Национальный исследовательский университет «Высшая школа экономики»
Факультет компьютерных наук
Программная инженерия

Операционные системы

Отчёт по индивидуальному домашнему заданию№1
Вариант 25

Работу выполнила:
М. В. Царахова
Группа: БПИ-213
Преподаватель:
А. И. Легалов

Москва
2023

Содержание

Постановка задачи

Разработать программу, которая определяет в ASCII-строке частоту встречаемости различных идентификаторов, являющихся словами, состоящими из букв и цифр, начинающихся с буквы. Разделителями являются все другие символы. Для тестирования можно использовать программы, написанные на различных языках программирования.

Программы должны быть написаны на языке программирования C и выполняться в среде ОС Linux

4 балла

Файлы ввода и вывода передаются через агрументы командной строки (argc, argv), если аргументов недостаточно выдаётся ошибка.

Мы создаём text_buf для записи туда текста (размер буфера 20000), count_num для записи туда результата подсчета - по индексу 0 количество слов которые просят подсчитать, по индексу 1 количество всех слов.

У нас есть 2 pipe - для передачи информации между потоками. Также у нас есть три потока. is_analyzer - процесс являющийся ребенком главного процесса, для того чтобы делать подсчёт. is_reader - процесс являющийся ребенком anlyzer-a, чтобы читать из файла. И главный процесс используется для того чтобы читать. Данная связь нужна чтобы процессы закрывались в нужном порядке.

В процессе is_reader мы читаем из файла и записываем данные в read_pipe[1]. В процессе is_analyzer мы читаем из read_pipe[0] и записываем данные в write_pipe[1]. И в главном процессе мы читаем из write_pipe[0] и записываем в файл результат.

Используем пайпы мы вот так:

```
read(read_fd, text_buf, sizeof(text_buf));
write(read_pipe[1], text_buf, sizeof(text_buf));
```

В файлике count.h расположены вспомогательные функции для вычисления количества правильных и всех слов. Функция void count(char *str, int *count_num) как раз проходится и записывает результаты в count_num.

Все тесты расположены в папке tests, результаты тестов расположены в той же папке что и код к задаче на конкретный балл.

5 баллов

В дополнение к требованиям на предыдущую оценку необходимо разработать программу, в которой взаимодействие между тремя дочерними процессами осуществляется через именованные каналы.

Мы объявляем пайпы с помощью

```
#define readpipe "/tmp/readpipe"
#define writepipe "/tmp/writepipe"
```

Инициализируем пайпы:

```
mknod(readpipe, S_IFIFO | 0666, 0);
mknod(writepipe, S_IFIFO | 0666, 0);
```

В этом варианте связь процессов такая же как и в прошлый раз. Используем пайпы вот так

```
int read_pipe = open(readpipe, O_WRONLY);
write(read_pipe, text_buf, sizeof(text_buf));
```

И необходимо их закрывать

```
close(read_pipe);
```

Также в конце при записи результата мы делаем unlink, чтобы закрыть каналы:

```
unlink(readpipe);
unlink(writepipe);
```

В данном случае мы использовали именованные каналы.

6 баллов

В дополнение к требованиям на предыдущую оценку разработать программу, которая осуществляет взаимодействие между двумя дочерними процессами с использованием неименованных каналов

Здесь как и в программе на 4 балла используются неименованные каналы - read_pipe[2], write_pipe[2]. У нас есть два процесса - основной и его дочерний процесс child_pid. Основной процесс сначала выполняет чтение и записывает буфер в read_pipe[1]. Далее он ждёт пока его дочерний процесс закончит процесс подсчёта. А после этого основной процесс выполняет запись в файл.

Мы используем неименованные каналы - read_pipe[2], write_pipe[2]. В процессе ребенка мы выполняем чтение в read_pipe[0] и запись в write_pipe[0].

```
read(read_pipe[0], text_buf, sizeof(text_buf));

count(text_buf, count_num);

write(write_pipe[1], count_num, sizeof(count_num));
```

В основном процессе, в моменте чтения мы выполняем чтение из файла и запись в пайп - read(read_fd), write(read_pipe[1]) :

```
read(read_fd, text_buf, sizeof(text_buf));

write(read_pipe[1], text_buf, sizeof(text_buf));
```

В основном процессе, в моменте записи мы выполняем чтение из пайпа и запись в файл - read(write_pipe[0], write(write_fd)

```
read(write_pipe[0], count_num, sizeof(count_num));

write(write_fd, output_buf, strlen(output_buf));
```

7 баллов

В дополнение к требованиям на предыдущую оценку разработать программу, которая осуществляет взаимодействие между двумя дочерними процессами с использованием именованных каналов

Здесь как и в программе на 5 баллов используются именованные каналы - readpipe, writepipe. У нас есть два процесса - основной и его дочерний процесс child_pid, процессы такие же как и в оценке на 6.

Основной процесс сначала выполняет чтение и записывает буфер в read_pipe. Далее он ждёт пока его дочерний процесс закончит процесс подсчёта. А после этого основной процесс выполняет запись в файл.