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, исполение 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.**

Выполнение

1. **[ОС] Работа в ОС, использование файловой системы, прав доступа, исполение файлов**

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

```
cd /usr/local

mkdir folder_max folder_min
```

1.2 Создать 2-х группы пользователей: group_max, group_min

```
groupadd group_max

groupadd group_min
```

1.3 Создать 2-х пользователей: user_max_1, user_min_1

```
useradd user_max_1

useradd 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

```
sudo apt install acl

setfacl -m:group_min:rwx folder_min

setfacl -m:group_max:rwx folder_max folder_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

```
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
```

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

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
```

не будет выполнено

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

```
ls -l
```

и аналогично внутри папок

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

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 lab_03 lab_04 lab_05

cd lab_01

mkdir build src doc

аналогично для остальных лаб

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"

git branch stg

git branch prd

git remote add origin git@github.com:

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

Заключение

Я получил первый опыт работы с Linux, разобрался с правами доступа. Научился писать простые скрипты. Создал первый контейнер и выполнил в нем простой скрипт с помощью docker. Разобрался с основами работы с гитом