Приложение 1. Отчет по лабораторной работе

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

  Факультет компьютерных наук

Департамент программной инженерии

Отчёт по лабораторной работе

**Лабораторная работа 10**

**Организация файловой системы в UNIX**

по курсу «Операционные системы»

Выполнил:

Принял

Фомин Сергей Сергеевич

Оценка:

Москва 2020г.

Оглавление

[Задание 1 3](#_Toc40988657)

[Задание 2 4](#_Toc40988658)

[Задание 3 5](#_Toc40988659)

[Задание 4 7](#_Toc40988660)

[Задание 5 9](#_Toc40988661)

**Для каждого задания указывается:**

1. Формулировка задания.
2. Исходный текст программы.
3. Формат вызова программы.
4. Результат работы программы.
5. Объяснения полученных результатов.

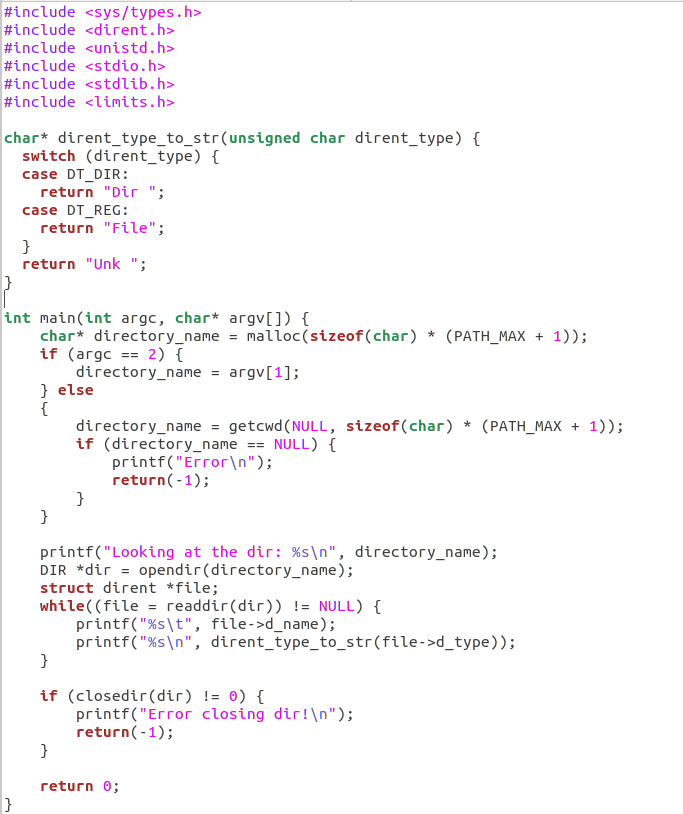
# Задание 1

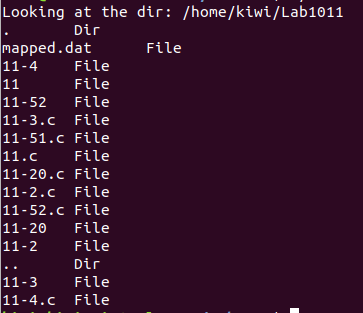
Напишите, откомпилируйте и прогоните программу, распечатывающую список файлов, входящих в каталог, с указанием их типов. Имя каталога задается как параметр командной строки. Если оно отсутствует, то выбирается текущий каталог.

10 баллов - оригинальная реализация

5 баллов – известная реализация

Исходный код:

  
Формат вызова программы:  
  
Результат работы:



Объяснение:

Изначально мы получаем директорию, в которой находимся и проверяем её на корректность, после чего объявляем структуру директории и при помощи команды readdir считываем содержимое директории и выводим по каждому файлу, узнавая его тип, который хранится в структуре типа dirent.

# Задание 2

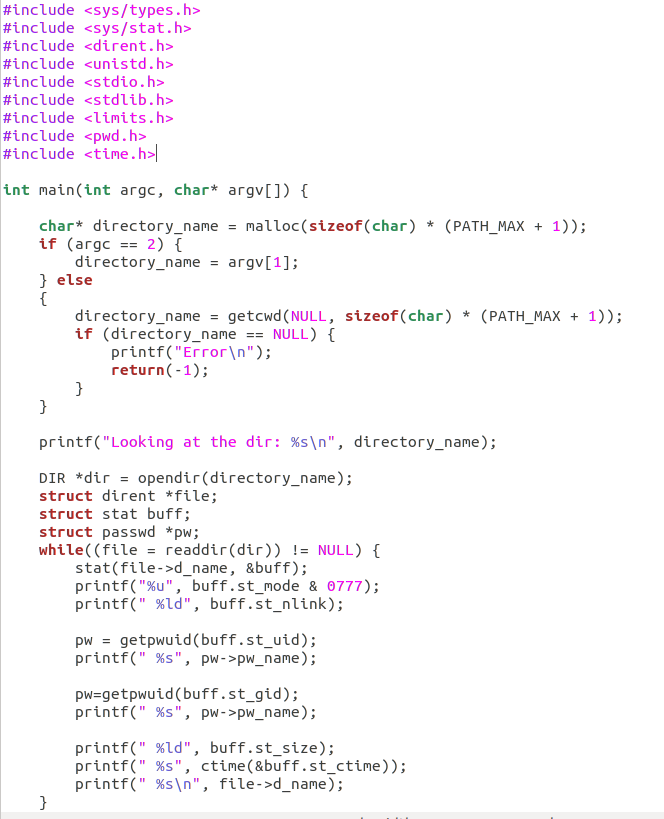
**Задача повышенной сложности:** напишите программу, распечатывающую содержимое заданного каталога в формате, аналогичном формату выдачи команды ls -al.

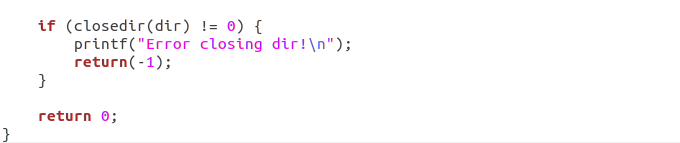
Для этого понадобится изучить в UNIX Manual функцию ctime(3) и системные вызовы time(2), readlink(2). Цифры после имен функций и системных вызовов – это номера соответствующих разделов для UNIX Manual.

15 баллов - оригинальная реализация

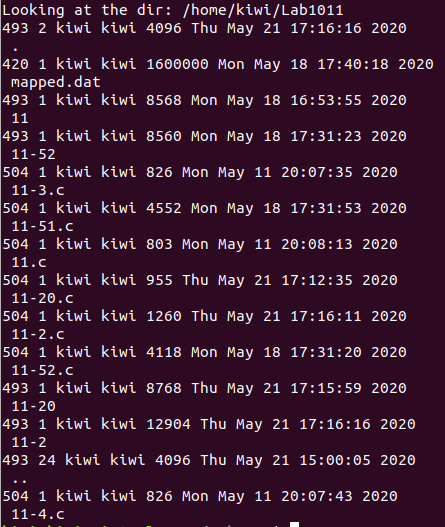
8 баллов – известная реализация

Исходный код:





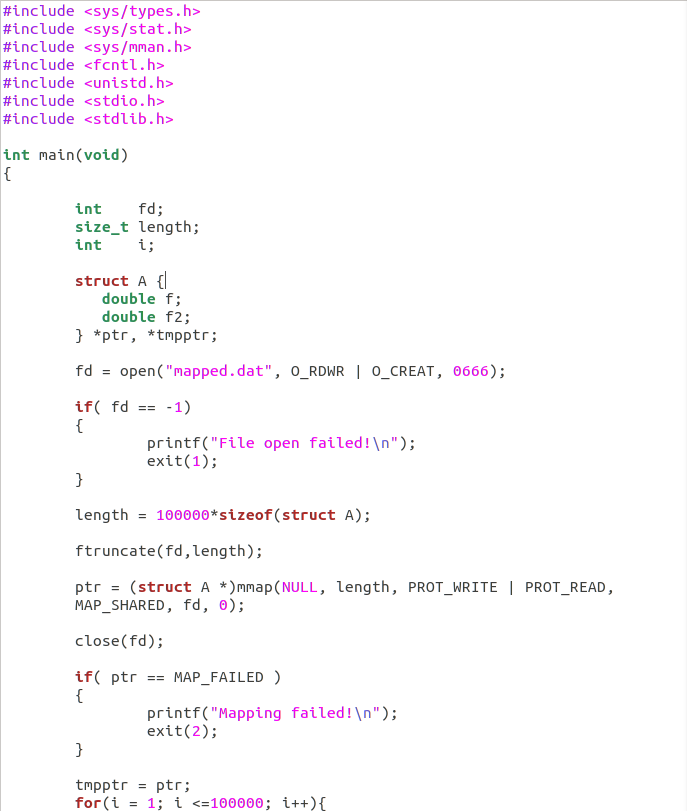
Формат вызова программы:  
  
Результат работы:

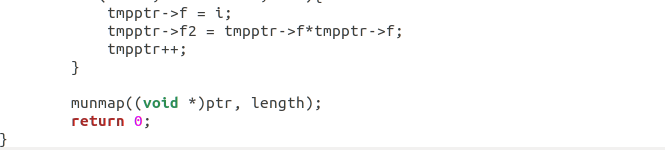
  
Объяснение:

Системные вызовы stat, fstat и lstat служат для получения информации об атрибутах файла. Системный вызов stat читает информацию об атрибутах файла, на имя которого указывает параметр filename, и заполняет ими структуру, расположенную по адресу buf. Заметим, что имя файла должно быть полным, либо должно строиться относительно той директории, которая является текущей для процесса, совершившего вызов. Если имя файла относится к файлу типа «связь», то рекурсивно читается информация об атрибутах файла, на который указывает символическая связь.  
Вызов getpwuid позволяет узнать имя пользователя по его ID. Функция ctime приводит значение поля st\_ctime (время модификации прав доступа файла (в нашем случае время создания файла, т.к. права доступа не были модифицированы с момента создания)) к строковому виду.

# Задание 3

Откомпилируйте и прогоните программу 11-1.с, которая создает файл, отображает его в адресное пространство процесса и заносит в него информацию с помощью обычных операций языка С.  
(5 баллов)  
Исходный код:



  
Формат вызова программы:

  
Результат работы:

  
Объяснение:

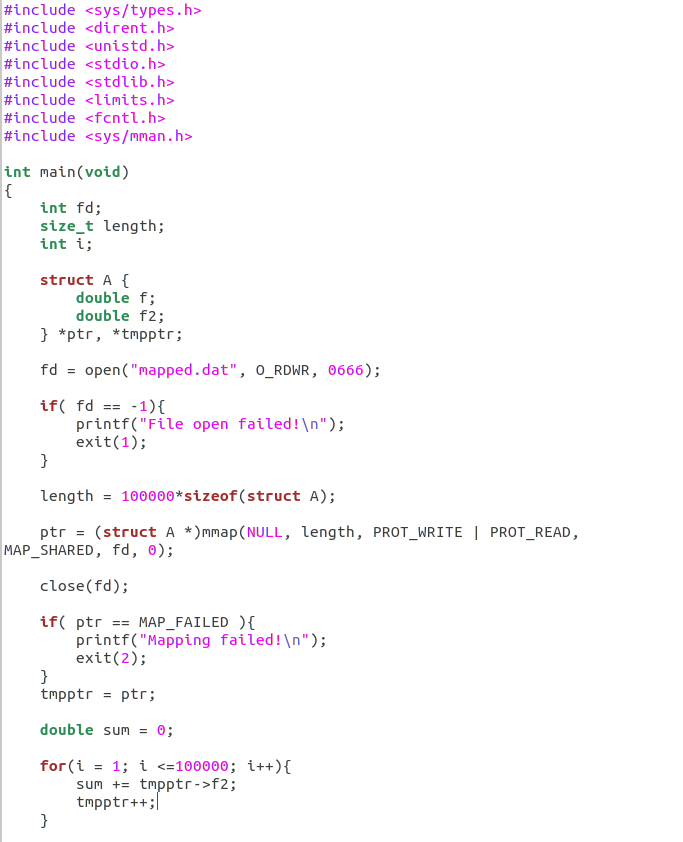
Эта программа создает файл, отображает его в адресное пространство процесса и заносит в него информацию с помощью обычных операций языка С.  
Объявляется файловый дескриптор, после чего объявляется тип структуры A, которым мы забьем файл, и двух указателей на подобный тип. Указатель ptr будет использоваться в качестве начального адреса выделенной области памяти, а указатель tmpptr – для перемещения внутри этой области.  
Открываем файл или сначала создаем его (если такого файла не было). Права доступа к файлу при создании определяем как read and write для всех категорий пользователей (0666). Из-за ошибки в Linux мы будем вынуждены ниже в системном вызове mmap() разрешить в отображении файла и чтение, и запись, хотя реально нам нужна только запись. Поэтому и при открытии файла мы вынуждены задавать O\_RDWR. После чего вычисляем будущую длину файла.  
Вновь созданный файл имеет длину 0. Если мы его отобразим в память с такой длиной, то любая попытка записи в выделенную память приведет к ошибке. Увеличиваем длину файла с помощью вызова ftruncate().  
Отображаем файл в память. Разрешенные операции над отображением указываем как PROT\_WRITE | PROT\_READ по уже названным причинам. Значение флагов ставим в MAP\_SHARED, так как мы хотим сохранить информацию, которую занесем в отображение, на диске. Файл отображаем с его начала (offset = 0) и до конца (length = длине файла).  
После чего закрывает файловый дескриптор, ибо он нам больше не нужен.  
В цикле заполняем образ файла числами от 1 до 100000 и их квадратами. Для перемещения по области памяти используем указатель tmpptr, так как указатель ptr на начало образа файла нам понадобится для прекращения отображения вызовом munmap().  
Прекращаем отображать файл в память, записываем содержимое отображения на диск и освобождаем память при помощи munmap().  
При отображении файла мы вынуждены разрешить в нем и запись, и чтение, хотя реально совершаем только запись. Это сделано для того, чтобы избежать ошибки в операционной системе Linux, связанной с использованием 486-х и 586-х процессоров. Такой список разрешенных операций однозначно требует, чтобы при открытии файла системным вызовом open() файл открывался и на запись, и на чтение. Поскольку информацию мы желаем сохранить на диске, при отображении использовано значение флагов MAP\_SHARED.

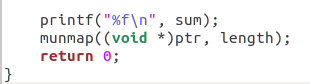
# Задание 4

Модифицируйте предыдущую программу (11-1.с) так, чтобы она отображала файл, записанный этой программой в память и считала сумму квадратов чисел от 1 до 100000, которые уже находятся в этом файле.

(3 балла)

Исходный код:



  
Формат вызова программы:

  
Результат работы:

  
Объяснение:  
В данной программе происходит отображение файла в адресное пространство вычислительной системы, откуда берутся записанные ранее значения (квадраты чисел) и суммируются, после чего указатель переходит на новый участок памяти. После считывания всех файлов, происходит прекращение отображения файла в адресное пространство системы при помощи функции munmap.

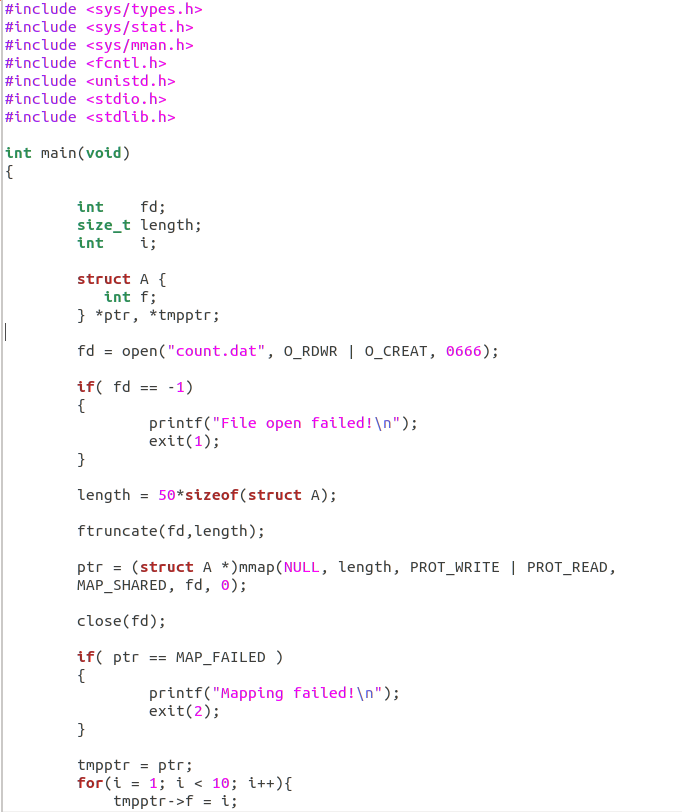
# Задание 5

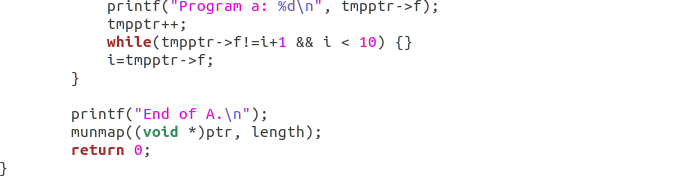
**Задача повышенной сложности:** напишите две программы, использующие memory mapped файл для обмена информацией при одновременной работе, подобно тому, как они могли бы использовать разделяемую память.

15 баллов - оригинальная реализация

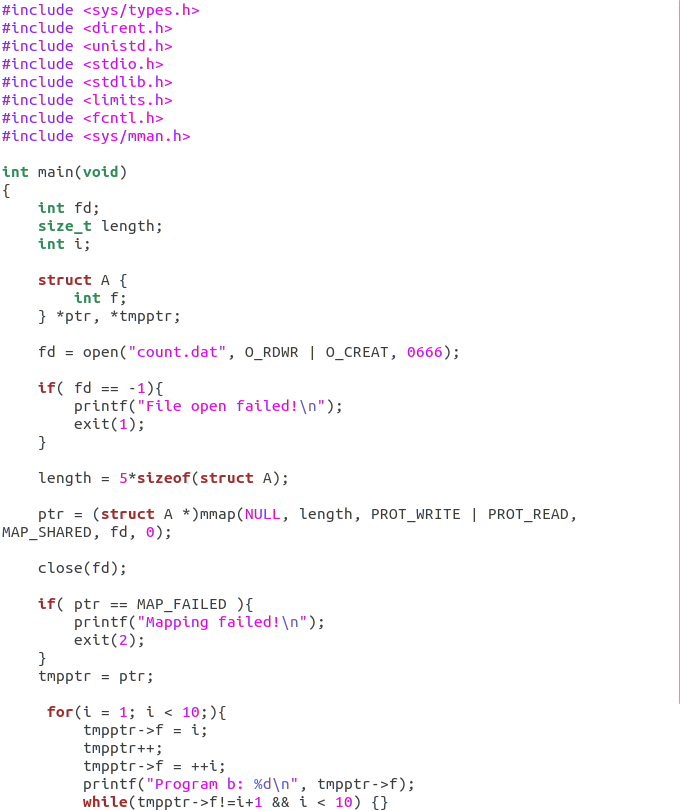
8 баллов – известная реализация

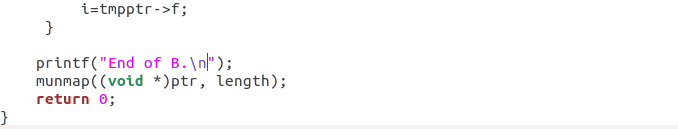
Исходный код:

11-51.с:  




11-52.с:

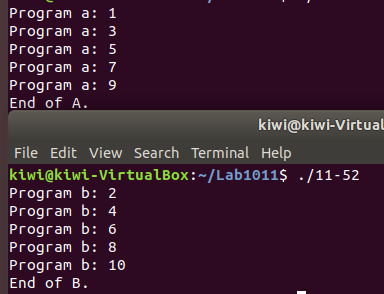




Формат вызова программы:



  
Результат работы:

  
Объяснение:  
При работе первой программы записывается первое значение. Далее пока значение в следующей ячейке не изменится на значение текущей программы +1, программа ожидает (вступает в бесконечный цикл с условием). В этот момент вторая программа смотрит на записанное раннее значение и кладёт в следующую ячейку увеличенное на единицу полученное прежде (которое мы «смотрели») значение и вступает в режим ожидания. Работа программ продолжается до тех пор, пока не будет пройдено 10 шагов (то есть по 5 от каждой программы).  
Программы используют файл одновременно, более того программы зависят друг от друга: если одна из программ не положит в ячейку увеличенное на 1 значение, то вторая не выйдет из бесконечного цикла. В итоге произойдёт зацикливание обеих программ.