

Реквизиты

Жолобов Даниил Валерьевич

3 курс

z33434

2024

ЛР #1: [C++ & UNIX]: UNIX знакомство: useradd, nano, chmod, docker, GIT, CI, CD

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)

Задача

1. [ОС] Работа в ОС, использование файловой системы, прав доступа, исполнение файлов
 - 1.1. В папке /USR/LOCAL/ создать 2 директории: folder_max, folder_min
 - 1.2. Создать 2-х группы пользователей: group_max, group_min
 - 1.3. Создать 2-х пользователей: user_max_1, user_min_1
 - 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

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

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`

1.7.

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

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`

1.9.

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

2. [КОНТЕЙНЕР] `docker build / run / ps / images`

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

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

2.3. Запустить образ (`docker run`)

- 2.4. Выполнить скрипт, который подложили при сборке образа
- 2.5. Вывести список пользователей в собранном образе
- 3. [GIT] GitHub / GitLab, в котором будут содержаться все выполненные ЛР
 - 3.1. Создать репозиторий в GitHub или GitLab
 - 3.2. Создать структуру репозитория:
 - 3.2.1. lab_01
 - 3.2.1.1. build
 - 3.2.1.2. src
 - 3.2.1.3. doc
 - 3.2.1.4. cmake (для ЛР 1 опционально)
 - 3.2.2. lab_02
 - 3.2.2.1.
 - ... идентично lab_01 ...
 - 3.3. Создать ветки dev / stg / prd, удалить ранее существующие ветки удаленно и локально
 - 3.4. Создать скрипт автоматического переноса ревизий из ветки dev в ветку stg с установкой метки времени (tag). Скрипт в корень репозитория
 - 3.5. Создать скрипт автоматического переноса ревизий из ветки stg в ветку prd с установкой метки времени (tag). Скрипт в корень репозитория
- 4. [SAVE] Всё, что было сделано в шагах 1-3, сохранить в репозиторий (+ отчет по данной ЛР в папку doc). Фиксацию ревизий производить строго через ветку dev. С помощью скриптов накатить ревизии на stg и на prd.

Решение

- 1. [OC] Работа в ОС, использование файловой системы, прав доступа, исполнение файлов
 - 1.1. В папке /usr/local/ создать 2 директории: folder_max, folder_min

```
cd /usr/local
sudo mkdir folder_max folder_min
```

```
ls
```

1.2. Создать 2-х группы пользователей: group_max, group_min

```
sudo groupadd group_max  
sudo groupadd group_min  
cat /etc/group
```

1.3. Создать 2-х пользователей: user_max_1, user_min_1

```
sudo useradd user_max_1  
sudo useradd user_min_1  
cat /etc/passwd
```

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

```
sudo chgrp -R group_max ./folder_max  
sudo chgrp -R group_min ./folder_min  
sudo chmod -R g+wx ./folder_max  
sudo chmod -R g+wx ./folder_min  
sudo setfacl -R -m g:group_max:rwX ./folder_min/  
getfacl ./folder_max  
getfacl ./folder_min  
ls -l
```

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

```
cd ./folder_max/  
su - user_max_1  
echo "date > ./output.log" > script.sh  
chmod u+x ./script.sh  
./script.sh  
cat ./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`

```
echo "date > ../folder_min/output.log" > script2.sh  
chmod u+x ./script2.sh  
./script2.sh  
cat ../folder_min/output.log
```

1.7.

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

```
su - user_min_1  
./script2.sh  
--> -sh: 2: ./script2.sh: Permission denied
```

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`

```
cd ../folder_min
echo "date > ../folder_max/output.log" > script3.sh
chmod u+x ./script3.sh
./script3.sh
--> ./script3.sh: 1: cannot create ../folder_max/output.log:
Permission denied
```

1.9.

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

```
tree -ugp
-->
├─ [drwxrwxr-x root      group_max]  folder_max
│   ├─ [-rw-rw-r-- user_max_1 user_max_1]  output.log
│   ├─ [-rwxrw-r-- user_max_1 user_max_1]  script2.sh
│   └─ [-rwxrw-r-- user_max_1 user_max_1]  script.sh
├─ [drwxrwxr-x root      group_min]  folder_min
│   ├─ [-rw-rw-r-- user_max_1 user_max_1]  output.log
│   └─ [-rwxrw-r-- user_min_1 user_min_1]  script3.sh
```

2. [КОНТЕЙНЕР] docker build / run / ps / images

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

```
echo "date > ./output.log" > script.sh
sudo chmod +x ./script.sh
```

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

```
printf "FROM alpine:3.14
COPY ./script.sh .
```

```
RUN apk add --no-cache bash nano" > Dockerfile
sudo docker build -t script .
```

2.3. Запустить образ (docker run)

```
sudo docker run -it script bash
```

2.4. Выполнить скрипт, который подложили при сборке образа

```
./script.sh
```

2.5. Вывести список пользователей в собранном образе

```
cat /etc/passwd
```

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

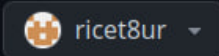
3.1. Создать репозиторий в GitHub или GitLab

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *



Repository name *

/ labs-cpp


✔ labs-cpp is available.

Great repository names are short and memorable. Need inspiration? How about **improved-parakeet** ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

3.2. Создать структуру репозитория:

3.2.1. lab_01

3.2.1.1. build

3.2.1.2. src

3.2.1.3. doc

3.2.1.4. cmake (для ЛР 1 опционально)

3.2.2. lab_02

3.2.2.1.

... идентично lab_01 ...


```
mkdir lab_01 lab_02
mv 1.md ./lab_01/
mv pic-create-repo.png ./lab_01/
mv Dockerfile ./lab_01/
mv script.sh ./lab_01/
cd lab_01
mkdir build src doc
cd ../lab_02
mkdir build src doc
```

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

```
git init
git branch -m dev
git add .
git commit -m "init + lab 1"
git branch stg
git branch prd
git remote add origin git@github.com:ricet8ur/labs-cpp.git
git push --set-upstream origin dev
git push --all origin
```

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

```
printf "
git checkout stg
git merge dev
git tag `date +"%%F_%%H-%%M-%%S"``
git checkout dev" > merge_dev2stg.sh
sudo chmod +x ./merge_dev2stg.sh
```

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

```
printf "  
git checkout prd  
git merge stg  
git tag `date +%F_%H-%M-%S`\  
git checkout dev" > merge_stg2prd.sh  
sudo chmod +x ./merge_stg2prd.sh
```

4. [SAVE] Всё, что было сделано в шагах 1-3, сохранить в репозиторий (+ отчет по данной ЛР в папку doc). Фиксацию ревизий производить строго через ветку dev. С помощью скриптов накатить ревизии на stg и на prd.

```
git add .  
git commit -m "finish lab 1"  
./merge_dev2stg.sh  
./merge_stg2prd.sh  
git push --all origin  
git push origin --tags
```

Заключение

Получил первый опыт работы с правами пользователей на файлы и папки. Узнал про существование `setfacl`, `getfacl`. Получше разобрался с Docker: ознакомился с различиями `docker-ce` и `docker.io`, научился подключаться к контейнеру через `bash`. Узнал про форматирование даты через утилиту `date`. Научился писать "вложенные" скрипты - с вызовами команд и использованием их результатов.