**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

Кафедра технологий программирования

Жуковский Павел Сергеевич

Программирование Мобильных и Встраиваемых Систем

Отчет по лабораторной работе №4

## «Управление процессами и потоками»

студента 3 курса 12 группы

**Преподаватель**

**Давидовская Мария Ивановна**

**Минск 2020**

**Вариант 9**

**Ход выполнения**

**Задание 1**

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.

**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.

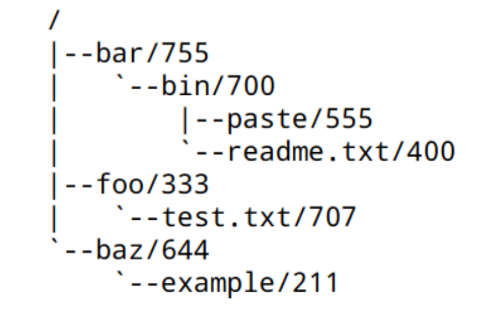
**Жуковский – 9 букв**

**Павел – 5 букв**

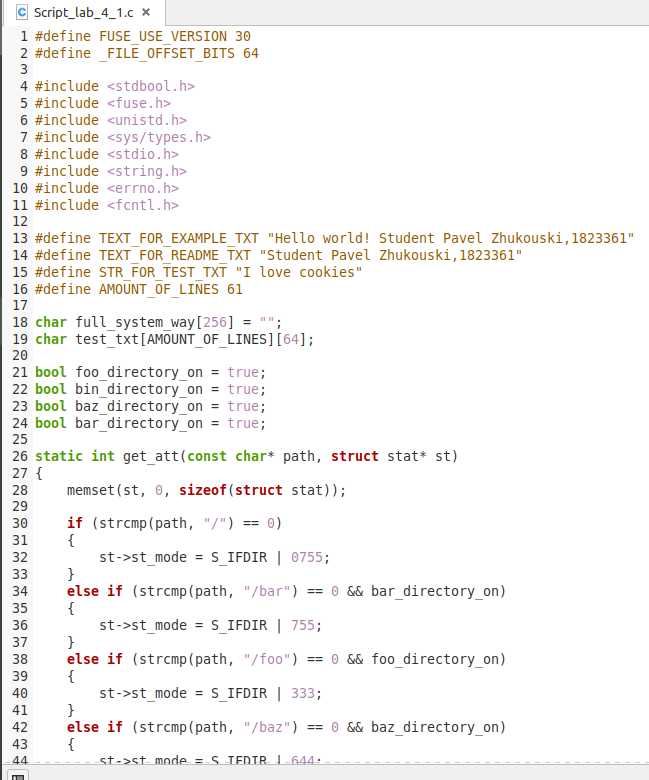
**Сергеевич – 9 букв**

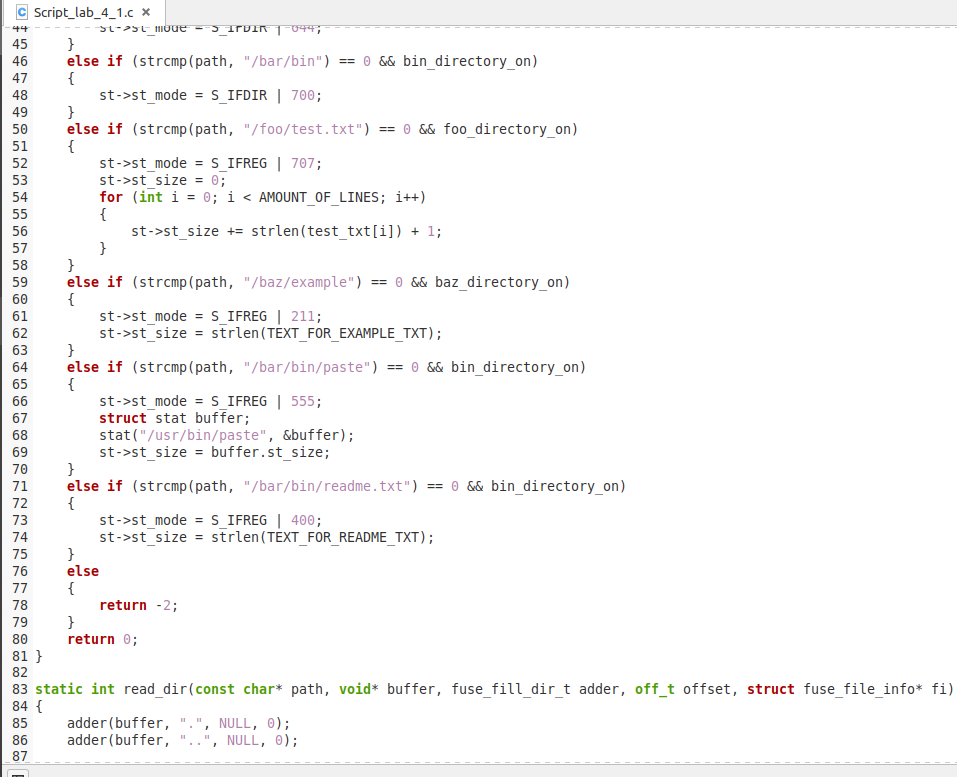
**k = (9\*5\*9) mod 22 = 405 mod 22 = 9 => 9 Вариант**

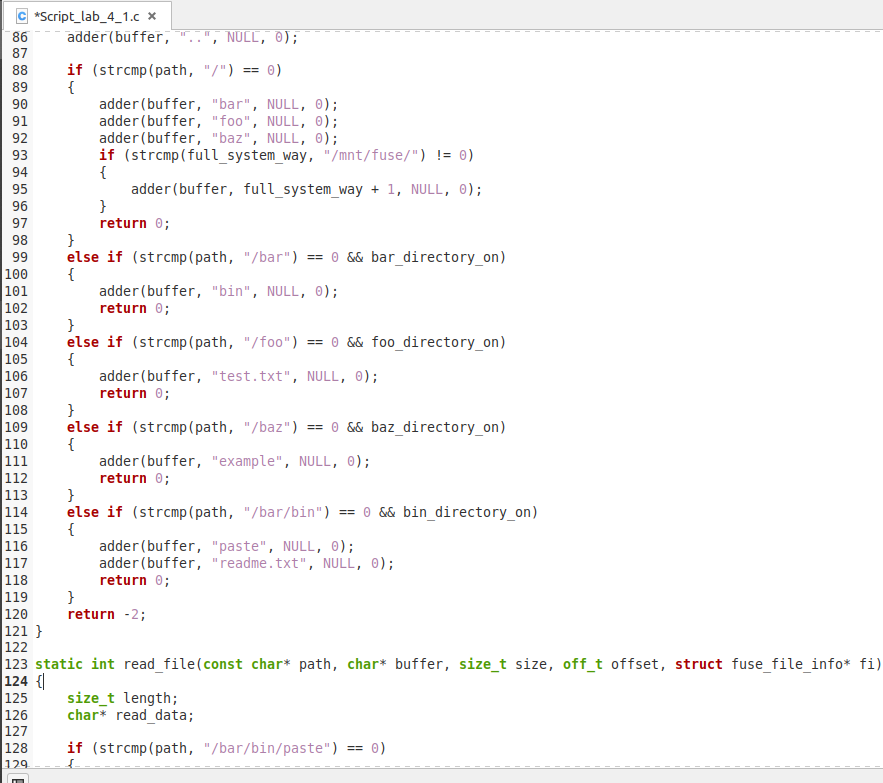
**Вариант 9**

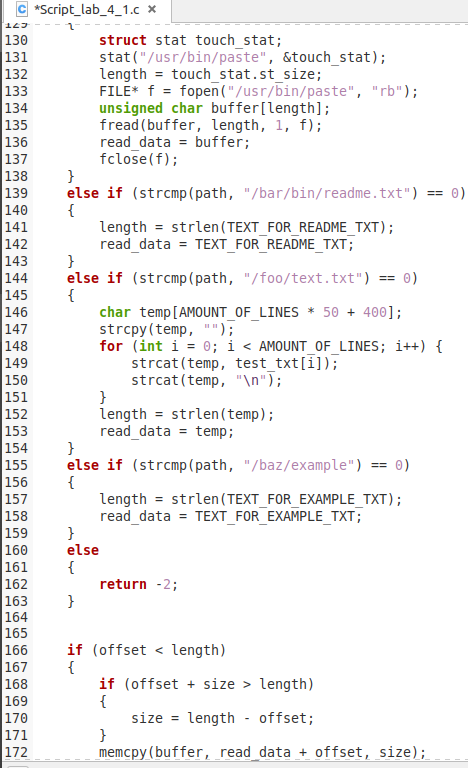


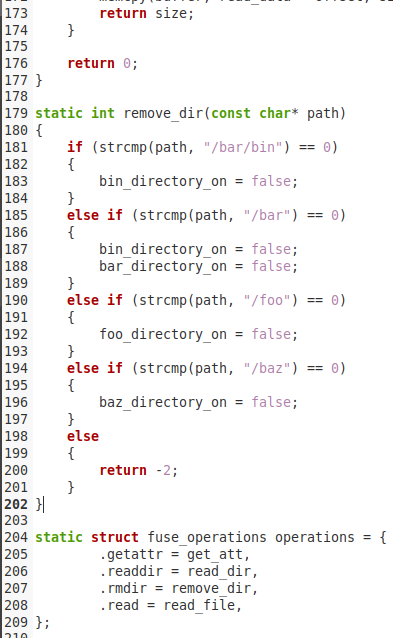
Исходный код:

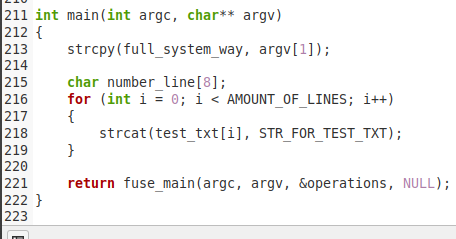




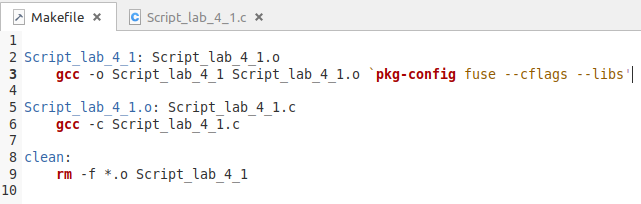








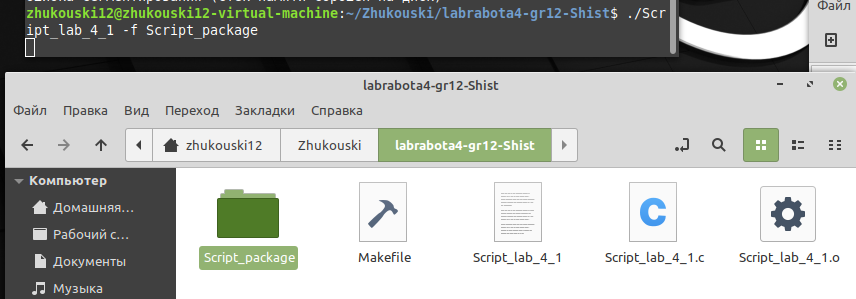
Содержимое Makefile для сборки:



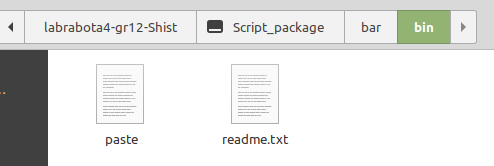
Сборка с помощью make:

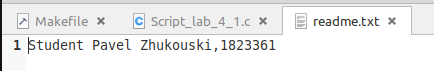


Запуск скрипта:

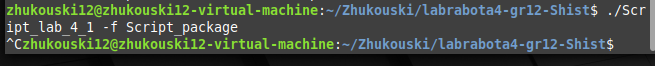


Скриншоты во время работы скрипта:





Завершение скрипта:



При завершении скрипта все созданные папки удаляются.

**Контрольные вопросы**

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

Файловая система FUSE состоит из двух основных компонентов: предохранитель модуля ядра, поддерживающийся в обычных хранилищах ядра, и libfuse библиотеки, поддерживающие пользовательское приложение в этом хранилище. При этом libfuse обеспечивает эталонную реализацию для связи с модулем FUSE ядра.

*2. Перечислить основные этапы, которые необходимо выполнить с помощью*

*системы сборки Autotools для генерации скрипта ./configure.*

1) Написание самой программы и сохранение её в файле \*.c

2) Создание configure.in и помещение в него необходимых макросов

3) Создание Makefile.am и src/Makefile.am

*3.Для чего предназначен сценарий ./configure?*

Сценарий configure предназначен для определения правильных значений для различных, зависящих от системы переменных, которые используются в процессе установки. Он использует эти переменные для создания файлов `Makefile' в каждом из каталогов пакета, а также может создавать один или несколько файлов `.h' содержащих зависящие от системы определения. В заключение, он создает скрипт командного процессора с именем `config.status', который вы можете в дальнейшем запускать для воссоздания текущей настройки, также создается файл `config.cache', который сохраняет результаты тестов, для ускорения перенастройки, и файл `config.log', содержащий вывод компилятора (этот файл в основном полезен для отладки configure).

*4. Какие функции реализованы в fuse\_operation?*

getattr()

readlink()

getdir()

mknod()

mkdir()

unlink()

rmdir()

symlink()

rename()

link()

chmod()

chown()

truncate()

utime()

open()

read()

write()

statfs()

flush()

release()

fsync()

setxattr()

getxattr()

listxattr

opendir()

readdir()

releasedir()

fsyncdir()

init()

destroy()

access()

create()

ftruncate()

fgetattr()

lock()

utimens()

bmap()

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

Загрузить модуль в ядро можно при помощи двух команд: «insmod» и «modprobe», отличающихся друг от друга возможностью просчета и удовлетворения зависимостей. Команда «insmod» загружает конкретный файл с расширением «ko», при этом, если модуль зависит от других модулей, еще не загруженных в ядро, команда выдаст ошибку, и не загрузит модуль. Команда «modprobe» работает только с деревом модулей, и возможна загрузка только оттуда по имени модуля, а не по имени файла. Отсюда следует область применения этих команд: при помощи «insmod» подгружается файл модуля из произвольного места файловой системы (например, пользователь скомпилировал модули и перед переносом в дерево ядра решил проверить его работоспособность), а «modprobe» — для подгрузки уже готовых модулей, включенных в дерево модулей текущей версии ядра. Например, для загрузки модуля ядра «rt73usb» из дерева ядра, включая все зависимости, и отключив аппаратное шифрование, нужно выполнить команду:  
  
*# modprobe rt73usb nohwcrypt=0*  
Загрузка этого модуля командой «insmod» произойдет следующим образом:  
  
*# insmod /lib/modules/2.6.38-gentoo-r1/kernel/drivers/net/wireless/rt2x00/rt73usb.ko nohwcrypt=0*  
Но нужно помнить, что при использовании «insmod» все зависимости придется подгружать вручную. Поэтому эта команда постепенно вытесняется командой «modprobe».

*6. Как выполняется сборка модулей ядра?*

Во первых, нужен компилятор. Обычно установка «gcc» устанавливает все, что нужно для сборки модуля. Если чего-то не хватает — программа сборки об этом скажет, и нужно будет доустановить недостающие пакеты.  
Во вторых, нужны заголовочные файлы ядра. Дело в том, что модули ядра всегда собираются вместе с ядром, используя его заголовочные файлы, т.к. любое отклонение и несоответствие версий модуля и загруженного ядра ведет к невозможности загрузить этот модуль в ядро.  
Если система работает на базе ядра дистрибутива, то нужно установить пакеты с заголовочными файлами ядра. В большинстве дистрибутивов это пакеты «kernel-headers» и/или «kernel-devel». Для сборки модулей этого будет достаточно. Если ядро собиралось вручную, то эти пакеты не нужны: достаточно сделать символическую ссылку "/usr/src/linux", ссылающуюся на дерево сконфигурированных исходных кодов текущего ядра.

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

Если мы хотим зарегистрировать новую файловую систему в ядре при загрузке модуля файловой системы, мы можем воспользоваться командой **init\_module**:

Вызов **init\_module**() загружает образ ELF в пространство ядра, выполняет все необходимые перемещения символов, инициализирует значения параметров модуля, предоставленные вызывающим и запускает функцию модуля *init*. Данный системный вызов требует дополнительных прав.