БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

#### ОТЧЕТ

#### по лабораторной работе №4

По курсу “Программирование мобильных и встраиваемых систем”

***Системное программирование в ОС Linux***

**Подготовил:**  
студент 3 курса 13 группы кафедры ТП  
Петров Андрей Александрович

## Минск, 2021

## ЦЕЛИ РАБОТЫ:

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

## ЗАДАНИЕ 1.

Изучите примеры задания 3.1 и выполните их в ОС Raspberry PI.

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 С 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).

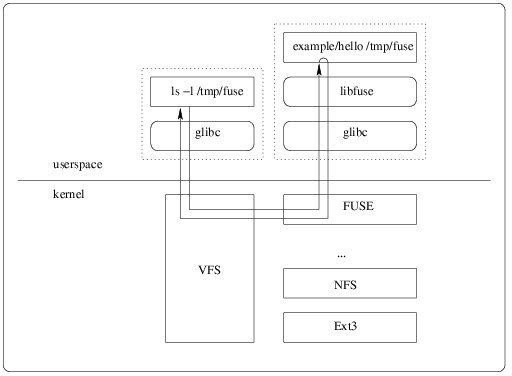
Файловая система содержит 4 директории: foo, bar, baz и bin, — а также 4 файла, из которых 3 — текстовые файлы: example, readme.txt, test, — и 1 бинарный. Содержимое бинарного файла должно быть взято из соответствующей стандартной системной утилиты, название которой соответствует названию файла:

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*

## 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.

(6\*6\*13) mod 22 = 6

### Вариант 6.

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

Изображение выглядит как текст

Автоматически созданное описание

*Задание 1. Вариант 6.*

Напишем на языке С программу с помощью библиотеки FUSE, которая подключает виртуальную файловую систему:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, электроника

Автоматически созданное описание

Изображение выглядит как текст, электроника

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

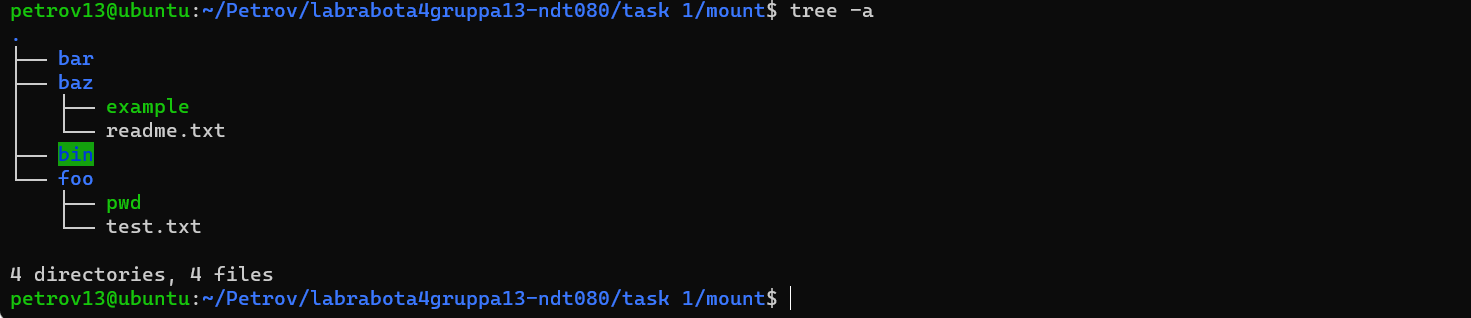
Автоматически созданное описание

Собираем программу и монтируем файловую систему:

Изображение выглядит как текст

Автоматически созданное описание

Проверяем структуру папок в смонтированной файловой системе:



Проверяем содержимое файлов:

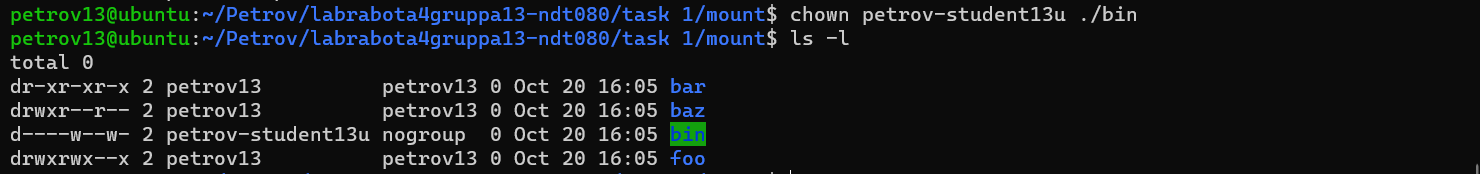




Изображение выглядит как текст

Автоматически созданное описание

Проверяем работу chown:

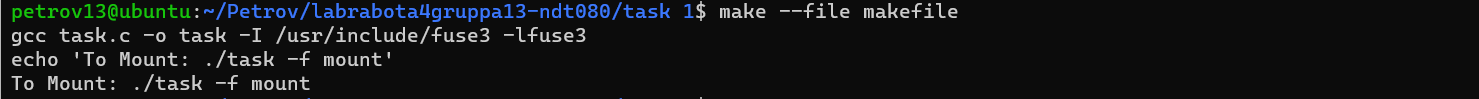


**Сборка**:

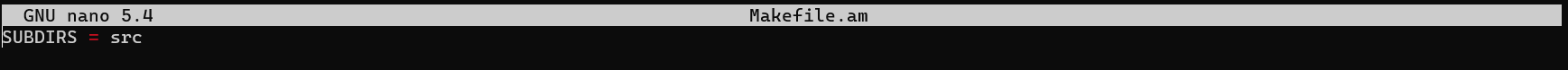
1. Для сборки создадим make file:

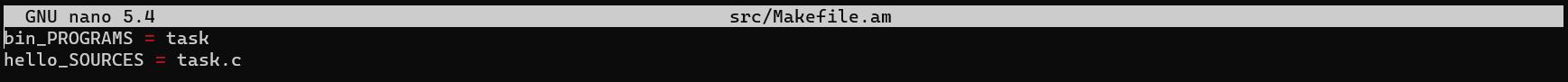
Изображение выглядит как текст

Автоматически созданное описание



1. Для сборки используем систему сборки Autotools с автоматической генерацией Makefile.





Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

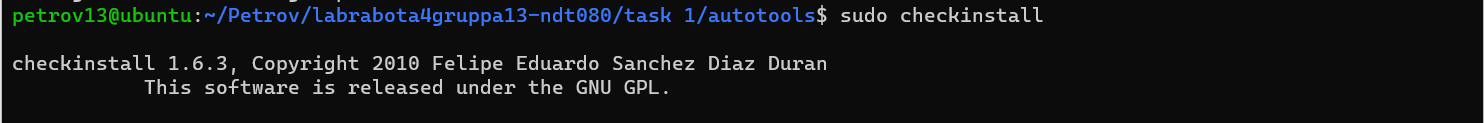
Автоматически созданное описание

Сформируем пакет с открытыми исходными кодами в формате tgz (tar.gz):

Изображение выглядит как текст

Автоматически созданное описание

Пакет также можно создать с помощью утилиты checkinstall:



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

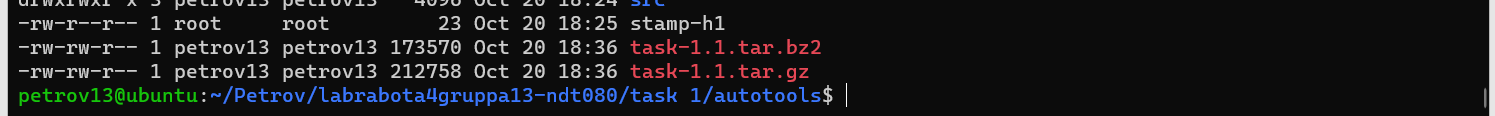
Или пакет также можно создать с помощью утилиты make distcheck:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



## ЗАДАНИЕ 2.

Написать на языке С 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).

# 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.

Cоздать файл configure.in или с помощью autoscan просканировать директорию и получить заготовку для будущего configure.in. В файле конфигурации прописать automake. После выполнить команды: autoheader, aclocal, automake, autoconf.

1. Для чего предназначен сценарий ./configure?

Сценарий конфигурации для создания makefile.

1. Какие функции реализованы в fuse\_operation?

Операции файловой системы. Все методы будут выполнять любую необходимую проверку разрешений. Вот список всех операций:

int(\* getattr )(const char \*, struct stat \*, struct fuse\_file\_info \*fi)

int(\* readlink )(const char \*, char \*, size\_t)

int(\* mknod )(const char \*, mode\_t, dev\_t)

int(\* mkdir )(const char \*, mode\_t)

int(\* unlink )(const char \*)

int(\* rmdir )(const char \*)

int(\* symlink )(const char \*, const char \*)

int(\* rename )(const char \*, const char \*, unsigned int flags)

int(\* link )(const char \*, const char \*

int(\* chmod )(const char \*, mode\_t, struct fuse\_file\_info \*fi)

int(\* chown )(const char \*, uid\_t, gid\_t, struct fuse\_file\_info \*fi)

int(\* truncate )(const char \*, off\_t, struct fuse\_file\_info \*fi)

int(\* open )(const char \*, struct fuse\_file\_info \*)

int(\* read )(const char \*, char \*, size\_t, off\_t, struct fuse\_file\_info \*)

int(\* write )(const char \*, const char \*, size\_t, off\_t, struct fuse\_file\_info \*)

int(\* statfs )(const char \*, struct statvfs \*)

int(\* flush )(const char \*, struct fuse\_file\_info \*)

int(\* release )(const char \*, struct fuse\_file\_info \*)

int(\* fsync )(const char \*, int, struct fuse\_file\_info \*)

int(\* setxattr )(const char \*, const char \*, const char \*, size\_t, int)

int(\* getxattr )(const char \*, const char \*, char \*, size\_t)

int(\* listxattr )(const char \*, char \*, size\_t)

int(\* removexattr )(const char \*, const char \*)

int(\* opendir )(const char \*, struct fuse\_file\_info \*)

int(\* readdir )(const char \*, void \*, fuse\_fill\_dir\_t, off\_t, struct fuse\_file\_info \*, enum fuse\_readdir\_flags)

int(\* releasedir )(const char \*, struct fuse\_file\_info \*)

int(\* fsyncdir )(const char \*, int, struct fuse\_file\_info \*)

void \*(\* init )(struct fuse\_conn\_info \*conn, struct fuse\_config \*cfg)

void(\* destroy )(void \*private\_data)

int(\* access )(const char \*, int)

int(\* create )(const char \*, mode\_t, struct fuse\_file\_info \*)

int(\* lock )(const char \*, struct fuse\_file\_info \*, int cmd, struct flock \*)

int(\* utimens )(const char \*, const struct timespec tv[2], struct fuse\_file\_info \*fi)

int(\* bmap )(const char \*, size\_t blocksize, uint64\_t \*idx)

int(\* ioctl )(const char \*, unsigned int cmd, void \*arg, struct fuse\_file\_info \*, unsigned int flags, void \*data)

int(\* poll )(const char \*, struct fuse\_file\_info \*, struct fuse\_pollhandle \*ph, unsigned \*reventsp)

int(\* write\_buf )(const char \*, struct fuse\_bufvec \*buf, off\_t off, struct fuse\_file\_info \*)

int(\* read\_buf )(const char \*, struct fuse\_bufvec \*\*bufp, size\_t size, off\_t off, struct fuse\_file\_info \*)

int(\* flock )(const char \*, struct fuse\_file\_info \*, int op)

int(\* fallocate )(const char \*, int, off\_t, off\_t, struct fuse\_file\_info \*)

off\_t(\* lseek )(const char \*, off\_t off, int whence, struct fuse\_file\_info \*)

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

Загрузить модуль ядра Linux можно с помощью команд modprobe или insmod.

1. Как выполняется сборка модулей ядра?

Для того, чтобы запустить процесс сборки модуля, нужно выполнить команду make -C /usr/src/linux-`uname -r` SUBDIRS=$PWD modules (если в каталоге /usr/src присутствует символическая ссылка linux на каталог с исходными текстами ядра, то команда сборки может быть несколько упрощена: make -C /usr/src/linux SUBDIRS=$PWD modules).

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

Зарегистрировать файловую систему можно с помощью вызова register\_filesystem