**Орловский государственный университет имени И. С. Тургенева**

Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Операционные системы»

Отчет к лабораторной работе № 4-5

«Управление процессами  
Межпроцессное взаимодействие через канал pipe»

Выполнила:

Студентка группы 21ПГ

Банных Мария Алексеевна

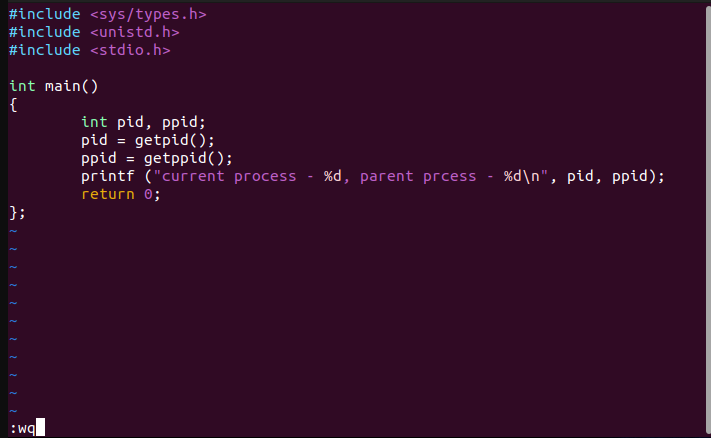
Принял:

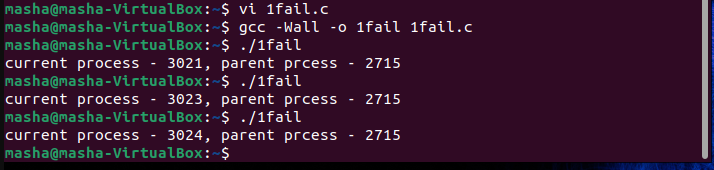
Доцент кафедры   
информационных систем   
и цифровых технологий  
Загородних Николай Анатольевич  
  
И.о. зав. кафедрой   
информационных систем   
и цифровых технологий  
Рыженков Денис Викторович

Орёл 2022 год

**Лабораторная работа №4**

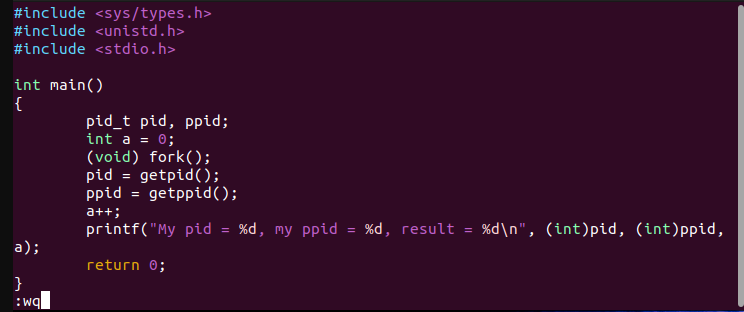
**Задание 1:** В качестве примера использования системных вызовов *getpid()* и *getppid()* самостоятельно напишите программу, печатающую значения *PID* и *PPID* для текущего процесса. Запустите ее несколько раз подряд. Посмотрите, как меняется идентификатор текущего процесса. Объясните наблюдаемые изменения.

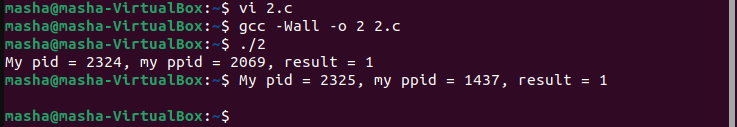




PID меняется каждый раз, так как каждый новый процесс (и каждый новый экземпляр программы с тем же именем) имеет новый идентификатор процесса. PPID не меняется, так как родительский процесс для запущенной программы один – текущий терминал.

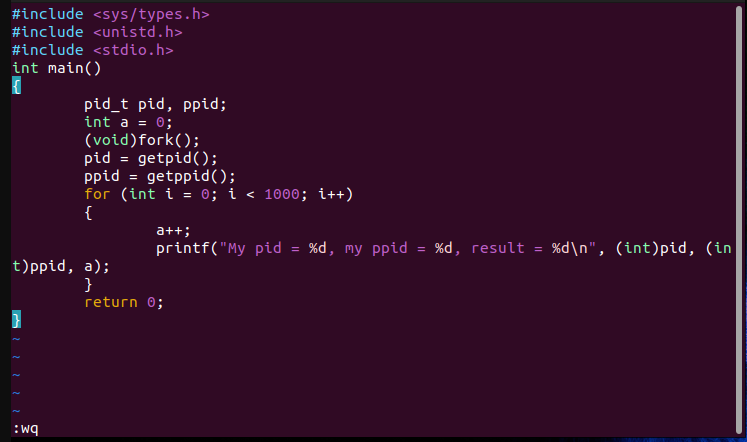
**Задание 2:** Наберите следующую программу, откомпилируйте ее и запустите на исполнение. Проанализируйте полученный результат.

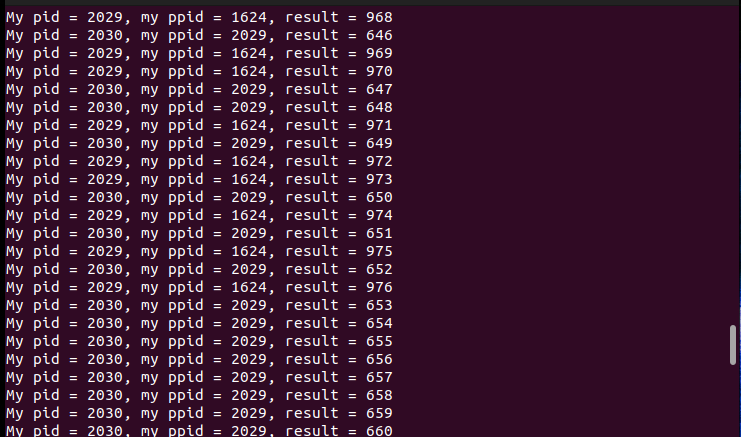




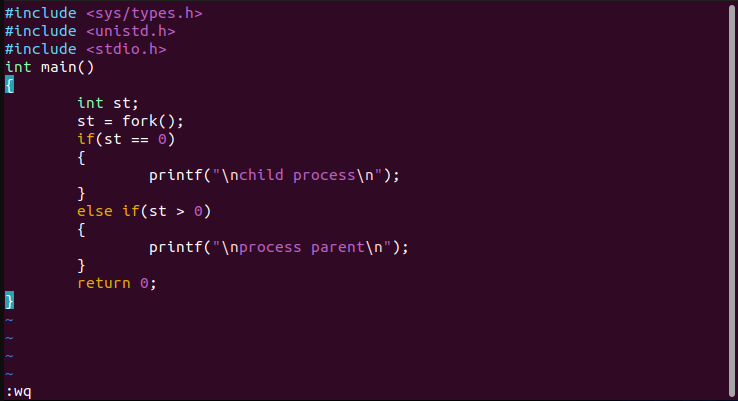
Родительский процесс завершается раньше, чем процесс-потомок успевает получить PPID, поэтому у него появляется «приёмный»   
родитель-специальный процесс.

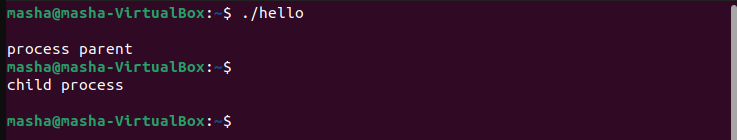
**Задание 3:** Измените программу так, чтобы увеличение значения переменной *а* и вывод на экран монитора выполнялось в цикле. Количество повторений выберите такое, чтобы за один квант времени, выделенный процессу, программа выполнила только часть цикла.



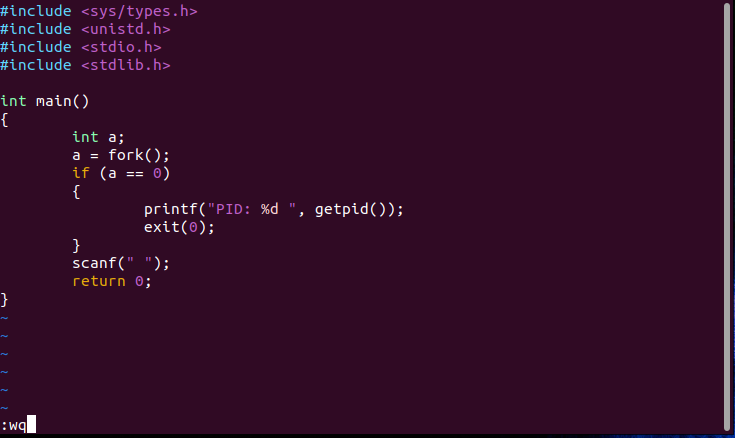


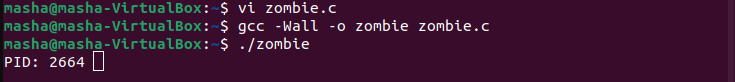
**Задание 4:** Измените программу так, чтобы после возвращения из системного вызова *fork()* процессы могли определить, кто из них является ребенком, а кто родителем. Сделайтетак, чтобы родитель и ребенок совершали разные действия (какие – не важно)

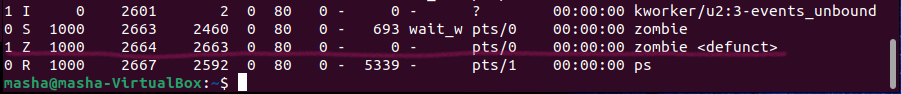
****

****

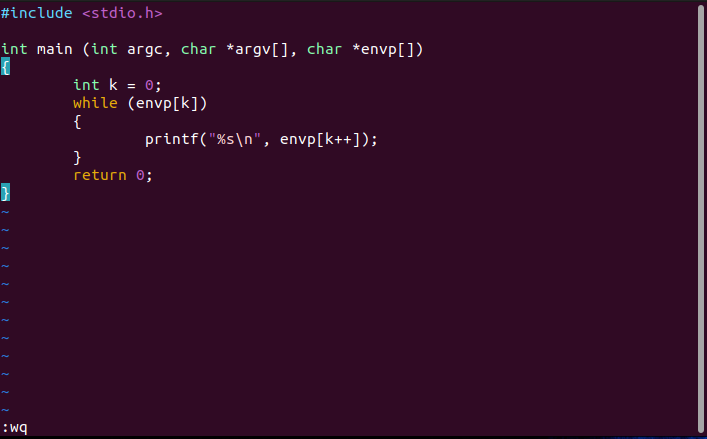
**Задание 5:** Напишите программу, создающую процесс-зомби. Выведите на экран его *pid* и посмотрите в другом окне терминала, как обозначается данный процесс при вызове команды просмотра состояния процессов (команда *ps).*

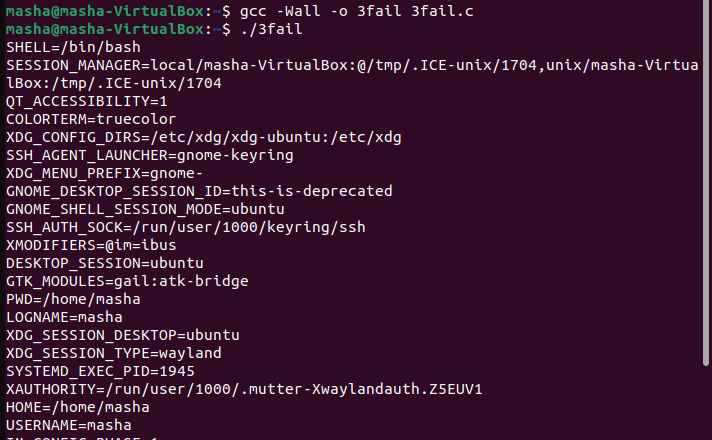
**



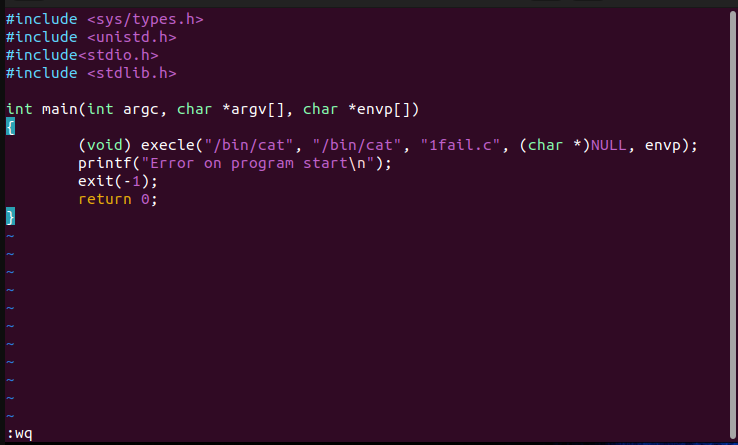


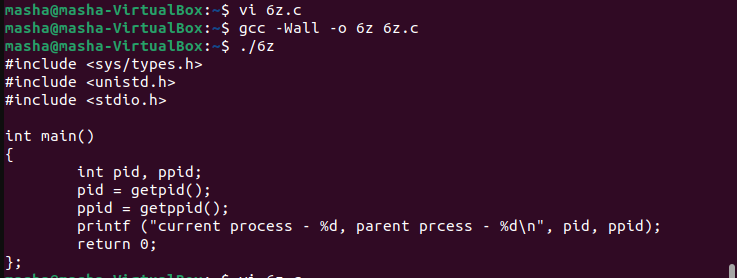
**Задание 6:** Напишите программу, которая распечатывает на экране значения переменных окружения.

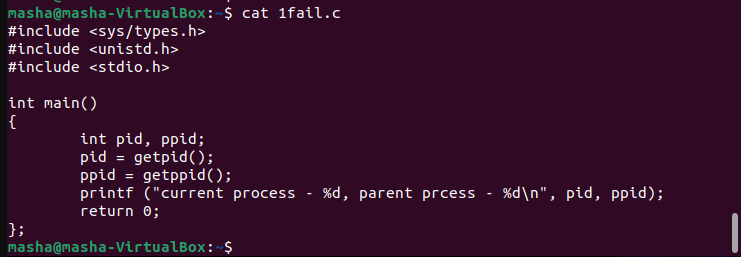




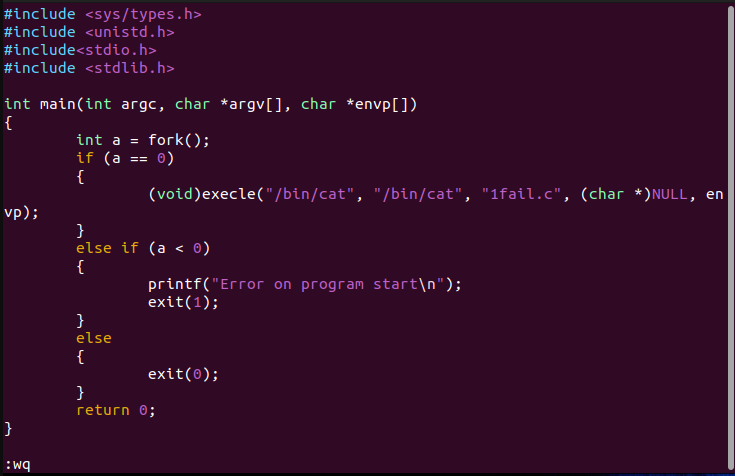
**Задание 7:** Рассмотрите пример использования системного вызова *exec().* Проанализируйте результат.

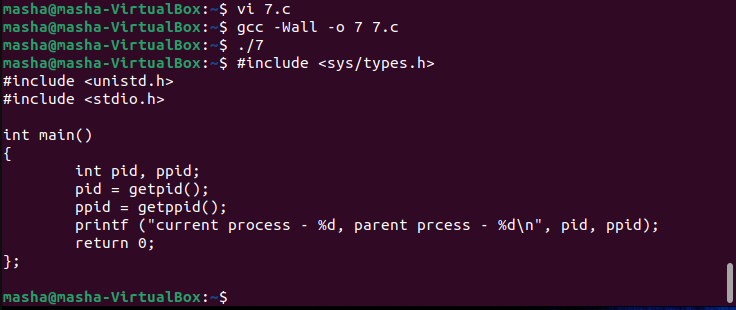






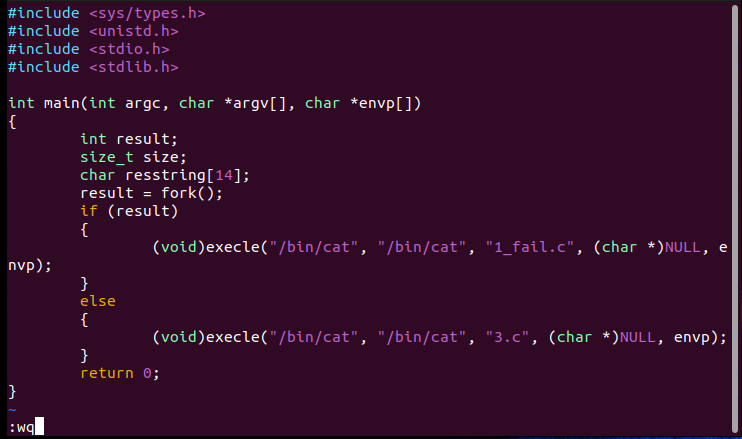
**Задание 8:** модифицируйте программу, созданную при выполнении задания, использующую вызов *fork()* с разным поведением процессов ребенка и родителя, так, чтобы порожденный процесс запускал на исполнение новую (любую) программу.

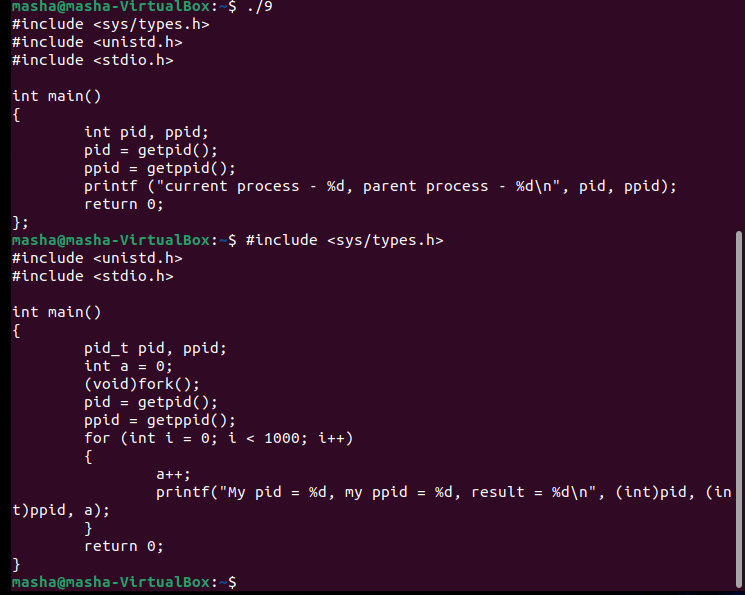




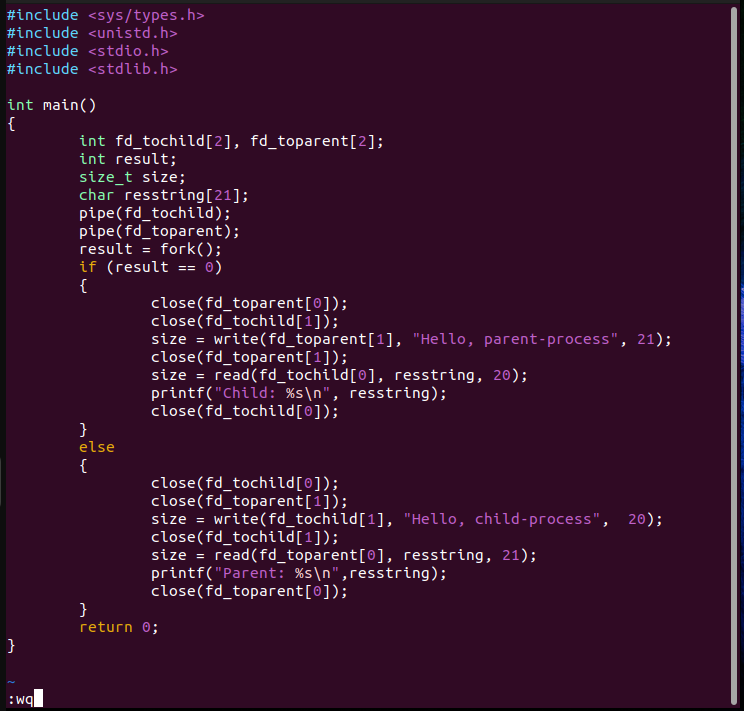
**Лабораторная работа №5**

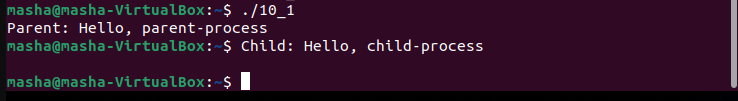
**Задание 9:** модифицируйте этот пример для связи между собой двух родственных процессов, исполняющих разные программы.



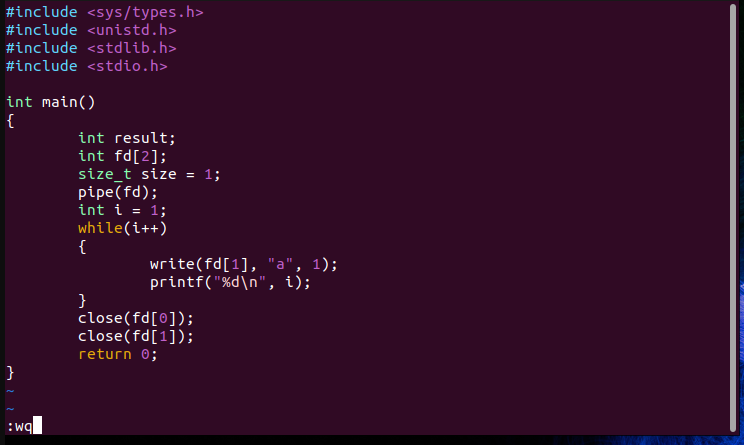


**Задание 10:** измените предыдущий пример, причем для двухсторонней связи используйте два канала: один для передачи от родителя ребенку, второй - от ребенка родителю.



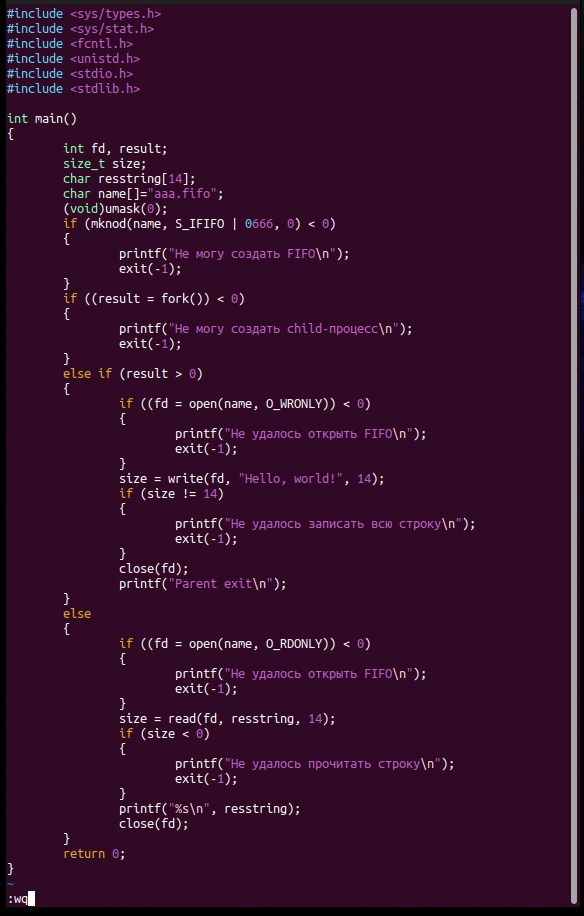


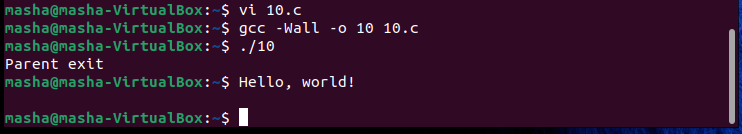
**Задание 11:** Напишите программу для определения в используемой операционной системе размера канала.



****

**Задание 12:** Для иллюстрации взаимодействия родственных процессов через FIFO рассмотрите приведенную ниже программу (добавьте проверку на успешность завершения операций).

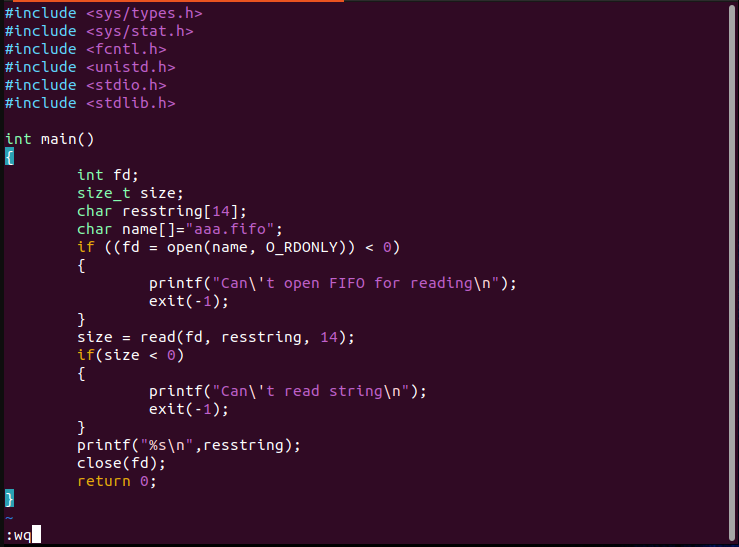




**Задание 13:** напишите на базе предыдущего примера две программы, одна из которых пишет информацию в *FIFO,* а вторая - читает из него, так чтобы между ними не было ярко выраженных родственных связей (то есть, чтобы ни одна из них не была потомком другой).



(содержимое файла 11.c)



(содержимое файла 12.с)

