



ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

[C++ UNIX]: UNIX знакомство: useradd, nano, chmod, docker, GIT, CI, CD

Выполнил студент:

Козин Роман Андреевич

3 курс, Z33434

Санкт-Петербург
2023 г.

Цель

Познакомить студента с основами администрирования программных комплексов в ОС семейства UNIX, продемонстрировать особенности виртуализации и контейнеризации, продемонстрировать преимущества использования систем контроля версий (на примере GIT)

1 Задача 1

[ОС] Работа в ОС, использование файловой системы, прав доступа, использование файлов

1.1

В папке /USR/LOCAL/ создать 2 директории: folder_max, folder_min

Команды:

```
sudo mkdir folder_max  
sudo mkdir folder_min
```

Результат:

Созданы папки folder_max и folder_min

1.2

Создать 2-х группы пользователей: group_max, group_min

Команды:

```
sudo addgroup group_max  
sudo addgroup group_min
```

Результат:

Созданы группы group_max и group_min

1.3

Создать 2-х пользователей: user_max_1, user_min_1

Команды:

```
sudo adduser user_max_1  
sudo adduser user_min_1
```

Результат: созданы пользователи user_max_1 и user_min_1

Команды:

```
sudo usermod -a -G group_max user_max_1  
sudo usermod -a -G group_min user_min_1
```

Результат:

Пользователи user_max_1 и user_min_1 помещены в группы group_max и group_min соответственно

1.4

Для пользователей из группы *_max дать полный доступ на директории *_max и *_min. Для пользователей группы *_min дать полный доступ только на директорию *_min

Команды:

```
sudo chown :group_max folder_max
```

```
sudo chown :group_max folder_min
```

Результат:

Группа group_max является владельцем папок folder_max и folder_min

Команды:

```
sudo setfacl -m g:group_max:rwX folder_max
```

```
sudo setfacl -m g:group_max:rwX folder_min
```

```
sudo setfacl -m g:group_min:rwX folder_min
```

Результат:

Группа *_max обладает полным доступом к папкам *_max и *_min, т.к. является владельцем. Группа *_min обладает правом на чтение и исполнение к папке *_max и обладает полным доступом к папке *_min

1.5

Создать и исполнить (пользователем из той же категории) скрипт в директории folder_max, который пишет текущую дату/время в файл output.log в текущей директории

Команды:

```
vi script.sh - создание и открытие скрипта при помощи vim
```

```
chmod +x script.sh - разрешение на выполнение для скрипта
```

Скрипт:

```
#!/bin/sh
```

```
echo $(date -u) >> output.log
```

Команды:

./script.sh - выполнение скрипта, в файл output.log записываются (дополняются) текущие дата и время

Результат:

В папке *_max создан скрипт, который пишет дату и время в output.log

1.6

Создать и исполнить (пользователем из той же категории) скрипт в директории folder_max, который пишет текущую дату/время в файл output.log в директории *_min

Команды аналогичны, только скрипт:

```
#!/bin/sh
echo $(date -u) >> ../folder_min/output.log
```

Результат:

Дата и время записываются (дополняются) в файл output.log в папке *_min

1.7

Исполнить (пользователем *_min) скрипт в директории folder_max, который пишет текущую дату/время в файл output.log в директории *_min

Команды:

```
./script_min.sh
```

Результат:

При выполнении скрипта отказано в доступе для создания output.log, который был создан ранее из-за запуска скрипта пользователем user_max_1. Файл создаётся с правами 664, владелец – user_max_1. Если предварительно через chmod изменить права доступа файла на 777, то пользователь *_min сможет исполнить скрипт успешно.

1.8

Создать и исполнить (пользователем из той же категории) скрипт в директории folder_min, который пишет текущую дату/время в файл output.log в директории *_max

Скрипт:

```
#!/bin/sh
echo $(date -u) >> ../folder_min/output.log
```

Команды:

```
chmod +x script_max.sh - разрешение на запуск скрипта
```

Результат:

Исполнить скрипт не удалось, так как нет разрешения на создание output.log в папке *_max

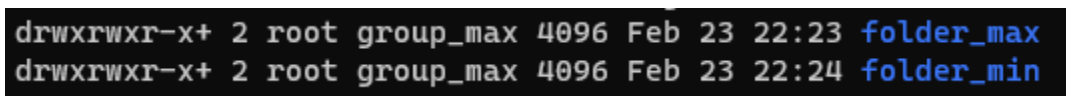
1.9

Вывести перечень прав доступа у папок *_min/ *_max, а также у всего содержимого внутри

Команда (внутри папки local):

```
ls -l
```

Результат:



```
drwxrwxr-x+ 2 root group_max 4096 Feb 23 22:23 folder_max
drwxrwxr-x+ 2 root group_max 4096 Feb 23 22:24 folder_min
```

Рис. 1: Права папок max и min

Команда (внутри папки local):

```
ls -l folder_max
```

Результат:

```
-rw-rw-r-- 1 user_max_1 user_max_1 58 Feb 23 21:24 output.log
-rwxrwxr-x 1 user_max_1 user_max_1 40 Feb 23 21:19 script.sh
-rwxrwxr-x 1 user_max_1 user_max_1 54 Feb 23 21:35 script_min.sh
```

Рис. 2: Права содержимого папки max

Команда (внутри папки local):

```
ls -l folder_min
```

Результат:

```
-rw-rw-r-- 1 user_max_1 user_max_1 29 Feb 23 21:57 output.log
-rwxrwxr-x 1 user_min_1 user_min_1 54 Feb 23 22:24 script_max.sh
```

Рис. 3: Права содержимого папки min

2 Задача 2

[КОНТЕЙНЕР] docker build / run / ps / images

2.1

Создать скрипт, который пишет текущую дату/время в файл output.log в текущей директории

Скрипт:

```
#!/bin/sh
```

```
echo $(date -u) >> output.log
```

Результат:

Скрипт создан

2.2

Собрать образ со скриптами выше и с пакетом nano (docker build)

Создаём файл Dockerfile, его содержание:

```
FROM ubuntu:20.04 #базовый образ для создаваемого образа
```

```
RUN apt update -y && apt install -y nano #установка в образ nano
```

```
COPY ./script.sh /script.sh #копирование скрипта в образ
```

В терминале пишем:

```
docker build -t lab1 .
```

Результат:

Собран образ со скриптом script.sh и с пакетом nano

2.3

Запустить образ

В терминале:

```
docker run --rm lab1 bash script.sh
```

Результат:

Скрипт выполняется, дата записывается в output.log. --rm используется для удаления контейнера после его использования

2.4

Выполнить скрипт, который подложили при сборке образа

При исполнении скрипт пишет текущую дату и время в файл output.log

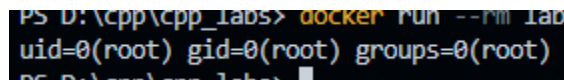
2.5

Вывести список пользователей в собранном образе

В терминале:

```
docker run --rm lab1 id
```

Результат:



```
PS D:\cpp\cpp_labs> docker run --rm lab1 id
uid=0(root) gid=0(root) groups=0(root)
```

Рис. 4: Список всех пользователей в собранном образе

В собранном образе только один пользователь root

3 Задача 3

[GIT] GitHub / GitLab, в котором будут содержаться все выполненные ЛР

3.1

Создать репозиторий в GitHub или GitLab

Действия:

Создание репозитория cpp_labs на github кнопкой new.

Результат:

Создан репозиторий cpp_labs

3.2

Создать структуру репозитория

Создаём структуру репозитория в vs code:

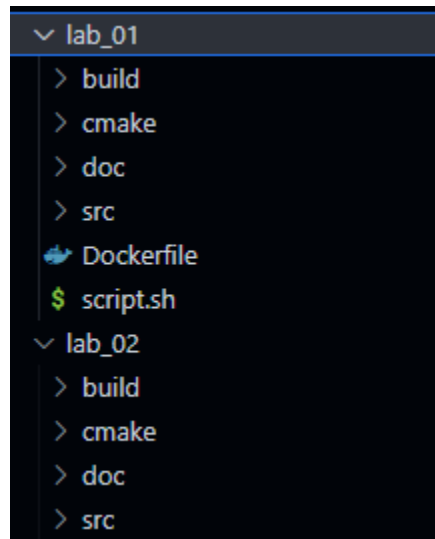


Рис. 5: Структура репозитория crr_labs

Добавляем структуру в репозиторий:

```
git init
```

```
git add -A
```

```
git commit -m "first commit"
```

Результат:

В репозитории создана требуемая структура

3.3

Создать ветки dev / stg / prd, удалить ранее существующие ветки удаленно и локально

Команды:

```
git branch -M prd
```

```
git push -u origin prd
```

```
git branch stg
```

```
git push -u origin stg
```

```
git branch dev
```

```
git push -u origin dev
```

Результат:

Созданы ветки prd, stg и dev, ранее существующих веток не было.

3.4

Создать скрипт автоматического переноса ревизий из ветки dev в ветку stg с установкой метки времени (tag). Скрипт в корень репозитория

Создан скрипт с содержанием:

```
#!/bin/bash VAR=$(date '+%d.%m.%Y.%H.%M.%S') #date in given format
git checkout stg #switch to stg
git merge --commit dev #merge with dev
git tag "$VAR" #tag the date git push origin stg
git push origin "$VAR"
git checkout dev #switch to dev
git commit -m "dev merged into stg"
```

Результат: После commit и исполнения скрипта ревизии из ветки dev были перемещены в stg с установкой метки времени

3.5

Создать скрипт автоматического переноса ревизий из ветки stg в ветку prd с установкой метки времени (tag). Скрипт в корень репозитория

Создан скрипт с содержанием:

```
#!/bin/bash VAR=$(date '+%d.%m.%Y.%H.%M.%S') #date in given format
git checkout prd #switch to prd
git merge --commit stg #merge with stg
git tag "$VAR" #tag the date git push origin prd
git push origin "$VAR"
git checkout dev #switch to dev
git commit -m "stg merged into prd"
```

Результат аналогичен предыдущему пункту, только теперь ревизии перенесены из stg в prd.

4 Выводы

Всё, что я делал в частях 1 и 2 этой лабораторной работы, было для меня новым, но команды и действия не были особо сложными, так что разобраться в чём-то самостоятельно или при помощи лекций не составило особого труда. С git я уже был частично знаком, так что в части 3 лабораторной я вспоминал действия с репозиториями.