```
File Edit View Search Terminal Help

gnome-initial-setup:x:977:975::/run/gnome-initial-setup/:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin

rngd:x:976:974:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin

tcpdump:x:72:72::/:/sbin/nologin

user:x:1000:1000:user:/home/user:/bin/bash

vboxadd:x:975:1::/var/run/vboxadd:/bin/false

[user@centos8 ~]$ wc -l /etc/passwd

48 /etc/passwd

[user@centos8 ~]$
```

Мы с вами уже выяснили, что Linux – система многопользовательская. Если с помощью команды wc посчитать количество строк в файле /etc/passwd - cat /etc/passwd; wc -l /etc/passwd, где перечислены все пользователи, мы увидим, что сейчас в системе 48 пользователей. Среди них есть наш пользователь user, суперпользователь root, а остальные, которых создавали не мы, а система, считаются сервисными пользователями.

```
Elik Edit View Search Terminal Help

[user@centos8 ~]$ sudo useradd user2

[sudo] password for user:

[user@centos8 ~]$ sudo passwd user2

Changing password for user user2.

New password:

BAD PASSWORD: The password is a palindrome

Retype new password:

passwd: all authentication tokens updated successfully.

[user@centos8 ~]$ ■
```

Нам сегодня понадобится ещё один пользователь, поэтому создадим его с помощью команды sudo useradd user2, введём наш пароль, а потом с помощью sudo passwd user2 зададим пароль для второго пользователя.

Начнём с команды su. Она позволяет залогиниться каким-то пользователем или запускать команды от имени другого пользователя. При этом нужно знать пароль этого другого пользователя. Это бывает нужно, когда у нашего пользователя нет нужных прав, либо когда нам нужно запустить какой-то процесс от имени другого пользователя, например, в целях безопасности.

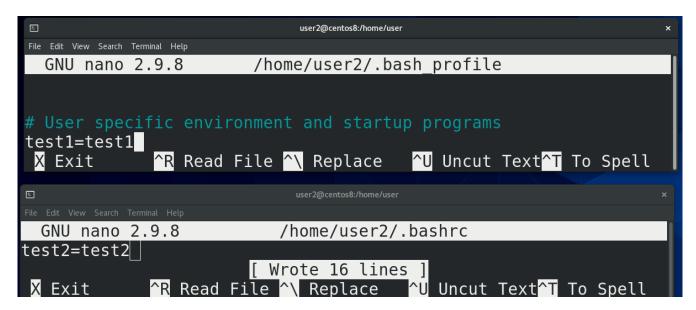
```
user@centos8:~ x
File Edit View Search Terminal Help

[user@centos8 ~]$ cd /home/user2
bash: cd: /home/user2: Permission denied

[user@centos8 ~]$ su user2
Password:
[user2@centos8 user]$ cd /home/user2
[user2@centos8 ~]$ pwd
/home/user2
[user2@centos8 ~]$ exit
exit
[user@centos8 ~]$
```

Например, сейчас мой пользователь не может зайти в директорию /home/user2 - cd /home/user2, потому что у него недостаточно прав. Я могу написать su user2, ввести пароль пользователя user2 и стать этим самым вторым пользователем, как видно в начале строки. А дальше смогу зайти в нужную директорию - cd /home/user2. Чтобы вернуться к моему пользователю, я могу написать exit, либо нажать Ctrl+d. Если написать просто su, либо su root, и ввести пароль рута, можно работать от пользователя root.

Но помните мы разбирали файлы  $\sim$ /.bash\_profile и  $\sim$ /.bashrc ? Мы писали там переменные и алиасы, и, в случае с  $\sim$ /.bash\_profile, нам нужно было перезалогиниться, а в случае с  $\sim$ /.bashrc нам достаточно было просто запустить новый эмулятор терминала. То есть  $\sim$ /.bash\_profile это файл настроек для login shell, а  $\sim$ /.bashrc для nonlogin shell.



Давайте сделаем вот что. У пользователя user2 - su user2 - в файле ~/.bash\_profile - nano /home/user2/.bash\_profile - создадим переменную test1 paвную test1 - test1=test1, а в файле ~/.bashrc - nano /home/user2/.bashrc - переменную test2 paвную test2 - test2=test2.

```
© user2@centos8:/home/user ×

File Edit View Search Terminal Help

[user2@centos8 ~]$ su user2

Password:
[user2@centos8 user]$ echo $test1

[user2@centos8 user]$ echo $test2

test2
[user2@centos8 user]$ echo $PATH
/home/user2/.local/bin:/home/user2/bin:/home/user/.local/bin:/home/user/bin:/home/user/.local/bin:/home/user/bin:/usr/local/sbin:/usr/bin:/usr/sbin
[user2@centos8 user]$ pwd
/home/user
[user2@centos8 user]$ pwd
/home/user
[user2@centos8 user]$
```

Теперь выйдем - ctrl+d - заново зайдём - su user2 - и посмотрим, как обстоят дела с переменными - echo \$test1; echo \$test2. Как видите, сработала настройка только из ~/.bashrc, то есть non-login shell. Это означает, что переменные окружения не прочитались с ~/.bash\_profile (login shell) пользователя user2. На самом деле, все переменные остались от предыдущего пользователя. Допустим, если посмотреть переменную РАТН, можно увидеть пути к директориям /home/user/bin, а это домашняя директория первого пользователя. Также, если писать su user2, можно заметить, что директория, в которой мы находимся, не меняется - pwd.

```
Elik View Search Terminal Help

[user@centos8 ~]$ su - user2

Password:

[user2@centos8 ~]$ echo $test1

test1

[user2@centos8 ~]$ echo $test2

test2

[user2@centos8 ~]$ pwd

/home/user2

[user2@centos8 ~]$ |
```

Зачастую нужно, чтобы при логине за другого пользователя поменялось окружение, то есть, чтобы применились настройки из ~/.bash\_profile нужного пользователя. Для этого после su следует писать дефис - su – user2. Теперь у меня есть обе переменные - echo \$test1; echo \$test2, то есть считался файл ~/.bash\_profile, а значит это был login shell. Также стоит заметить, что при su с дефисом поменялась и директория — раньше мы находились в домашней директории пользователя user, а после "su -" меняется текущая директория — pwd - /home/user2.

```
E user2@centos8.~ x

File Edit View Search Terminal Help

[user2@centos8 ~]$ su user2

Password:
[user2@centos8 user]$ echo $0

bash

[user2@centos8 user]$ exit
[user@centos8 ~]$ su - user2

Password:
[user2@centos8 ~]$ echo $0

-bash

[user2@centos8 ~]$ echo $PATH
/home/user2/.local/bin:/home/user2/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[user2@centos8 ~]$ ■
```

Кстати, чтобы понять, текущий shell – login или non-login, не обязательно выдумывать каждый раз какие-то проверки с переменными, достаточно проверить значение переменной \$0 - echo \$0. При non-login shell значение будет просто bash, а при login shell "-bash". И если проверить ту же переменную РАТН - echo \$РАТН, можно увидеть, что теперь здесь нет пути /home/user/bin, то есть переменные окружения не передались от предыдущего пользователя, а появились как следует.

Так вот, подводя итоги. Когда вы вводите свой логин и пароль, будь то удалённо с помощью ssh, либо локально, зайдя в виртуальный терминал, либо залогинившись в графической оболочке - запускается оболочка со входом - login shell. Это оболочка с авторизацией, она считывает настройки — те же переменные, алиасы, функции сначала с файла /etc/profile, где написано смотреть на файлы в директории /etc/profile.d/ и в файл ~/.bash\_profile в домашней директории пользователя. Там также написано смотреть в файл ~/.bashrc в домашней директории пользователя, в котором также написано смотреть в файл /etc/bashrc.

Когда же вы запускаете программу эмулятор терминала, то там нет логина, вы без логина и пароля можете вводить команды - оболочка без входа - non-login shell. В случае с non-login shell сначала считывается файл ~/.bashrc в домашней директории пользователя, затем считывается файл /etc/bashrc, который в свою очередь ссылается на файлы в /etc/profile.d/. Но приведённая схема может отличаться на других дистрибутивах.

Кроме этого, оболочки делятся на interactive и non-interactive. Если вы логинитесь и запускаете команды - это interactive login shell. Если вы работаете в эмуляторе терминала - это interactive non-login shell. Когда работают скрипты, они обычно запускаются без всякого логина - тут уже non-interactive non-login shell. Да, когда вы запускаете эмулятор терминала, в этом эмуляторе запускается bash и запускает ~/.bashrc. Сами по себе ~/.bashrc и ~/.bash\_profile - это просто набор команд в одном файле. То есть это скрипты. Когда же вы логинитесь, не важно каким образом, тоже запускается bash, при этом он запускает ~/.bash\_profile.

```
ser@centos8:~

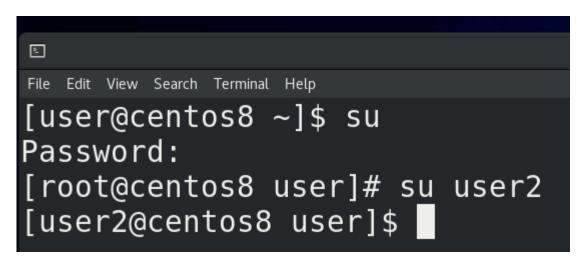
File Edit View Search Terminal Help

[user@centos8 ~]$ su user2 -c "touch file"

Password:
touch: cannot touch 'file': Permission denied
[user@centos8 ~]$ su - user2 -c "touch file"

Password:
[user@centos8 ~]$ su - -c "nano /etc/passwd"
```

Ладно, со сменой пользователя разобрались. Мы еще говорили, что ѕи позволяет запускать команды от имени другого пользователя. Для этого используется ключ -с. Например - ѕи user2 -с "touch file"; ѕи — user2 -с "touch file". Объясните в комментариях, почему первая команда завершилась с ошибкой, а вторая без? Ну и зачастую, ѕи используют чтобы работать от имени гоот пользователя, для примера запустим папо от рута - ѕи - -с "nano /etc/passwd" - и теперь мы можем редактировать файл /etc/passwd и в целом можем делать всё что угодно.



Кстати, если запускать su от имени рута - su; su user2 - то никаких паролей не потребуется, root может логиниться кем угодно.