

User Environment

Accounts, Users and Groups:

Identifying the Current User:

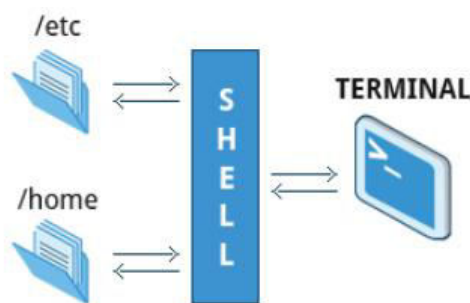
1. As you know, Linux is a multi-user operating system, meaning more than one user can log on at the same time.
 - To identify the current user, type **whoami**.
 - To list the currently logged-on users, type **who**.
2. Giving who the **-a** option will give more detailed information.

```
student@ubuntu:~$ whoami
student
student@ubuntu:~$ who
student  :0                2021-06-04 05:51 (:0)
student pts/2             2021-06-04 06:30 (192.168.235.1)
student pts/3             2021-06-04 06:31 (127.0.0.1)
student@ubuntu:~$ |
```

Identifying the Current User

User Startup Files:

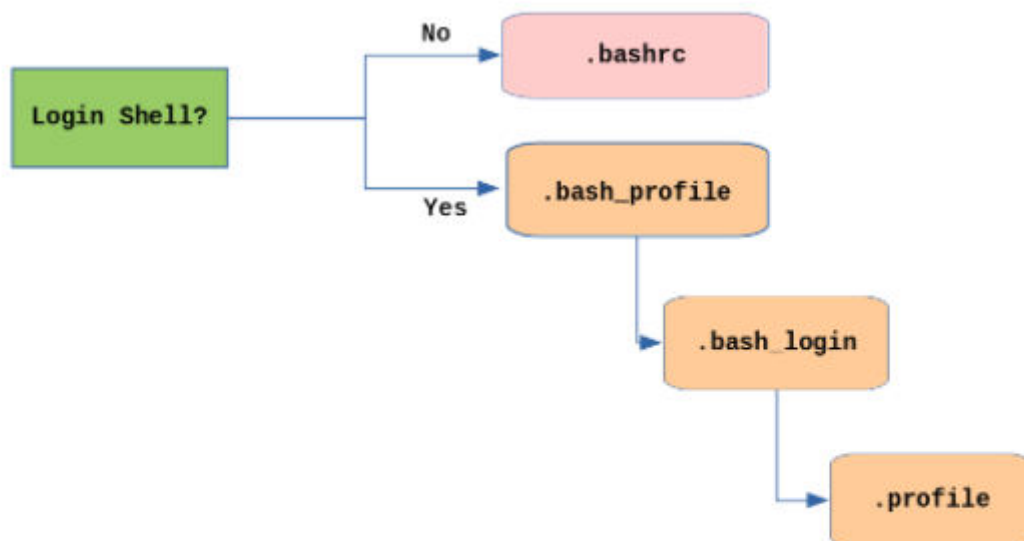
1. In Linux, the command shell program (generally bash) uses one or more startup files to configure the user environment.
2. Files in the **/etc** directory define global settings for all users, while initialization files in the user's home directory can include and/or override the global settings.
3. The startup files can do anything the user would like to do in every command shell, such as:
 - Customizing the prompt
 - Defining command line shortcuts and aliases
 - Setting the default text editor
 - Setting the path for where to find executable programs



User Startup Files

Order of the Startup Files:

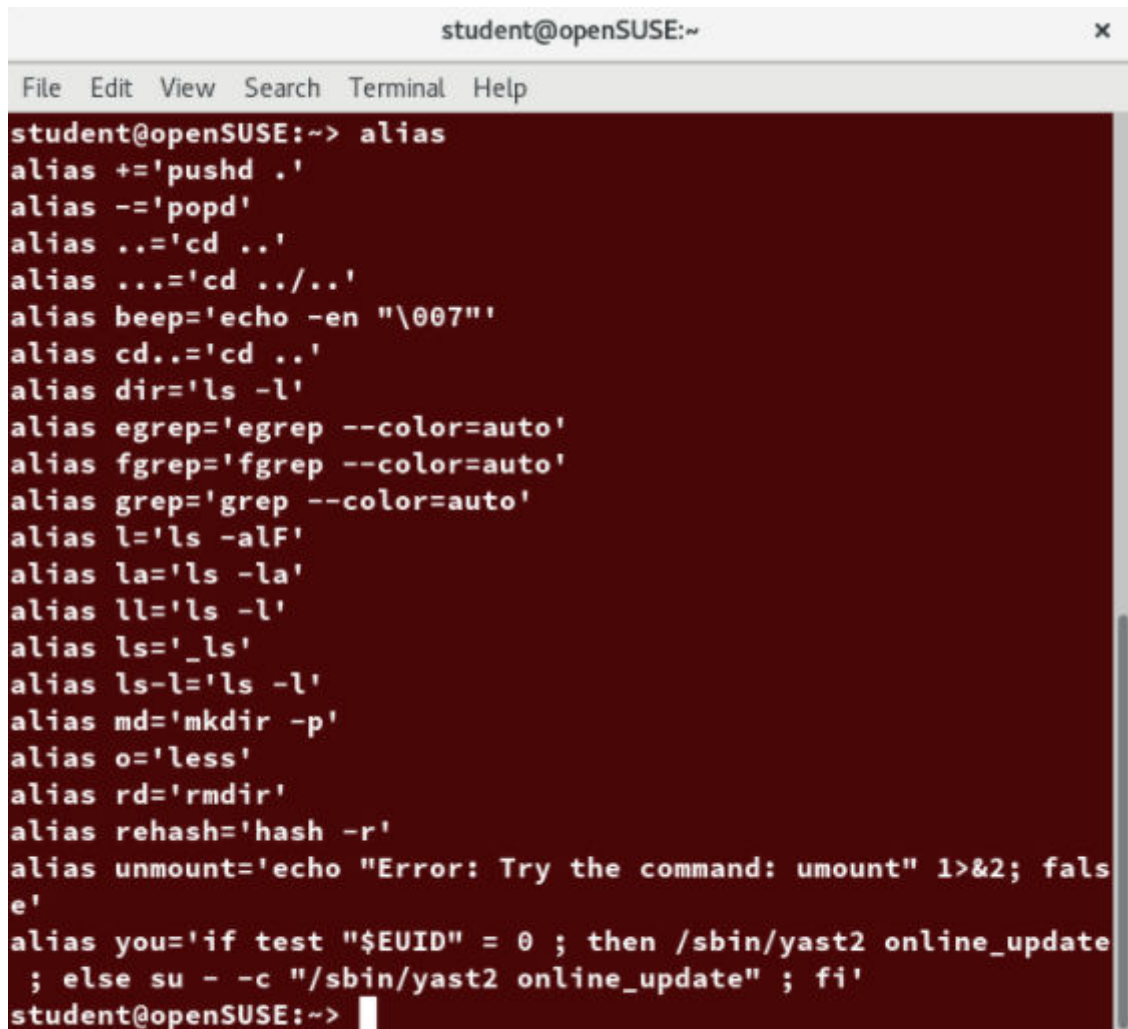
1. The standard prescription is that when you first login to Linux, `/etc/profile` is read and evaluated, after which the following files are searched (if they exist) in the listed order:
 - `~/.bash_profile`
 - `~/.bash_login`
 - `~/.profile`
2. where in the above files `~/.` denotes the user's home directory.
3. The Linux login shell evaluates whatever startup file that it comes across first and ignores the rest.
4. This means that if it finds `~/.bash_profile`, it ignores `~/.bash_login` and `~/.profile`. Different distributions may use different startup files.
5. However, every time you create a new shell, or terminal window, etc., you do not perform a full system login; only a file named `~/.bashrc` file is read and evaluated.
6. Although this file is not read and evaluated along with the login shell, most distributions and/or users include the `~/.bashrc` file from within one of the three user-owned startup files.
7. Most commonly, users only fiddle with `~/.bashrc`, as it is invoked every time a new command line shell initiates, or another program is launched from a terminal window, while the other files are read and executed only when the user first logs onto the system.
8. Recent distributions sometimes do not even have `.bash_profile` and/or `.bash_login`, and some just do little more than include `.bashrc`.



Order of the Startup Files

Creating Aliases:

1. You can create customized commands or modify the behaviour of already existing ones by creating aliases.
2. Most often, these aliases are placed in your `~/.bashrc` file so they are available to any command shells you create.
3. `unalias` removes an alias. Typing `alias` with no arguments will list currently defined aliases.
4. Please note there should not be any spaces on either side of the equal sign and the alias definition needs to be placed within either single or double quotes if it contains any spaces.



```
student@openSUSE:~  
File Edit View Search Terminal Help  
student@openSUSE:~> alias  
alias += 'pushd .'  
alias -= 'popd'  
alias .. = 'cd ..'  
alias ... = 'cd ../../..'  
alias beep = 'echo -en "\007"'  
alias cd.. = 'cd ..'  
alias dir = 'ls -l'  
alias egrep = 'egrep --color=auto'  
alias fgrep = 'fgrep --color=auto'  
alias grep = 'grep --color=auto'  
alias l = 'ls -alF'  
alias la = 'ls -la'  
alias ll = 'ls -l'  
alias ls = '_ls'  
alias ls-l = 'ls -l'  
alias md = 'mkdir -p'  
alias o = 'less'  
alias rd = 'rmdir'  
alias rehash = 'hash -r'  
alias umount = 'echo "Error: Try the command: umount" 1>&2; false'  
alias you = 'if test "$EUID" = 0 ; then /sbin/yast2 online_update  
; else su - -c "/sbin/yast2 online_update" ; fi'  
student@openSUSE:~>
```

Creating Aliases

Basics of Users and Groups:

1. All Linux users are assigned a unique user ID (uid), which is just an integer; normal users start with a uid of 1000 or greater.
2. Linux uses groups for organizing users. Groups are collections of accounts with certain shared permissions.

3. Control of group membership is administered through the `/etc/group` file, which shows a list of groups and their members.
4. By default, every user belongs to a default or primary group.
5. When a user logs in, the group membership is set for their primary group and all the members enjoy the same level of access and privilege. Permissions on various files and directories can be modified at the group level.
6. Users also have one or more group IDs (`gid`), including a default one which is the same as the user ID.
7. These numbers are associated with names through the files `/etc/passwd` and `/etc/group`.
8. Groups are used to establish a set of users who have common interests for the purposes of access rights, privileges, and security considerations.
9. Access rights to files (and devices) are granted on the basis of the user and the group they belong to.
10. For example, `/etc/passwd` might contain `george:x:1002:1002:George Metesky:/home/george:/bin/bash` and `/etc/group` might contain `george:x:1002`.

Adding and Removing Users:

1. Adding a new user is done with **useradd** and removing an existing user is done with **userdel**.
2. In the simplest form, an account for the new user `bjmoose` would be done with the below example
`sudo useradd bjmoose`
3. which, by default, sets the home directory to `/home/bjmoose`, populates it with some basic files (copied from `/etc/skel`) and adds a line to `/etc/passwd` such as the below and sets the default shell to `/bin/bash`.
`bjmoose:x:1002:1002:./home/bjmoose:/bin/bash`
4. Removing a user account is as easy as typing **userdel bjmoose**. However, this will leave the `/home/bjmoose` directory intact. This might be useful if it is a temporary inactivation.
5. To remove the home directory while removing the account one needs to use the **-r** option to **userdel**.
6. Typing `id` with no argument gives information about the current user, as in the below. If given the name of another user as an argument, `id` will report information about that other user.
Ex: `id`
Output: `uid=1002(bjmoose) gid=1002(bjmoose) groups=106(fuse),1002(bjmoose)`

```
File Edit View Search Terminal Help
c7:/home/coop>sudo useradd -s /bin/csh -m -k/etc/skel -c "Bullwinkle J Moose" bmoose
c7:/home/coop>ls -la /home/bmoose
ls: cannot open directory /home/bmoose: Permission denied
c7:/home/coop>sudo ls -la /home/bmoose
total 28
drwx----- 3 bmoose bmoose 4096 Dec 14 09:57 .
drwxr-xr-x 3 root root 4096 Dec 14 09:57 ..
-rw-r--r-- 1 bmoose bmoose 18 Oct 17 05:05 .bash_logout
-rw-r--r-- 1 bmoose bmoose 193 Oct 17 05:05 .bash_profile
-rw-r--r-- 1 bmoose bmoose 231 Oct 17 05:05 .bashrc
-rw-r--r-- 1 bmoose bmoose 334 Oct 7 2015 .emacs
drwxr-xr-x 4 bmoose bmoose 4096 Jan 26 2014 .mozilla
c7:/home/coop>sudo passwd bmoose
Changing password for user bmoose.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
c7:/home/coop>su bmoose
Password:
Attempting to create directory /home/bmoose/perl5
[bmoose@c7 coop]$ exit
exit
c7:/home/coop>sudo userdel -r bmoose
c7:/home/coop>ls /home
coop linux1
c7:/home/coop>
```

Adding and Removing Users

Adding and Removing Groups:

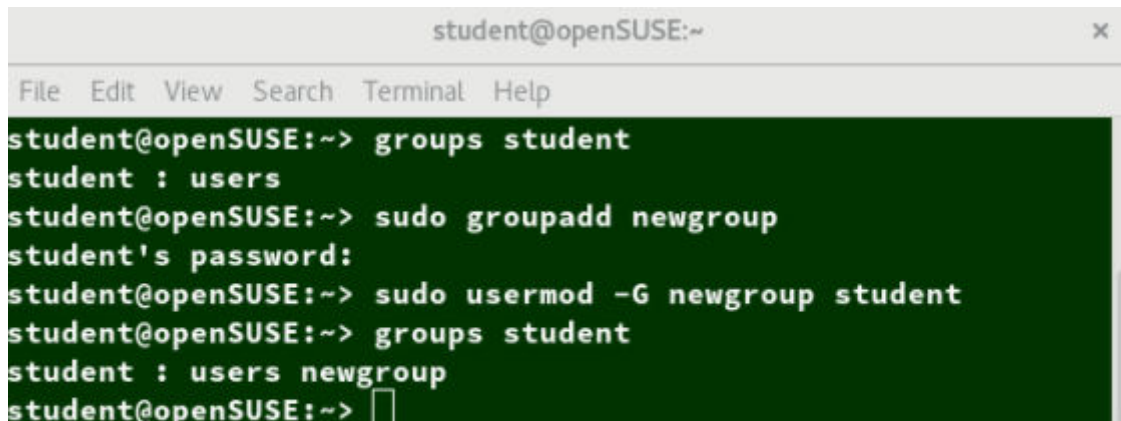
1. Adding a new group is done with **groupadd**:
`sudo /usr/sbin/groupadd anewgroup`
2. The group can be removed with **groupdel**:
`sudo /usr/sbin/groupdel anewgroup`
3. Adding a user to an already existing group is done with **usermod**.
4. For example, you would first look at what groups the user already belongs to:
Ex: `groups rjsquirrel`
O/P: `bjmoose : rjsquirrel`
5. After the getting group details of the user from the above command add the user to the new group using the below command
Ex: `sudo /usr/sbin/usermod -a -G anewgroup rjsquirrel`
Ex: `groups rjsquirrel`
O/P: `rjsquirrel: rjsquirrel anewgroup`
6. These utilities update `/etc/group` as necessary.
7. Make sure to use the **-a** option, for append, so as to avoid removing already existing groups.
8. **groupmod** can be used to change group properties, such as the Group ID (gid) with the **-g** option or its name with then **-n** option.

9. Removing a user from the group is somewhat trickier. The **-G** option to **usermod** must give a complete list of groups. Thus, if you do:

Ex: `sudo /usr/sbin/usermod -G rjsquirrel rjsquirrel`

Ex: `groups rjsquirrel`

O/P: `rjsquirrel : rjsquirrel`

A terminal window titled 'student@openSUSE:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
student@openSUSE:~> groups student
student : users
student@openSUSE:~> sudo groupadd newgroup
student's password:
student@openSUSE:~> sudo usermod -G newgroup student
student@openSUSE:~> groups student
student : users newgroup
student@openSUSE:~> 
```

Adding and Removing Groups

The root Account:

1. The root account is very powerful and has full access to the system.
2. Other operating systems often call this the administrator account; in Linux, it is often called the **superuser account**.
3. You must be extremely cautious before granting full root access to a user; it is rarely, if ever, justified. External attacks often consist of tricks used to elevate to the root account.
4. However, you can use **sudo** to assign more limited privileges to user accounts:
 - Only on a temporary basis
 - Only for a specific subset of commands.

su and sudo:

1. When assigning elevated privileges, you can use the command **su** (switch or substitute user) to launch a new shell running as another user (you must type the password of the user you are becoming).
2. Most often, this other user is **root**, and the new shell allows the use of elevated privileges until it is exited.
3. It is almost always a bad (dangerous for both security and stability) practice to use **su** to become **root**.
4. Resulting errors can include deletion of vital files from the system and security breaches.
5. Granting privileges using **sudo** is less dangerous and is preferred. By default, **sudo** must be enabled on a per-user basis.
6. However, some distributions (such as Ubuntu) enable it by default for at least one main user, or give this as an installation option.