lpthread
$ ./main ../tests/input1.txt ../tests/output1.txt
reader started
Reader exit
process started
Process exit
writer started
Writer exit

Parent: All children have exited.
```

Как видим, все сработало без ошибок. Проверим файл output1.txt в папке tests:

```
11
```

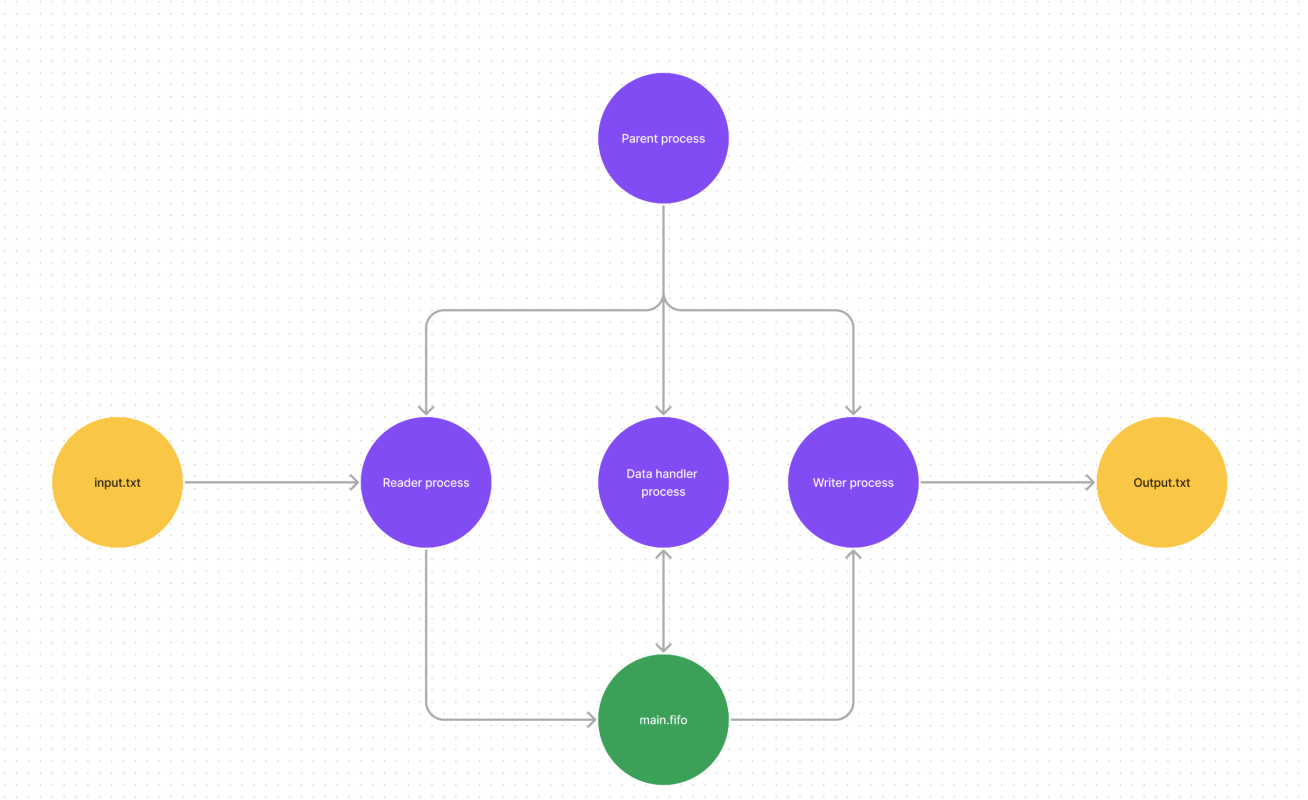
Как видим, программа работает корректно.

Работа с файлами производится с помощью системных вызовов open, read, write, close. Для синхронизации доступа к каналам используются семафоры.

Выполнены все требования к оценке 4.

На оценку 5

Общая схема(каналы были созданы с помощью функции mknod):



Пример запуска программы, где последний аргумент это имя канала:

```
$ gcc main.c -o main -lpthread
$ ./main ../tests/input1.txt ../tests/output1.txt main.fifo
```

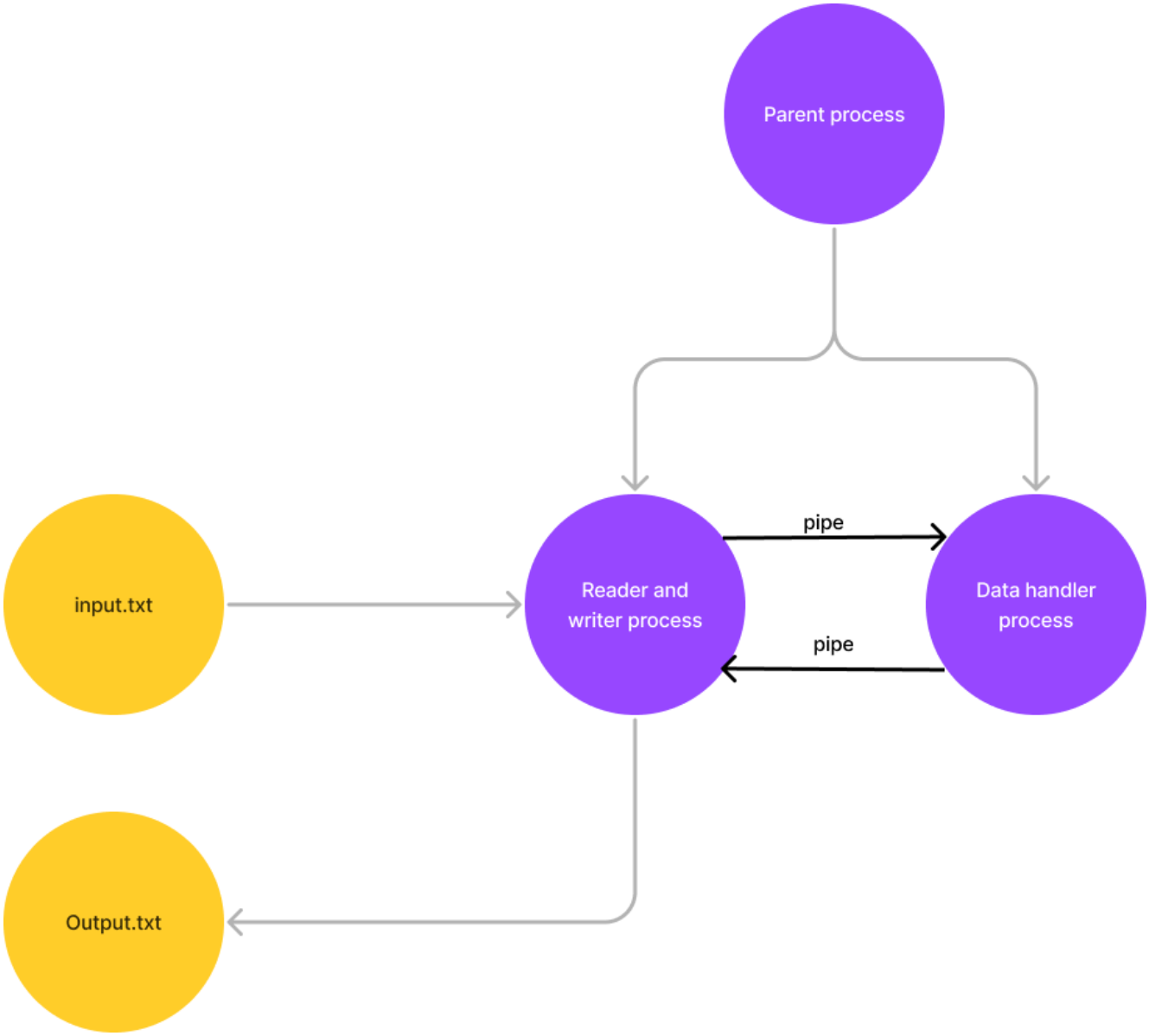
```
reader started
Reader exit
process started
Process exit
writer started
Writer exit

Parent: All children have exited.
```

Код проходит все тесты. Синхронизирован с помощью семафоров.

На оценку 6

Общая схема(каналы были созданы с помощью функции pipe). И не совсем понятно по заданию как указать имена каналов, если они неименованные):



Для синхронизации также использовались семафоры.

Запуск программы:

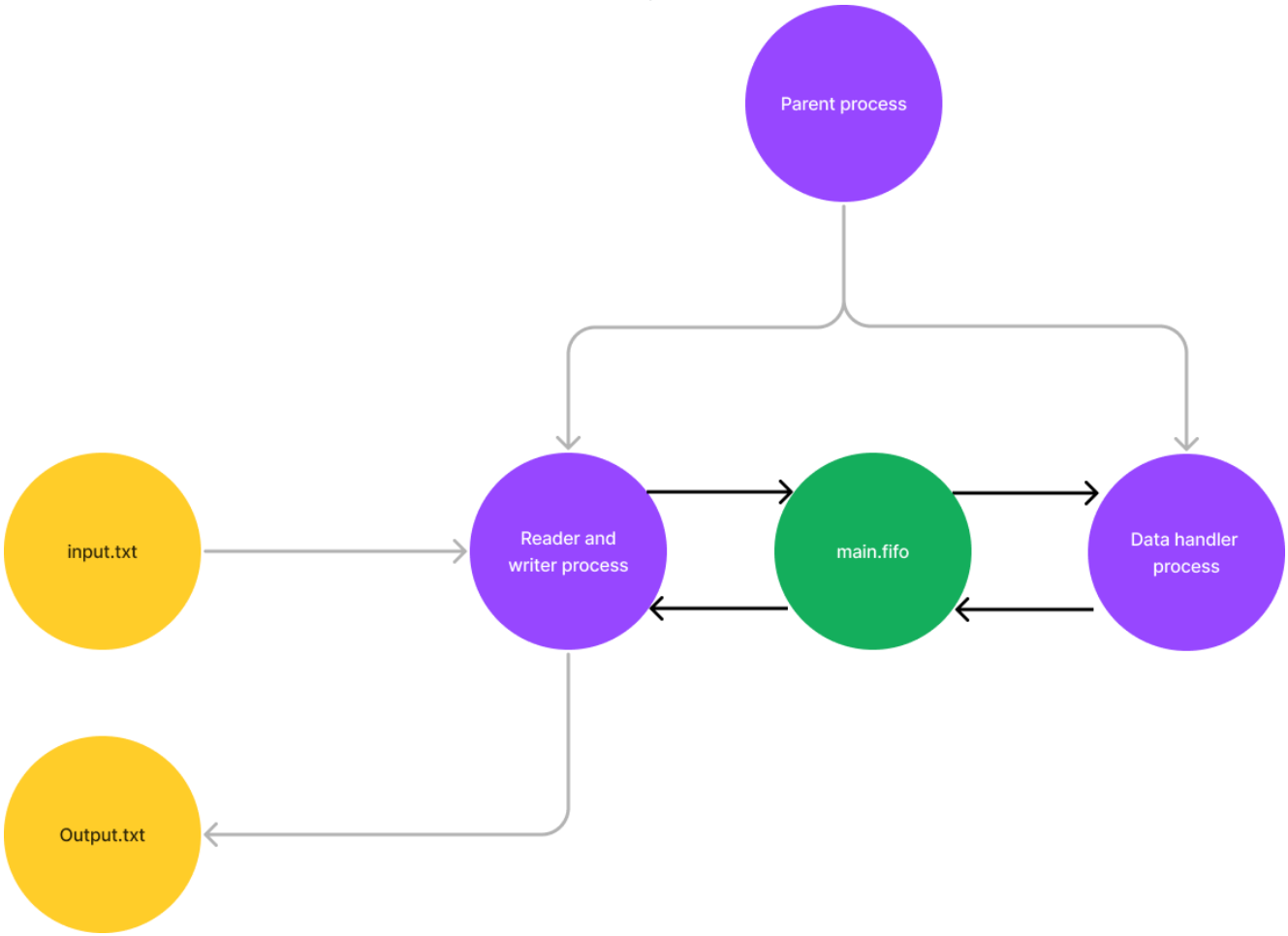
```
$ gcc main.c -o main -lpthread
$ ./main ../tests/input1.txt ../tests/output1.txt main.fifo
```

```
reader started
Reader exit
process started
Process exit
writer started
Writer exit

Parent: All children have exited.
```

На оценку 7

Общая схема(каналы были созданы с помощью функции mknod):



Для синхронизации также использовались семафоры.

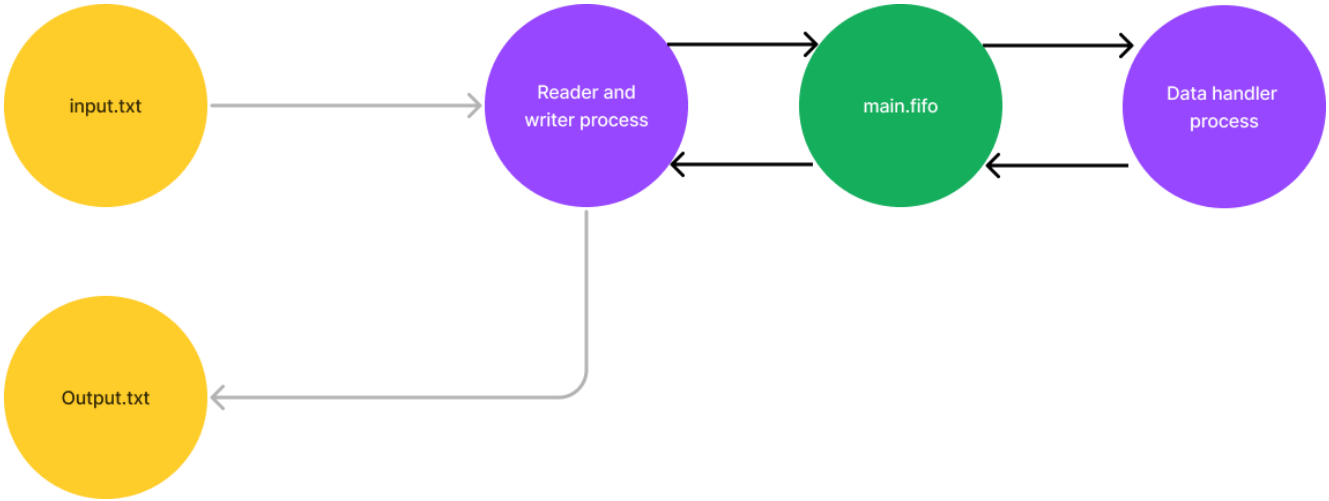
Запуск программы:

```
$ gcc main.c -o main -lpthread
$ ./main ../tests/input1.txt ../tests/output1.txt main.fifo
reader started
Reader exit
process started
Process exit
writer started
Writer exit

Parent: All children have exited.
```

На оценку 8

Общая схема(каналы были созданы с помощью функции mknod):



Первым запускается файл fileWorker.c. После запуска, он засыпает на 5 секунд чтобы было время запустить второй файл(dataProcess.c). Тут не нужна синхронизация, так как канал не может читать из пустого, поэтому ждет данных: Запуск программы (В разных терминалах):

```
# Запуск первого файла
$ gcc fileWorker.c -o fileWorker
$ ./fileWorker ../tests/input2.txt ../tests/output2.txt main.fifo
5 секунд для запуска файла dataProcess
reader started
Reader exit
writer started
Writer exit
```

```
# Запуск второго файла
$ gcc dataProcess.c -o dataProcess
$ ./dataProcess main.fifo
process started
Process exit
```

Как видим, все сработало без ошибок и тесты по файлам отработали корректно.