Лабораторная работа № 4

Содержание

Сборка программ с помощью системы сборки GNU Autotools....2
Утилита make и Makefile....3
Разработка файловой системы....3
Разработка модуля ядра ОС Linux....4
Задание....6
Критерии оценивания:....6
Задание 1....6
Задание 2....8
Варианты....8
Вариант 0....8
Вариант 1....9
Вариант 2....9
Вариант 3....10
Вариант 4....10
Вариант 5....11
Вариант 6....12
Вариант 7....12
Вариант 8....13
Вариант 9....13
Вариант 10....14
Вариант 11....14
Вариант 12....15
Вариант 13....15
Вариант 14....16
Вариант 15....16
Вариант 16....17
Вариант 17....17
Вариант 18....18
Вариант 19....18
Вариант 20....19
Вариант 21....19
Контрольные вопросы....20

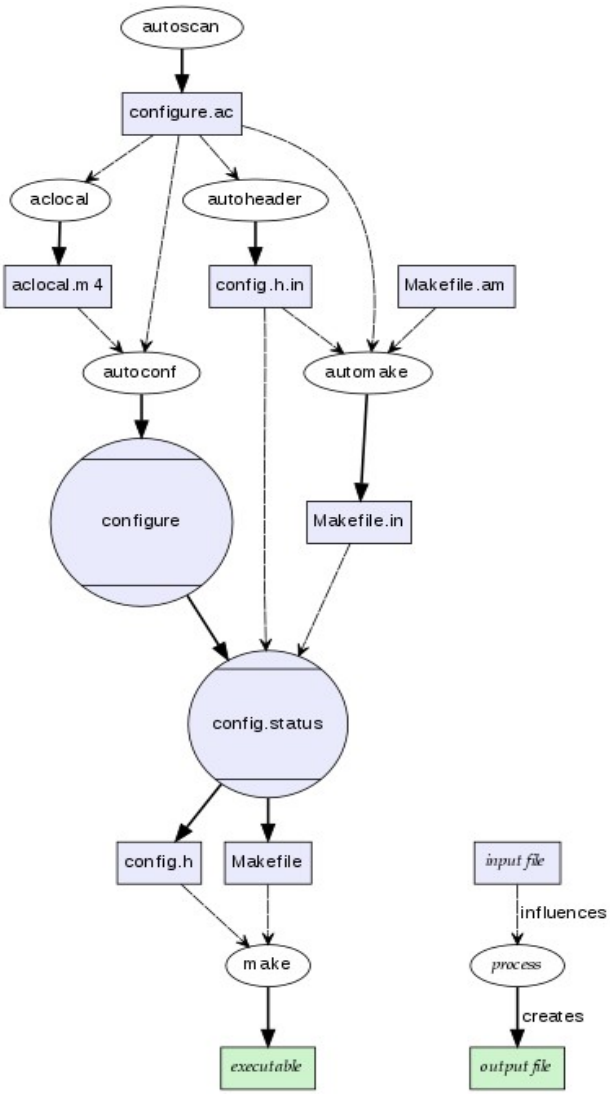
Цель — изучить материалы по разработке файловой системе и по разработке модулей ядра ОС Linux.

Сборка программ с помощью системы сборки GNU Autotools

Autotools, или система сборки GNU,— это набор программных средств, предназначенных для поддержки переносимости исходного кода программ между UNIX-подобными системами.

Перенос кода с одной системы на другую может оказаться непростой задачей. Различные реализации компилятора языка Си могут существенно различаться: некоторые функции языка могут отсутствовать, иметь другое имя или находиться в разных библиотеках. Программист может решить эту задачу, используя макросы и директивы препроцессора, например `#if`, `#ifdef` и прочие. Но в таком случае пользователь, компилирующий программу на своей системе, должен будет определить все эти макросы, что не так просто, поскольку существует множество разных дистрибутивов и вариаций систем. `Autotools` вызываются последовательностью команд `./configure && make && make install` и решают эти проблемы автоматически.

Система сборки `GNU Autotools` является частью `GNU toolchain` и широко используется во многих проектах с открытым исходным кодом. Средства сборки распространяются в соответствии с GNU General Public License с возможностью использования их в коммерческих проектах.



В состав Autotools входят следующие утилиты:

- Autoconf
- Automake
- Libtool
- Gnulib
- Другие средства:
  - make
  - gettext
  - pkg-config
  - gcc
  - binutils

На рисунке выше представлена схема работы autoconf и automake.

Изучить материалы по использованию системы сборки Autotools и автоматической генерации Makefile:

1. http://www.h-wrt.com/ru/mini-how-to/autotoolsSimpleProject
2. https://eax.me/autotools/
3. https://help.ubuntu.ru/wiki/using_gnu_autotools
4. https://opensource.com/article/19/7/introduction-gnu-autotools
5. https://liberatum.ru/page/primer-ispolzovaniya-gnu-autotools
6. https://www.gnu.org/software/automake/faq/autotools-faq.html

Утилита make и Makefile

Утилита `make` автоматически определяет какие части большой программы должны быть перекомпилированы, и выполняет необходимые для этого действия. Наиболее часто **make** используется для компиляции C-программ и содержит особенности, ориентированные именно на такие задачи, но можно использовать **make** с любым языком программирования. Более того, применение утилиты **make** не ограничивается программами. Можно использовать ее для описания любой задачи, где некоторые файлы должны автоматически порождаться из других всегда, когда те изменяются.

Ознакомьтесь с рекомендациями по разработке файлов Makefile и применению утилиты make:

- http://parallel.uran.ru/book/export/html/16
- http://linux.yaroslavl.ru/docs/prog/gnu_make_3-79_russian_manual.html
- https://www.gnu.org/software/make/manual/html_node/index.html#SEC_Contents

Разработка файловой системы

https://github.com/libfuse/libfuse

Разработка собственной файловой системы с помощью FUSE — http://www.ibm.com/developerworks/ru/library/l-fuse/

Учебник по FUSE — http://wiki.linuxformat.ru/wiki/LXF76:%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA_Fuse

Пример написания файловой системы с помощью FUSE на языке Go - https://4gophers.ru/articles/pishem-failovuyu-ssitemu-na-go-i-fuse/

Пример создания файловой системы с помощью FUSE - http://rus-linux.net/MyLDP/algol/code_ninja_make_a_filesystem_with_fuse.html

Разработка модуля ядра ОС Linux

Рассмотрим простейший драйвер "hello world" имеет следующие вид:

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("Dual BSD/GPL");
static int hello_init(void)
{
    printk(KERN_ALERT "Hello, world\n");
    return 0;
}
static void hello_exit(void)
{
    printk(KERN_ALERT "Goodbye, cruel world\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

В этой программе макросы `modulde_init` и `module_exit` определяют, какие функции являются точкой входа и выхода из модуля.

В модуле ядра нет доступа к стандартным потокам ввода-вывода, поэтому печать может осуществляться только в системный лог-файл с помощью функции `printk`. Результат этого вывода можно увидеть с помощью команды `dmesg`.

Кроме того, в коде модуля нет доступа к функциям стандартной библиотеки C, такими как, например, printf. Вместо них в ядре реализована собственная "стандартная" библиотека — Linux Kernel API (https://www.kernel.org/doc/htmldocs/kernel-api).

Для сборки модуля "hello world" необходимо создать следующий `Makefile`:

```
obj-m := hello.o
KDIR := /lib/modules/<версия ядра>/build
PWD := $(shell pwd)
default: $(MAKE) -C $(KDIR) M=$(PWD) modules
```

Текущую версию ядра ОС можно узнать с помощью команды `uname -r`:

```
$ uname -r
```

В случае отсутствия необходимых заголовочных файлов их можно установить с помощью менеджера пакетов ОС. Например, в Debian-подобных Linux системах это можно сделать следующим образом:

```
$ sudo apt-get install build-essential linux-headers-
$(uname -r)
```

Для загрузки собранного модуля используется команда `insmod`. Для выгрузки — `rmmod`.

```
$ sudo insmod hello.ko
$ dmesg | tail -1
[ 8394.731865] Hello, world
$ sudo rmmod hello.ko
$ dmesg | tail -1
[ 8707.989819] Goodbye, cruel world
```

Изучить примеры из учебных материалов, опубликованных в курсе:

- Разработка ядра Linux
- Программирование в Linux

А также дополнительные материалы, опубликованные ниже:

How to Write Your Own Linux Kernel Module with a Simple Example — http://www.thegeekstuff.com/2013/07/write-linux-kernel-module/

Linux Device Drivers — http://lwn.net/Kernel/LDD3/

The Linux Kernel Module Programming Guide — http://www.tldp.org/LDP/lkmpg/2.6/html/lkmpg.html

Linux kernel and driver development training — http://free-electrons.com/doc/training/linux-kernel/linux-kernel-labs.pdf

Как начать писать под ядро Linux (видео) — https://www.youtube.com/watch?v=m5Bgh5qyTI4

Задание

Изучить теоретическую часть лабораторной работы, включая материалы по использованию системы автоматической сборки Autools и разработке файлов Makefile.

Критерии оценивания:

Оценка 4-5

Выполнено задание 1 без реализации дополнительной функции. Структура программы соответствует КИС, содержит Makefile для сборки, созданный вручную. В Makefile продемонстрировать использование переменных, автоматических переменных, шаблонных имен и, при необходимости, поиск пререквизитов по каталогам, абстрактные цели. Представлен отчет, исходный код проекта в git-репозитории. Отчет, файлы протоколов команд и меток времени без ошибок. Лабораторная работа сдана с задержкой в 1-2 недели.

Оценка 6-7

Выполнено задание 1 с реализацией дополнительной функции. Структура программы соответствует КИС и для сборки применяется система сборки Autotools с автоматической генерацией Makefile, а так же должен быть создан пакет проекта с помощью команды `make distcheck`.

Для задания 1 настроить сборку с помощью сервиса непрерывной интеграции Travis-CI.
Представлен отчет, ответы на контрольные вопросы, исходные коды скриптов в git-репозитории. Отчет, исходный код может содержать незначительные ошибки. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 8-9

Выполнены задания 1-2 на отличном уровне. Структура программы соответствует КИС и для сборки применяется система сборки Autotools с автоматической генерацией Makefile, а так же должен быть создан пакет проекта с помощью команды `make distcheck`.
Для задания 1 настроить сборку с помощью сервиса непрерывной интеграции Travis-CI.
Представлен отчет, ответы на контрольные вопросы, исходные коды скриптов в git-репозитории. Отчет, исходный код не содержат ошибок. Лабораторная работа сдана в срок.

Задание 1

Изучите пример реализации файловой системы FUSE на языке C https://www.maastaar.net/fuse/linux/filesystem/c/2016/05/21/writing-a-simple-filesystem-using-fuse/ .

Напишите на языке C программу с помощью библиотеки FUSE, которая подключает виртуальную файловую систему, дерево директорий которой (полученное с помощью команды `tree`) задано ниже.

Требования к программе:

- структура программы должна соответствовать модели КИС (https://www.opennet.ru/docs/RUS/zlp/002.html).
- Сборка выполняться с помощью утилиты make. При написании Makefile продемонстрировать использование шаблонов, переменных, пререквизитов **(для оценки 4-5)**.
- Для сборки использовать систему сборки Autotools с автоматической генерацией Makefile **(для оценки 6-9)**.
- Сформировать пакет с открытыми исходными кодами в формате tgz (tar.gz) **(для оценки 6-9)**.
- Продемонстрировать автоматическую сборку с Travis-CI и результаты выполнения приложения **(для оценки 6-9)**.

Файловая система содержит 4 директории: `foo, bar, baz и bin`,— а также 4 файла, из которых 3 — текстовые файлы: `example, readme.txt, test`,— и 1 бинарный. Содержимое бинарного файла должно быть взято из соответствующей стандартной системной утилиты, название которой соответствует названию файла: `ls, grep, pwd,… в зависимости от задания.`

- Содержимое остальных файлов:
  readme.txt: `Student <имя и фамилия>, <номер зачетки>`
  test.txt: <`Любой текст на ваш выбор с количеством строк равным последним двум цифрам номера зачетки`>
- *example*: `Hello world! Student <имя и фамилия>, group <номер группы>, task <вариант>.`

Файловая система должна монтироваться в папку `/mnt/fuse/` , после чего должна быть возможность осуществить листинг ее директорий и просмотр содержимого виртуальных файлов. При обращении к файловой системе должны проверяться права доступа (маска прав указана в дереве директорий через слеш после имени файла). Владельцем всех файлов должен быть текущий пользователь, который выполняет монтирование системы.

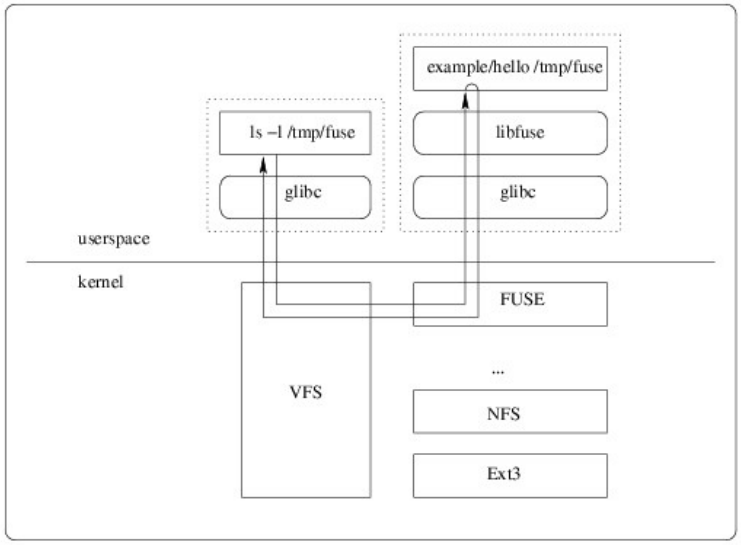


Figure 1: Схема работы FUSE

Задание 2

Написать на языке C модуль ОС Linux, который реализует то же задание, что и в задании 1 без использования FUSE согласно своему варианту.

Требования к программе:

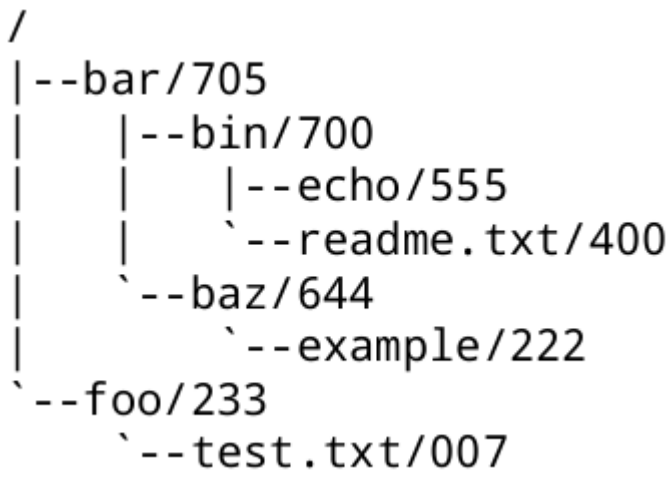
- структура программы должна соответствовать модели КИС (https://www.opennet.ru/docs/RUS/zlp/002.html).
- Для сборки использовать систему сборки Autotools с автоматической генерацией Makefile.
- Сформировать пакет с открытыми исходными кодами в формате tgz (tar.gz).

Варианты

Номер варианта индивидуального задания **K** равен числу букв **N1** вашей фамилии, умноженному на число букв **N2** Вашего имени (по паспорту), умноженному на число букв **N3** вашего отчества по модулю 22: k= (N1*N2*N3) mod 22.

Вариант 0.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция mkdir.

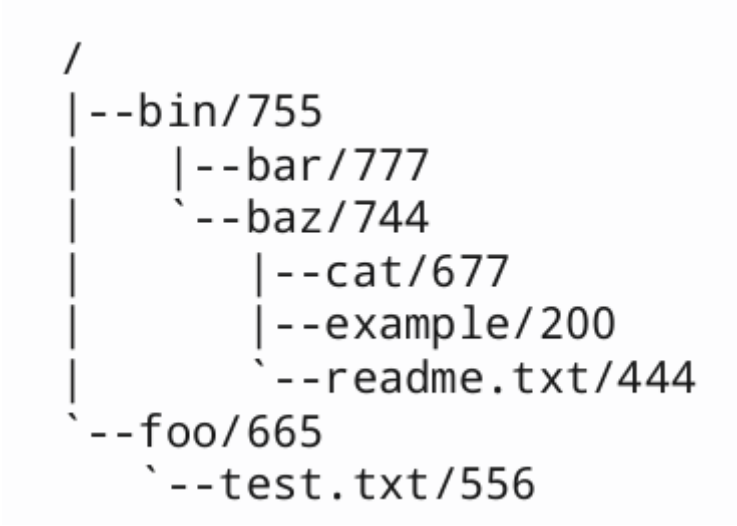


Задание 1. Вариант 0

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 1.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rmdir.



Задание 1. Вариант 1

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 2.

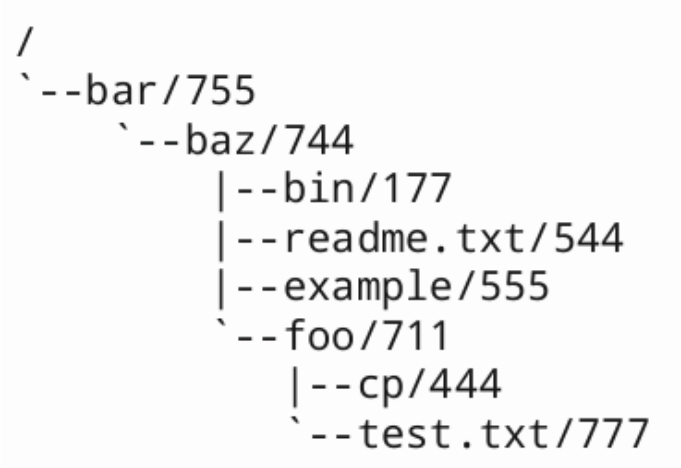
1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция symlink.

Задание 1. Вариант 2

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 3.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rename.

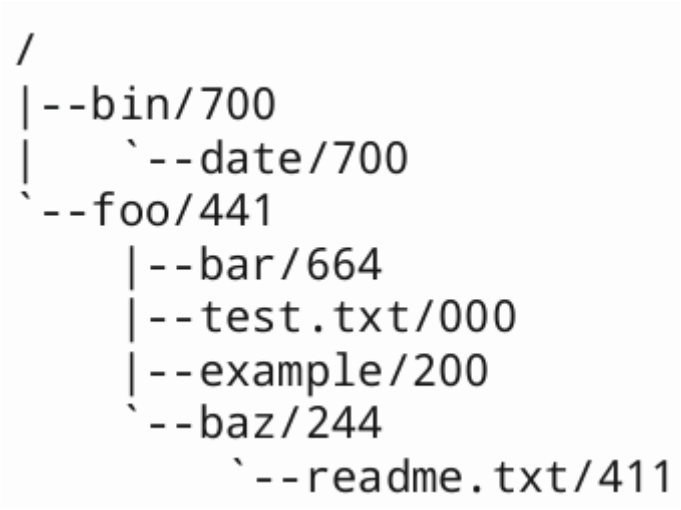


Задание 1. Вариант 3

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 4.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция write.

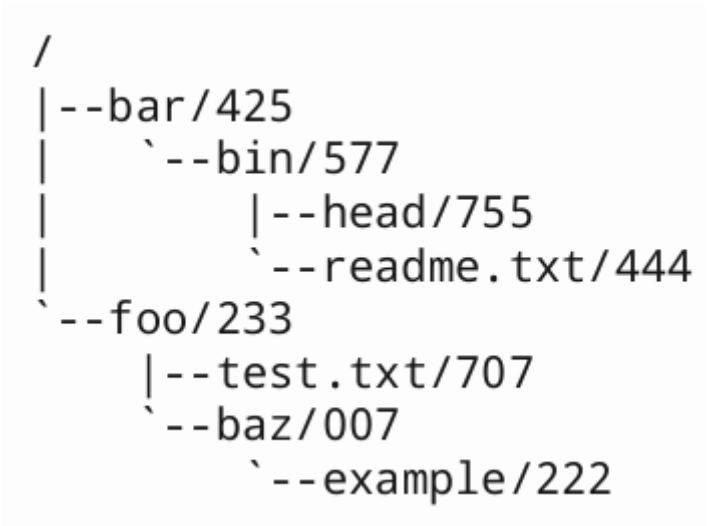


Задание 1. Вариант 4

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 5.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rmdir.



Задание 1. Вариант 5

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 6.

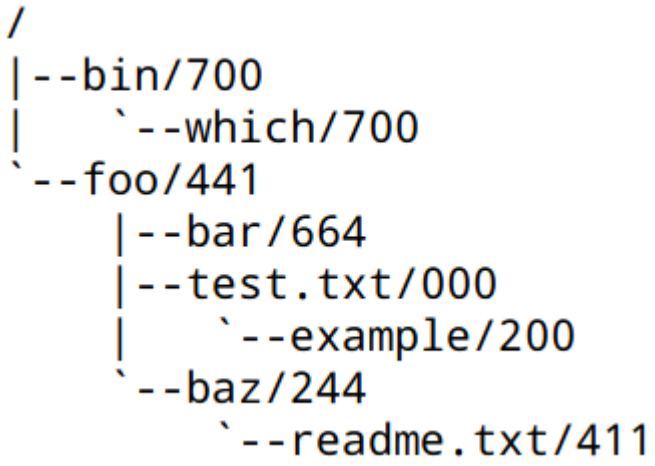
1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chown.

Задание 1. Вариант 6.

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 7.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rename.

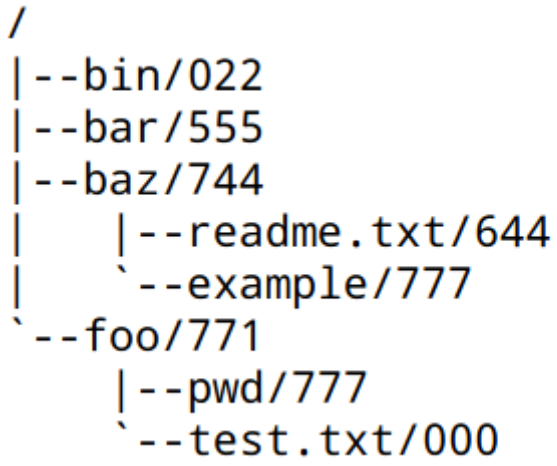


Задание 1. Вариант 7

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 8.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chown.

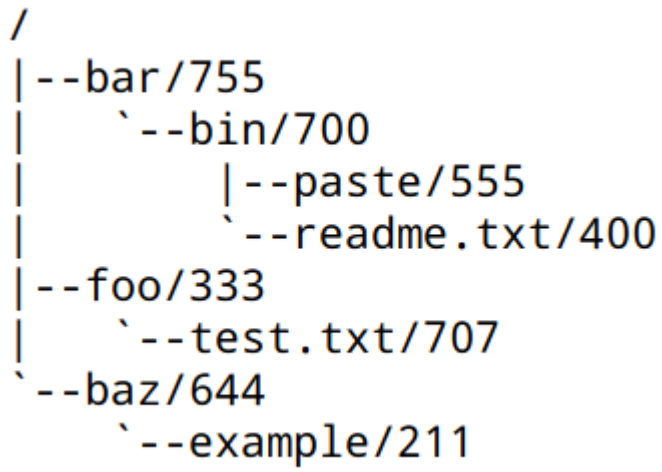


Задание 1. Вариант 8

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 9.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chmod.

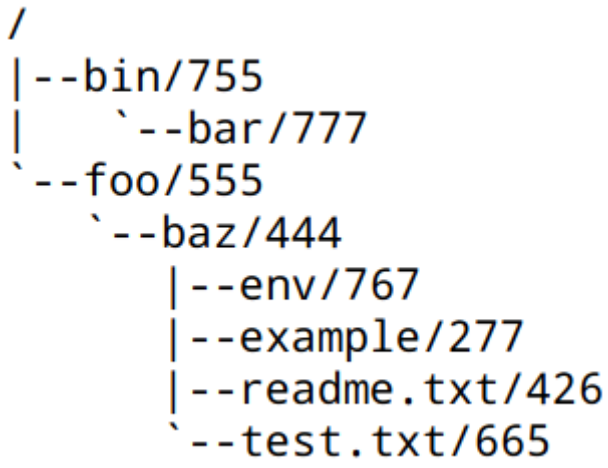


Задание 1. Вариант 9

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 10.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция mkdir.

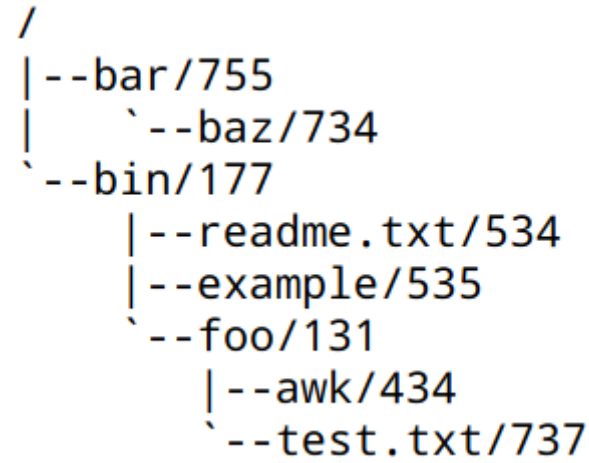


Задание 1. Вариант 10

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 11.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rename.

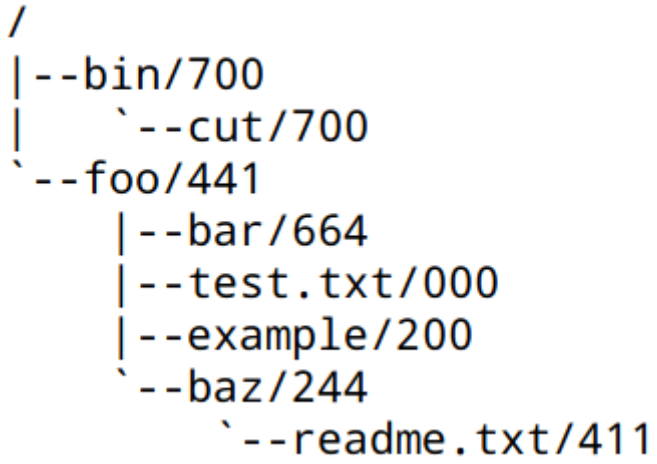


Задание 1. Вариант 11

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 12.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chown.

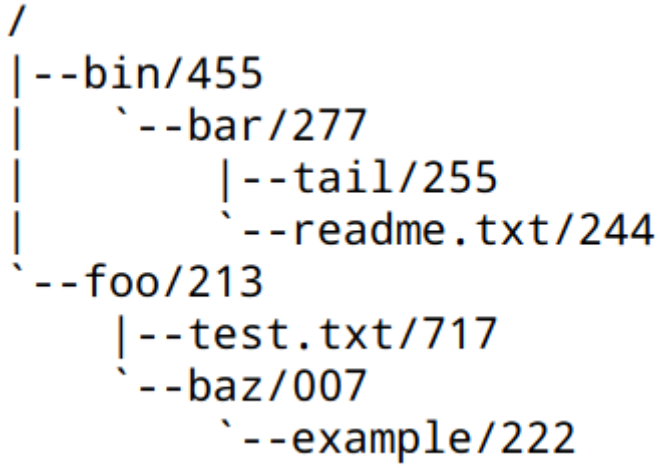


Задание 1. Вариант 12

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 13.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция rmdir.



Задание 1. Вариант 13

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 14.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция mkdir.

```
/
|--bar/000
|    |--readme.txt/555
|    `--baz/444
|         `--example/222
`--foo/711
     |--test.txt/000
     `--bin/766
          `--touch/777
```

Задание 1. Вариант 14

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 15.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция symlilnk.

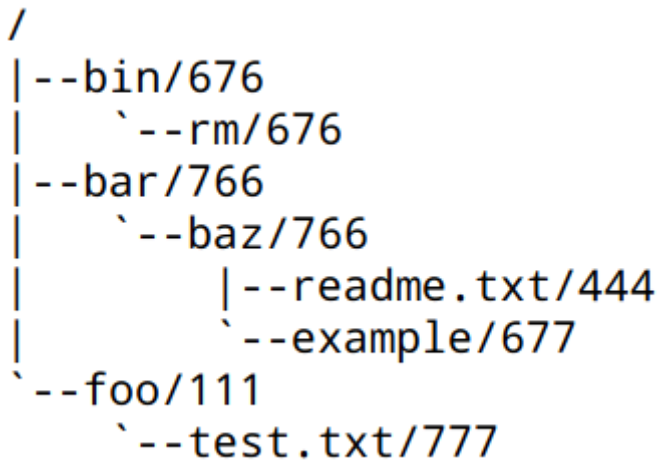
```
/
`--bin/700
    `--foo/441
        `--bar/664
            `--baz/244
                |--test.txt/000
                |--kill/700
                |--example/200
                `--readme.txt/411
```

Задание 1. Вариант 15

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 16.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция mkdir.

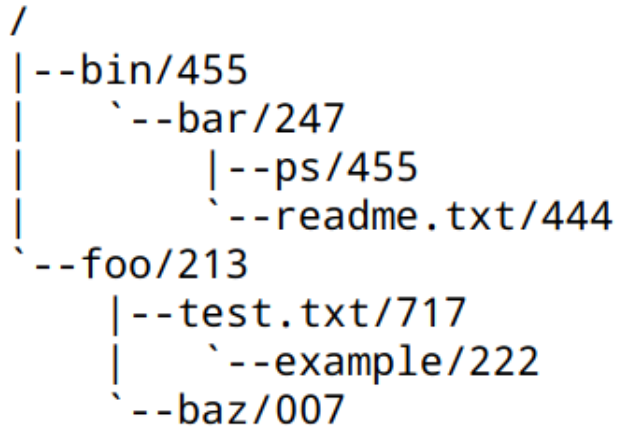


Задание 1. Вариант 16

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 17.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция symlink.

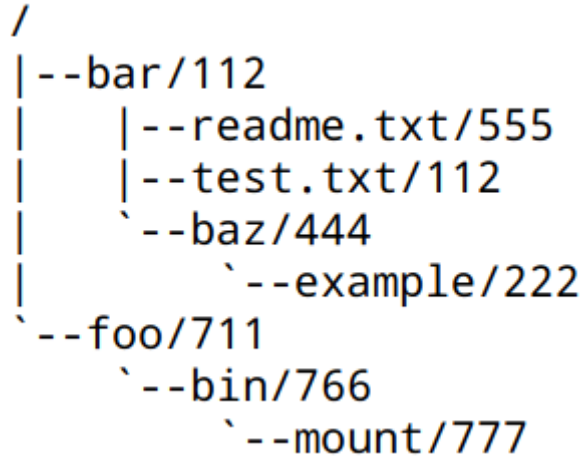


Задание 1. Вариант 17

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 18.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chown.

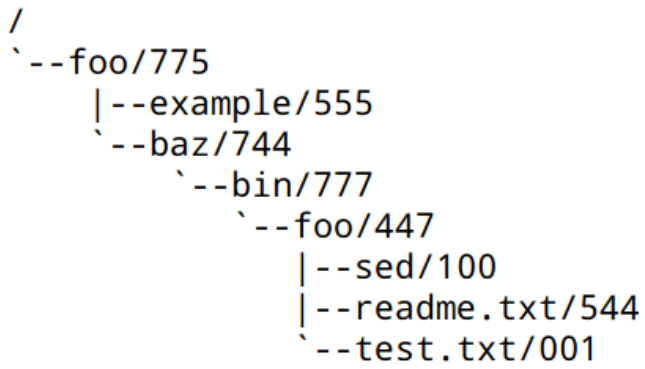


Задание 1. Вариант 18

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 19.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция chmod.

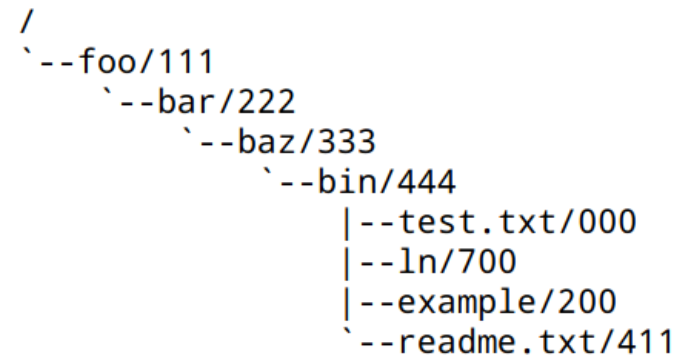


Задание 1. Вариант 19

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 20.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция mkdir.

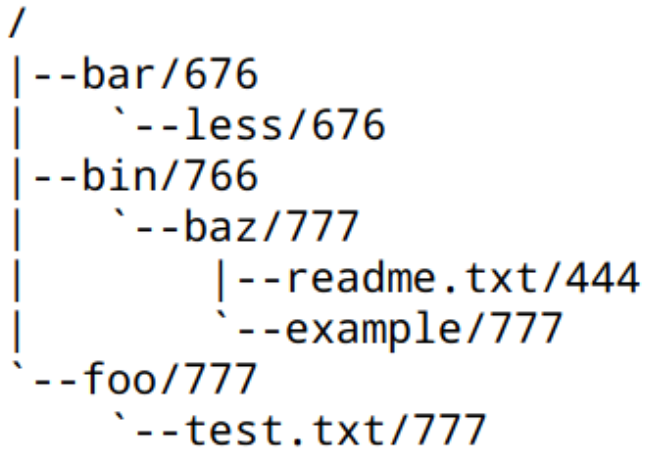


Задание 1. Вариант 20

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Вариант 21.

1. Дополнительно к условию задания 1 (общее описание) должна быть реализована функция symlink.



Задание 1. Вариант 21

2. Написать на языке C модуль ОС Linux, который реализует то же задание, что и в предыдущем задании (в этот раз без использования FUSE).

Контрольные вопросы

1. Какая структура катологов файловой системы FUSE? Опишите предназначение каждого каталога.

2. Перечислить основные этапы, которые необходимо выполнить с помощью системы сборки Autotools для генерации скрипта ./configure.

3.Для чего предназначен сценарий ./configure?

4. Какие функции реализованы в fuse_operation?

5. Как подключить модуль в ядро Linux? Какие команды требуются?

6. Как выполняется сборка модулей ядра?

7. Какая функция используется при регистрации новой файловой системы в ядре при загрузке модуля файловой системы?