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

Реквизиты

Семушев Климент Витальевич, Z3243 (2 курс), 2024 год

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

1.1.1. └─$ cd ../usr/local

1.1.2. └─$ sudo mkdir folder_max

1.1.3. └─$ sudo mkdir folder_min

1.1.4. Результат: созданы папки, проверено их существование командой ls

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

1.2.1. └─$ sudo groupadd –p maxpass group_max

1.2.2. └─$ sudo groupadd group_min

1.2.3. Результат: созданы группы, проверено их существование командой cat /etc/group

```
group_min:x:1001:
group_max:x:1002:
```

1.3. Создать 2-х пользователей: user_max_1, user_min_1

1.3.1. └─$ sudo adduser --gid 1001 user_min_1 (пароль passmin (это чтобы я сам не забыл))

1.3.2. └─$ sudo adduser --gid 1002 user_max_1

1.3.3. └─$ sudo usermod -a -G 1001 user_max_1

1.3.4. Результат: созданы пользователи, проверено их существование командой cat /etc/passwd

```
user_min_1:x:1003:1001:,,,:/home/user_min_1:/bin/bash
user_max_1:x:1004:1002:,,,:/home/user_max_1:/bin/bash
```

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

1.4.1. └─$ sudo chmod 774 folder_max

1.4.2. └─$ sudo chmod 774 folder_min

1.4.3. └─$ sudo chown user_min_1:group_min folder_min

1.4.4. └─$ sudo chown user_max_1:group_max folder_max

Результат: изменены права доступа на папки, изменены владельцы папок в соответствии с заданием, проверено командой ls -l

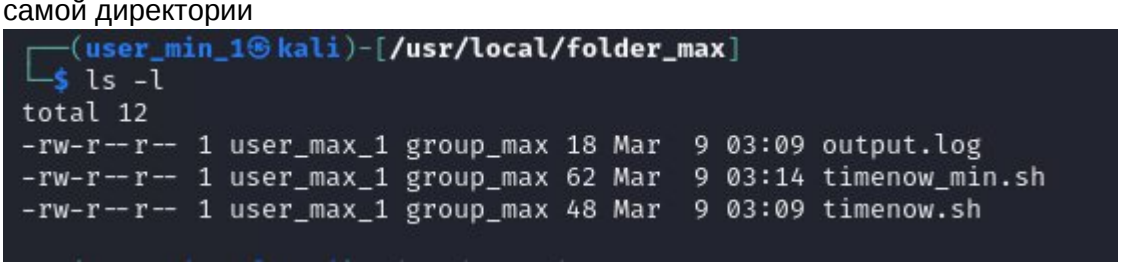
```
total 40
drwxr-xr-x 2 root      root 4096 Nov 30 14:52 bin
drwxr-xr-x 2 root      root 4096 Nov 30 11:35 etc
drwxrwxr-- 2 user_max_1 group_max 4096 Mar  9 02:02 folder_max
drwxrwxr-- 2 user_min_1 group_min 4096 Mar  9 02:04 folder_min
...
```

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

1.5.1. └─$ su user_max_1
1.5.2. └─$ cd folder_max
1.5.3. └─$ nano timenow.sh
1.5.3.1. echo $(date '+%d.%m.%Y %H:%M:%S') >> output.log
1.5.3.2. ^S^X
1.5.4. └─$ bash timenow.sh
1.5.5. Проверка: └─$ nano output.log – в формате append в заданном формате в файл пишется дата/время

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
1.6.1. └─$ nano timenow_min.sh
1.6.1.1. echo $(date '+%d.%m.%Y %H:%M:%S') >> ../folder_min/output.log
1.6.1.2. ^S^X
1.6.2. └─$ bash timenow_min.sh
1.6.3. Проверка: └─$ nano ../folder_min/output.log – в формате append в заданном формате в файл пишется дата/время

1.7. Исполнить (пользователем *_min) скрипт в директории folder_max, который пишет текущую дату/время в файл output.log в директории *_min
1.7.1. └─$ su user_min_1
1.7.2. └─$ bash timenow_min.sh
1.7.3. bash: ... Permission denied -> надо поменять права на 775 -> поменял за кулисами
1.7.4. └─$ bash timenow_min.sh
1.7.5. Не помогло: права на файлы в директории не совпадают с правами самой директории



1.7.6. Апдейт: вероятно дело даже не в правах на сами файлы, а в том, что изначальный output.log создан пользователем из другой группы, запущенный скрипт из-под user_min_1 не имеет прав на запись в файл, созданный из-под user_max_1

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
1.8.1. └─$ cd folder_min
1.8.2. └─$ nano timenow_max.sh
1.8.3. echo $(date '+%d.%m.%Y %H:%M:%S') >> ../folder_max/output.log
1.8.4. ^S^X

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

```
┌──(user_min_1㉿kali)-[/usr/local]
└─$ ls -l
total 40
drwxr-xr-x 2 root       root      4096 Nov 30 14:52 bin
drwxr-xr-x 2 root       root      4096 Nov 30 11:35 etc
drwxrwxr-x 2 user_max_1 group_max 4096 Mar  9 03:14 folder_max
drwxrwxr-- 2 user_min_1 group_min 4096 Mar 15 10:47 folder_min
drwxr-xr-x 2 root       root      4096 Nov 30 11:35 games
drwxr-xr-x 2 root       root      4096 Nov 30 11:35 include
drwxr-xr-x 4 root       root      4096 Nov 30 11:44 lib
lrwxrwxrwx 1 root       root         9 Nov 30 11:35 man → share/man
drwxr-xr-x 2 root       root      4096 Feb 21 04:20 sbin
drwxr-xr-x 9 root       root      4096 Nov 30 11:51 share
drwxr-xr-x 2 root       root      4096 Nov 30 11:35 src

┌──(user_min_1㉿kali)-[/usr/local]
└─$ ls -l folder_max
total 12
-rw-r--r-- 1 user_max_1 group_max 18 Mar  9 03:09 output.log
-rw-r--r-- 1 user_max_1 group_max 62 Mar  9 03:14 timenow_min.sh
-rw-r--r-- 1 user_max_1 group_max 48 Mar  9 03:09 timenow.sh

┌──(user_min_1㉿kali)-[/usr/local]
└─$ ls -l folder_min
total 8
-rw-r--r-- 1 user_max_1 group_max 20 Mar  9 03:25 output.log
-rw-r--r-- 1 user_min_1 group_min 63 Mar 15 10:47 timenow_max.sh

┌──(user_min_1㉿kali)-[/usr/local]
└─$
```

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)

```
  GNU nano 7.2
FROM ubuntu:22.04

ADD script.sh /

RUN apt-get update && \
        apt-get install nano && \
        bash script.sh
```

```
┌──(kali㉿kali)-[/usr/local/director]
└─$ sudo docker build -t logger .
Sending build context to Docker daemon  4.096kB
Step 1/3 : FROM ubuntu:22.04
 ---> ca2b0f26964c
Step 2/3 : ADD script.sh /
 ---> Using cache
 ---> bedfbd9ace39
Step 3/3 : RUN apt-get update &&        apt-get install nano &&        bash script.sh
 ---> Using cache
 ---> 39c3e31d9ed4
Successfully built 39c3e31d9ed4
Successfully tagged logger:latest
```

2.3. Запустить образ (docker run)

2.3.1. sudo docker run –it logger

2.4. Выполнить скрипт, который подложили при сборке образа

2.4.1. bash script.sh

2.4.2. nano output.log

```
  GNU nano 6.2   output.log
16:08:43 15.03.2024
20:04:32 15.03.2024
```

2.4.3. Здесь 16:08:43 – время инициализации контейнера, в которой я тоже прописал запуск скрипта. Из-за того, что я не до конца был уверен в своих действиях, прошло 4 часа до момента, когда я все-таки запустил образ и скрипт в нем)

2.5. Вывести список пользователей в собранном образе

```
root@e93f3123cb0a:/# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

2.5.1. Вижу, что существуют все дефолтные пользователи, включая _apt, потому что я пользовался apt-get при инициализации контейнера

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

3.1.1.1.1. Обновлю Dockerfile для поддержки git с ssh

```
  GNU nano 7.2                    Dockerfile
FROM ubuntu:22.04
MAINTAINER Kliment Semushev

RUN apt-get update && \
        apt-get install -y nano
RUN apt-get install -y git

RUN mkdir /root/.ssh/
ADD id_rsa /root/.ssh/id_rsa
RUN touch /root/.ssh/known_hosts
RUN ssh-keyscan -t rsa github.com > /root/.ssh/known_hosts
WORKDIR /home
ADD script.sh .
RUN bash script.sh
RUN git clone git@github.com:dyed-eye/cpp-and-unix.git /home/assignments
```

3.1.1.1.2. Пересоберу образ, сбросив кэш, поскольку файловую организацию пришлось немного поменять

3.1.1.1.2.1. sudo docker build –t logger —no-cache .

3.1.1.1.2.2. sudo docker run –it logger

3.2. Создать репозиторий в GitHub или GitLab

3.2.1.1.1. https://github.com/dyed-eye/cpp-and-unix

3.3. Создать структуру репозитория:

3.3.1. lab_01

3.3.1.1. build

3.3.1.2. src

3.3.1.3. doc
3.3.1.4. cmake (для ЛР 1 опционально)
3.3.2. lab_02
3.3.2.1. ... идентично lab_01 …

```
root@6502dfb6b351:/home/assignments# mkdir lab_01/ lab_01/build/ lab_01/src/ lab_01/doc/ lab_01/cmake/
root@6502dfb6b351:/home/assignments# mkdir lab_02/ lab_02/build/ lab_02/src/ lab_02/doc/ lab_02/cmake/
```

3.4. Создать ветки dev / stg / prd, удалить ранее существующие ветки удаленно и локально
3.4.1. git checkout –b dev (stg/prd)
3.4.2. git push –u origin dev(stg/prd)
3.4.3. git branch –d main
3.4.4. git branch –rd origin/main
3.4.5. Проверка:

```
root@6502dfb6b351:/home/assignments# git branch --all
  dev
  prd
* stg
  remotes/origin/dev
  remotes/origin/prd
  remotes/origin/stg
```

3.5. Создать скрипт автоматического переноса ревизий из ветки dev в ветку stg с установкой метки времени (tag). Скрипт в корень репозитория
3.5.1. git checkout dev
3.5.2. nano stage.sh

```
  GNU nano 6.2                                stage.sh *
git checkout stg
git merge dev
timetag=$(date '+%H:%M:%S %d.%m.%Y')
git tag $timetag
git checkout dev
```

3.5.3. git add .
3.5.4. git commit –m «stage script»
3.5.5. bash stage.sh

```
root@6502dfb6b351:/home/assignments# git log --graph --decorate --oneline
* 24adad5 (HEAD → dev) stage script
* baddc28 (tag: clear_setup, origin/stg, origin/prd, origin/dev, stg, prd) In
itial commit
root@6502dfb6b351:/home/assignments# bash stage.sh
Switched to branch 'stg'
Your branch is up to date with 'origin/stg'.
Updating baddc28..24adad5
Fast-forward
 stage.sh | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 stage.sh
fatal: Failed to resolve '16.03.2024' as a valid ref.
Switched to branch 'dev'
Your branch is ahead of 'origin/dev' by 1 commit.
  (use "git push" to publish your local commits)
root@6502dfb6b351:/home/assignments# git log --graph --decorate --oneline
* 24adad5 (HEAD → dev, stg) stage script
* baddc28 (tag: clear_setup, origin/stg, origin/prd, origin/dev, prd) Initial
 commit
root@6502dfb6b351:/home/assignments#
```

3.5.6. Не сработало, поменяем форматирование тэга

```
 GNU nano 6.2                                 stage.sh
git checkout stg
git merge dev
timetag=$(date '+%H-%M-%S')
git tag $timetag
git checkout dev
```

3.5.7. bash stage.sh

```
root@b2e837d0ff7b:/home/assignments# git commit -m "stage script"
[dev 531ad43] stage script
 1 file changed, 5 insertions(+)
 create mode 100644 stage.sh
root@b2e837d0ff7b:/home/assignments# bash stage.sh
Switched to branch 'stg'
Your branch is up to date with 'origin/stg'.
Updating baddc28..531ad43
Fast-forward
 stage.sh | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 stage.sh
Switched to branch 'dev'
Your branch is ahead of 'origin/dev' by 1 commit.
  (use "git push" to publish your local commits)
```

3.5.8. Сработало, проверяем:

```
root@6502dfb6b351:/home/assignments# git show-ref --tags -d
24adad594e31a112b07bdd28bb034c85feba9131 refs/tags/06-41-43
baddc289ab1bed70910a242b23a920a8e35cb456 refs/tags/clear_setup
```

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

3.6.1. cp stage.sh prod.sh

3.6.2. nano prod.sh

```
 GNU nano 6.2                                  prod.sh
git checkout prd
git merge stg
timetag=$(date '+%H-%M-%S')
git tag $timetag
git checkout dev
```

3.6.3. git add .

3.6.4. git commit –m «prod script»

3.6.5. bash stage.sh

3.6.6. bash prod.sh

3.6.7. git push —all

3.6.7.1. Внезапно все перестало работать в контейнере отвалился интернет. Пришлось вручную создавать сетевой мост для контейнера

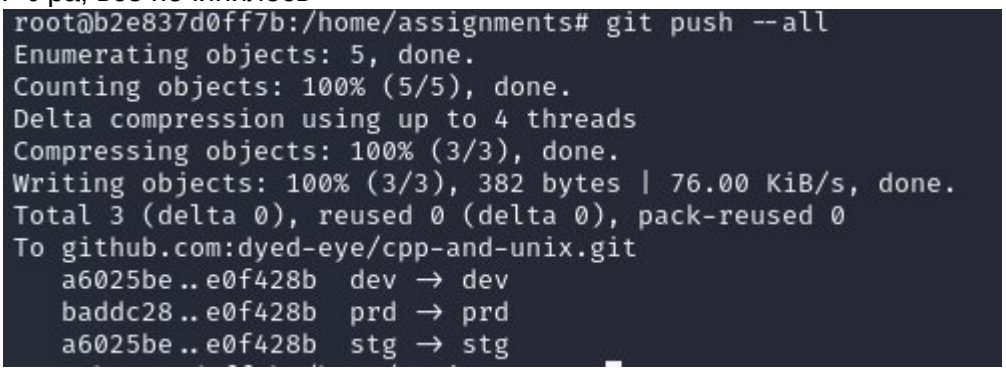
3.6.7.2. exit

3.6.7.3. sudo docker network create —driver bridge common

3.6.7.4. sudo docker run –it —network common logger

3.6.7.5. Повторяем все пункты организации гита

3.6.8. Ура, все починилось



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.**