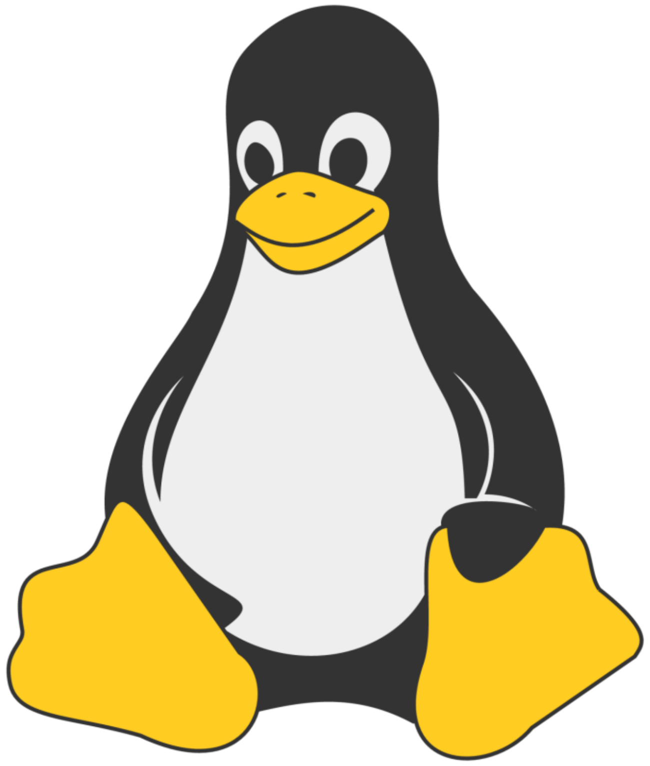


Module 05

Users and Groups

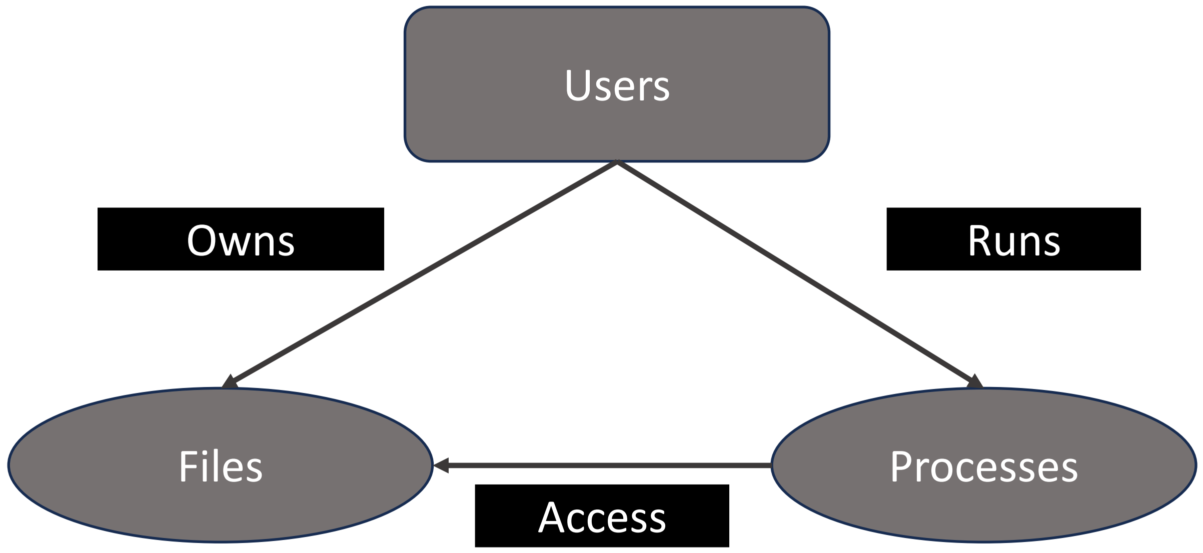
Course Materials

Course materials are in GitHub

https://github.com/lalindrait/linux-administration-1

Users

- A user account is used to provide security boundaries between different people and programs that can run commands
- Users have user names to identify them to human users and make them easier to work with.
- Internally, the system distinguishes user accounts by the unique identification number assigned to them, the user ID or UID.

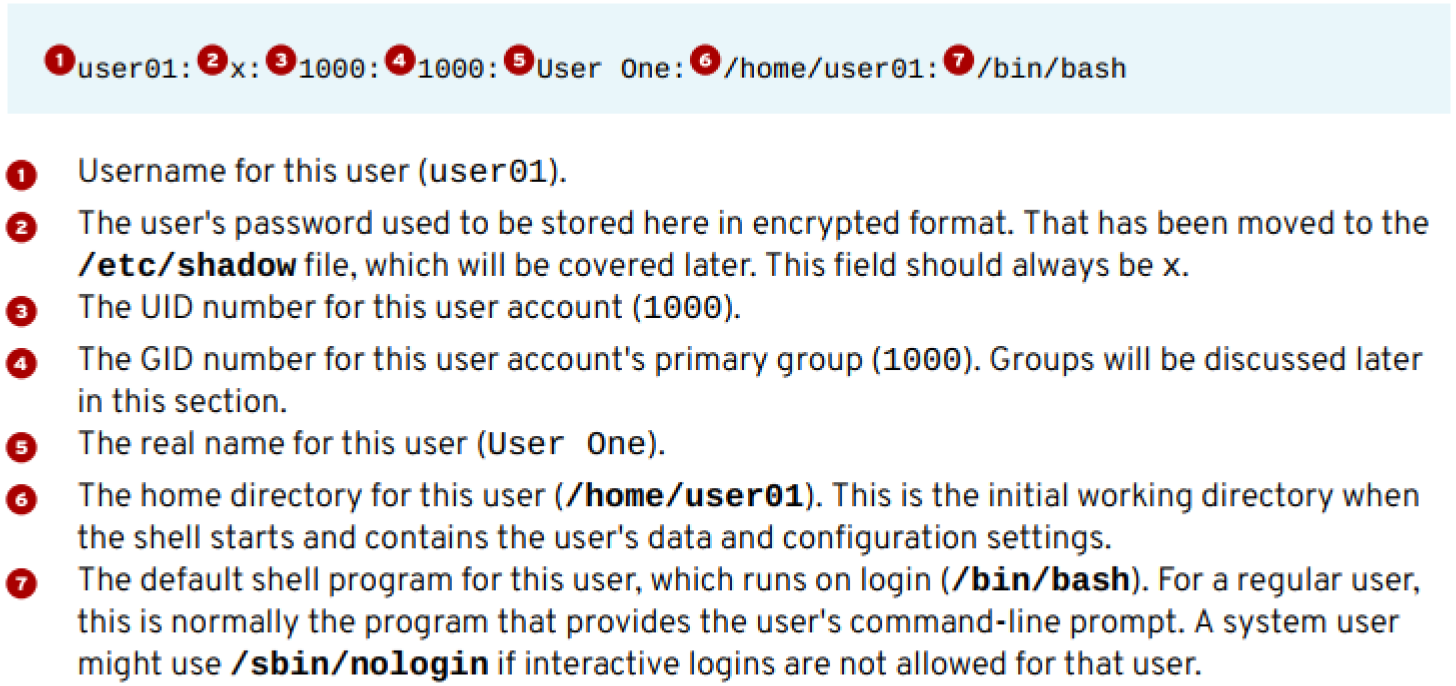


- There are three main types of user account: the superuser, system users, and regular users.
  - The **superuser account** is for administration of the system. The name of the superuser is root and the account has UID 0. The superuser has full access to the system.
  - The system has **system user** accounts which are used by processes that provide supporting services. These processes, or daemons, usually do not need to run as the superuser. They are assigned non-privileged accounts. Users do not interactively log in using a system user account.
  - Most users have **regular user accounts** which they use for their day-to-day work. Like system users, regular users have limited access to the system.

/etc/passwd

- User information is stored in Linux in /etc/passwd file

- It is divided in to 7 fields separated by colons



```
❶user01:❷x:❸1000:❹1000:❺User One:❻/home/user01:❼/bin/bash
```

❶ Username for this user (`user01`).

❷ The user's password used to be stored here in encrypted format. That has been moved to the **`/etc/shadow`** file, which will be covered later. This field should always be `x`.

❸ The UID number for this user account (`1000`).

❹ The GID number for this user account's primary group (`1000`). Groups will be discussed later in this section.

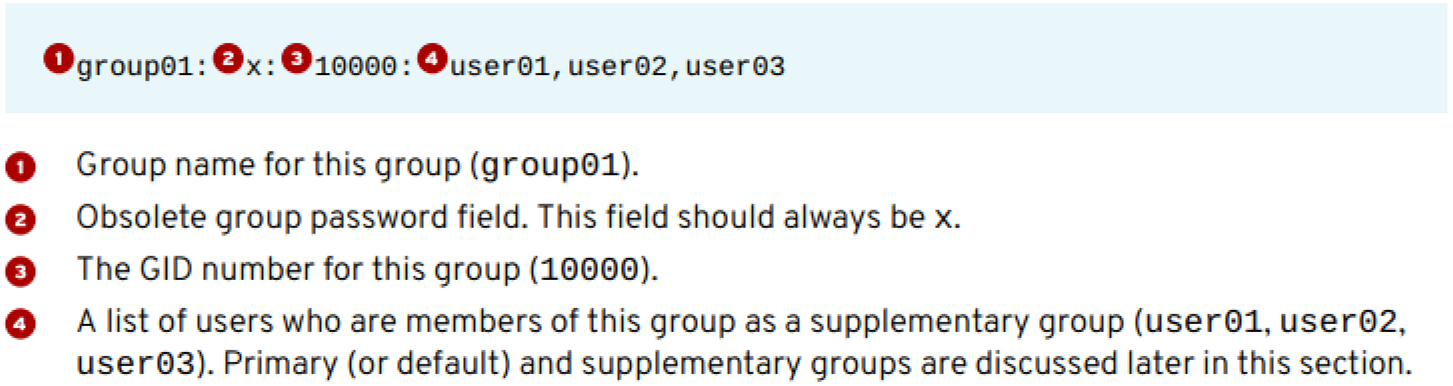
❺ The real name for this user (`User One`).

❻ The home directory for this user (**`/home/user01`**). This is the initial working directory when the shell starts and contains the user's data and configuration settings.

❼ The default shell program for this user, which runs on login (**`/bin/bash`**). For a regular user, this is normally the program that provides the user's command-line prompt. A system user might use **`/sbin/nologin`** if interactive logins are not allowed for that user.

Groups & /etc/group

- A group is a collection of users that need to share access to files and other system resources.
- Groups can be used to grant access to files to a set of users instead of just a single user.
- Like users, groups have group names to make them easier to work with. Internally, the system distinguishes groups by the unique identification number assigned to them, the group ID or GID.
- The mapping of group names to GIDs is defined in databases of group account information.
- By default, systems use the /etc/group file to store information about local groups.



```
❶group01:❷x:❸10000:❹user01,user02,user03
```

❶ Group name for this group (`group01`).

❷ Obsolete group password field. This field should always be `x`.

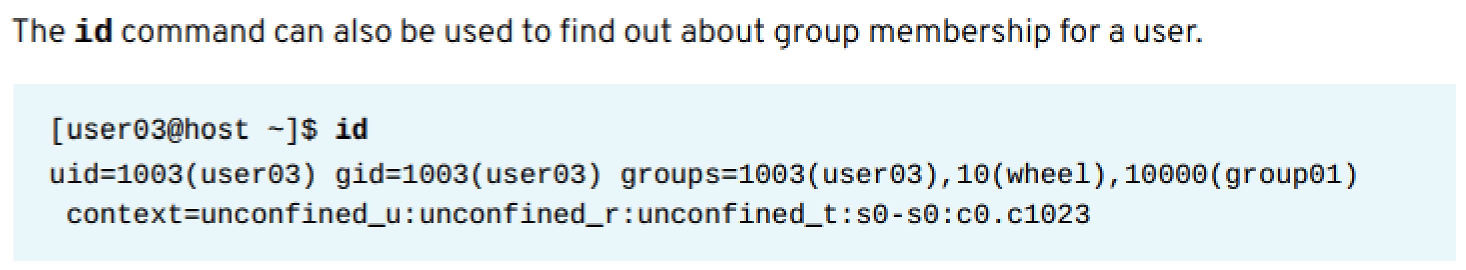
❸ The GID number for this group (`10000`).

❹ A list of users who are members of this group as a supplementary group (`user01`, `user02`, `user03`). Primary (or default) and supplementary groups are discussed later in this section.

Primary Groups and Secondary Groups

- Every user has exactly one primary group.
- For local users, this is the group listed by GID number in the /etc/passwd file.
- By default, this is the group that will own new files created by the user.
- Normally, when you create a new regular user, a new group with the same name as that user is created.
- That group is used as the primary group for the new user, and that user is the only member of this User Private Group.

- Users may also have supplementary groups.
- Membership in supplementary groups is determined by the /etc/group file.
- Users are granted access to files based on whether any of their groups have access.
- It doesn't matter if the group or groups that have access are primary or supplementary for the user.



The **id** command can also be used to find out about group membership for a user.

```
[user03@host ~]$ id
uid=1003(user03) gid=1003(user03) groups=1003(user03),10(wheel),10000(group01)
 context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Superuser

- Most operating systems have some sort of superuser, a user that has all power over the system.
- In Linux this is the root user.
- This user has the power to override normal privileges on the file system, and is used to manage and administer the system.
- To perform tasks such as installing or removing software and to manage system files and directories, users must escalate their privileges to the root user.

```
[user01@host ~]$ su - user02
Password:
[user02@host ~]$
```

If you omit the user name, the **su** or **su -** command attempts to switch to `root` by default.

```
[user01@host ~]$ su -
Password:
[root@host ~]#
```

Login as root user only when needed

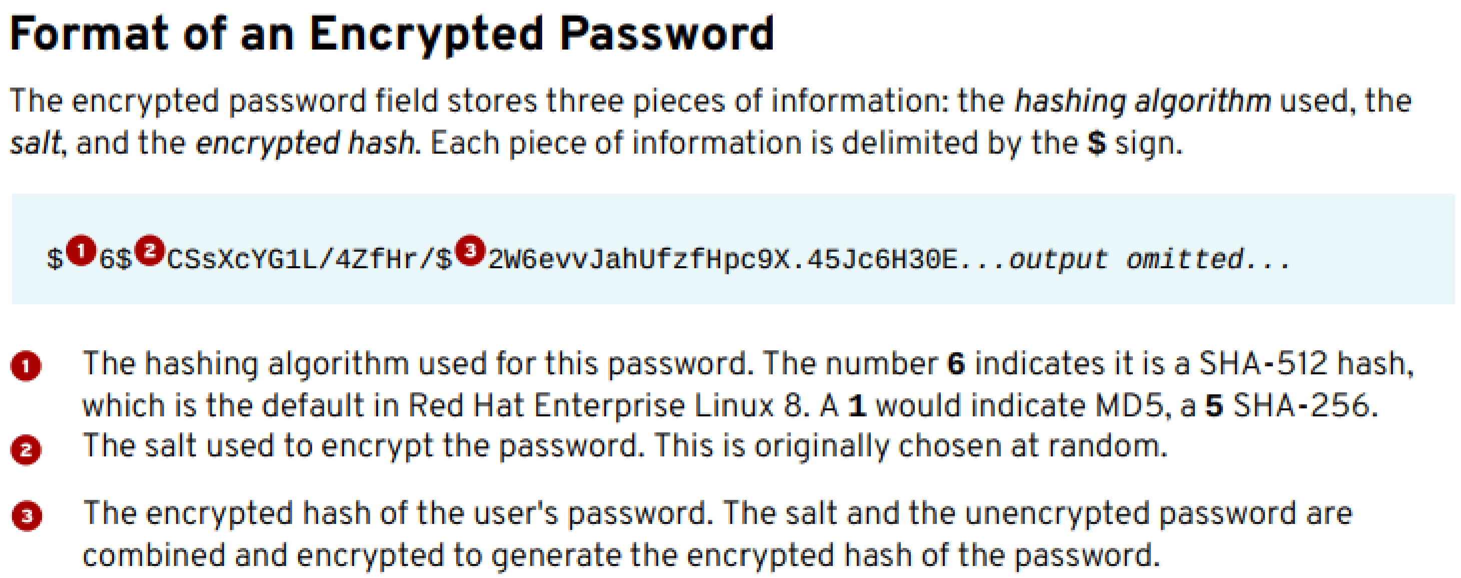
- *This unlimited privilege, however, comes with responsibility.*
- *The root user has unlimited power to damage the system: remove files and directories, remove user accounts, add back doors, and so on. If the root user's account is compromised, someone else would have administrative control of the system.*
- ***You should always log in as a normal user and escalate privileges to root only when needed.***

UID – User ID

- *UID 0* is always assigned to the superuser account, **root**.
- *UID 1-200* is a range of "system users" assigned statically to system processes by Red Hat.
- *UID 201-999* is a range of "system users" used by system processes that do not own files on the file system.
    - They are typically assigned dynamically from the available pool when the software that needs them is installed.
    - Programs run as these "unprivileged" system users in order to limit their access to only the resources they need to function.
- *UID 1000+* is the range available for assignment to regular users.

User Passwords

- At one time, encrypted passwords were stored in the world-readable /etc/passwd file.
- This was thought to be reasonably secure until dictionary attacks on encrypted passwords became common.
- At that point, the encrypted passwords were moved to a separate /etc/shadow file which is readable only by root.
- This new file also allowed password aging and expiration features to be implemented.



Format of an Encrypted Password

The encrypted password field stores three pieces of information: the *hashing algorithm* used, the *salt*, and the *encrypted hash*. Each piece of information is delimited by the **$** sign.

```
$❶6$❷CSsXcYG1L/4ZfHr/$❸2W6evvJahUfzfHpc9X.45Jc6H30E...output omitted...
```

❶ The hashing algorithm used for this password. The number **6** indicates it is a SHA-512 hash, which is the default in Red Hat Enterprise Linux 8. A **1** would indicate MD5, a **5** SHA-256.

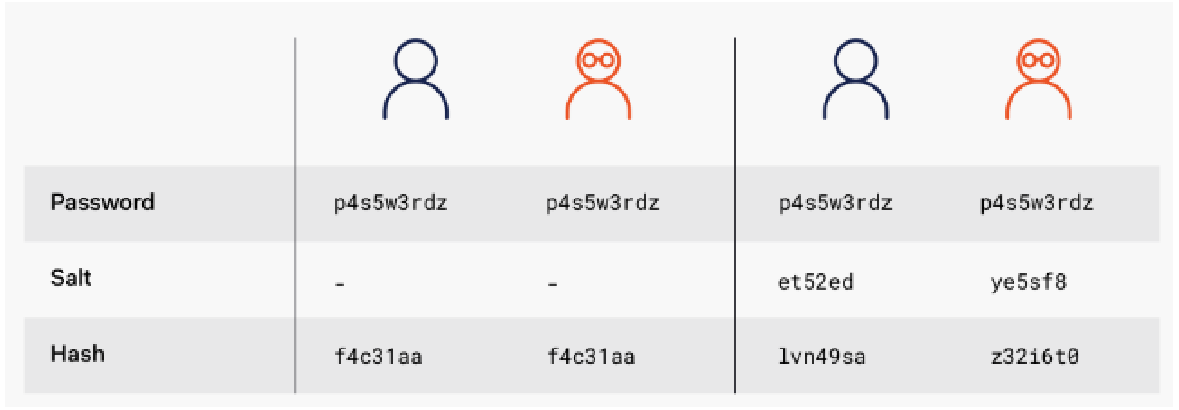
❷ The salt used to encrypt the password. This is originally chosen at random.

❸ The encrypted hash of the user's password. The salt and the unencrypted password are combined and encrypted to generate the encrypted hash of the password.

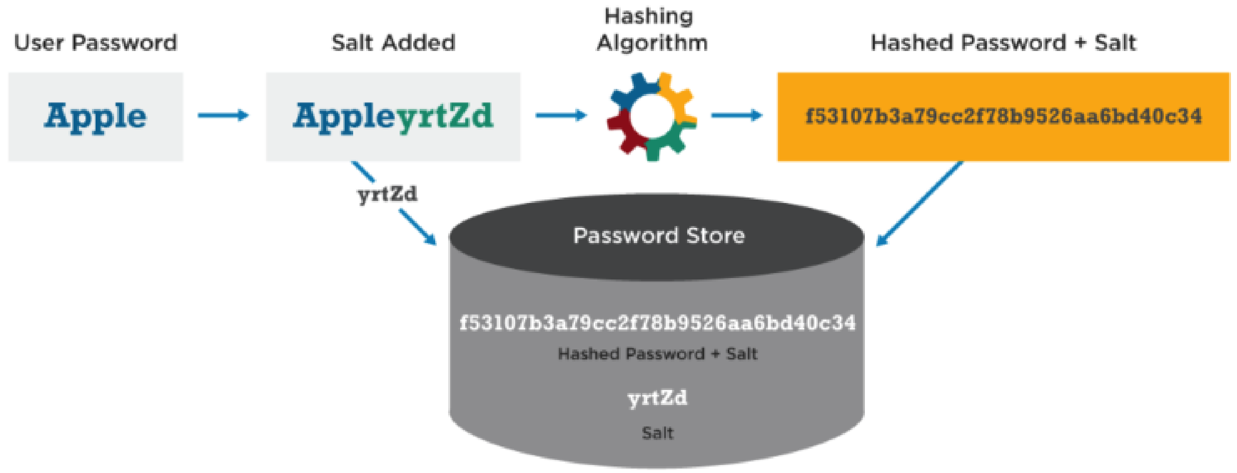
Salting

- In hashing same file/word gives you the same hash
- If you know a hash you can derive the original value
- This is a problem when storing password in a database
- Password salting is one of the most secure ways to protect passwords

- Salting is a cryptographic technique that adds a random string to a password before it's hashed.
- Salting creates unique passwords, even if two users choose the same password.
- It also makes it difficult for an attacker to know the original plaintext without having access to both the salt and the hash.



Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0



/etc/shadow

```
❶user03:❷$6$CSsX...output omitted...:❸17933:❹0:❺99999:❻7:❼2:❽18113:❾
```

❶ Username of the account this password belongs to.

❷ The *encrypted password* of the user. The format of encrypted passwords is discussed later in this section.

❸ The day on which the password was last changed. This is set in days since 1970-01-01, and is calculated in the UTC time zone.

❹ The minimum number of days that have to elapse since the last password change before the user can change it again.

❺ The maximum number of days that can pass without a password change before the password expires. An empty field means it does not expire based on time since the last change.

❻ Warning period. The user will be warned about an expiring password when they login for this number of days before the deadline.

❼ Inactivity period. Once the password has expired, it will still be accepted for login for this many days. After this period has elapsed, the account will be locked.

❽ The day on which the password expires. This is set in days since 1970-01-01, and is calculated in the UTC time zone. An empty field means it does not expire on a particular date.

❾ The last field is usually empty and is reserved for future use.

Password Verification

- When a user tries to log in, the system looks up the entry for the user in /etc/shadow, combines the salt for the user with the unencrypted password that was typed in, and encrypts them using the hashing algorithm specified.
- If the result matches the encrypted hash, the user typed in the right password. If the result does not match the encrypted hash, the user typed in the wrong password and the login attempt fails.

nologin shell

- The nologin shell acts as a replacement shell for the user accounts not intended to interactively log into the system.
- It is wise from the security standpoint to disable the user account from logging into the system when the user account serves a responsibility that does not require the user to log into the system.
- For example, a mail server may require an account to store mail and a password for the user to authenticate with a mail client used to retrieve mail.
- That user does not need to log directly into the system. A common solution to this situation is to set the user's login shell to /sbin/nologin.
- If the user attempts to log in to the system directly, the nologin shell closes the connection.

User related files

- In any home directory there will be some dot files in the home directory
- These dot or hidden files are used to when users login in or log out of the system to configure the system accordingly
- These files include
    - .bashrc - short for bash read command
    - .bash_profile
    - .bash_logout

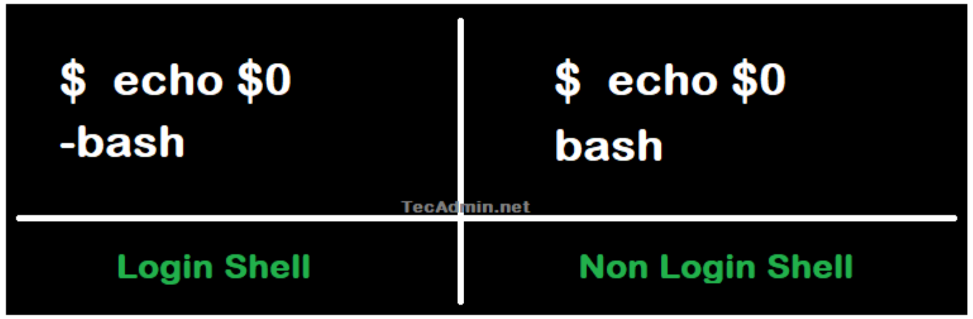
Login shell vs Non-Login shell

Login Shell

- When a user successfully logs in to a Linux system via terminal, SSH, or switches to a user with the “su -” command, a Login shell is created.
- The following scripts are executed by the Login Shell:
  - Login shell invokes /etc/profile
  - /etc/profile invokes scripts in /etc/profile.d/*.sh
  - Then executes users ~/.bash_profile
  - ~/.bash_profile invokes users ~/.bashrc
  - ~/.bashrc invokes /etc/bashrc

Non-Login Shell

- A non-login shell is started by a login shell. For example, a shell that you start from another shell or from a program is a non-login shell.
- A shell that is not used to log in to the system executes the following script to set the shell environment.
  - Non login shell first executes ~/.bashrc
  - Then ~/.bashrc executes /etc/bashrc
  - /etc/bashrc calls the scripts in /etc/profile.d



```
$ echo $0
-bash
```

```
$ echo $0
bash
```

.bashrc

- The .bashrc file is a configuration file for the Bash shell.
- The file consists of commands, functions, aliases, and scripts that execute every time a bash session starts.
- The file executes automatically in both interactive and non-interactive non-login shells.
- When running a non-login shell (sub shell), the primary read configuration file is the /etc/bashrc.
  - The file contains system-wide configurations for non-login shells.

<u>Interactive vs Non-Interactive Shells</u>

- An interactive shell reads user input typed on a keyboard. Commands run in the current shell and the system waits for further instructions from the user.
  - A Bash shell is an interactive shell.
- A non-interactive shell executes commands read from a file in a new subshell.
  - The shell executes commands from the file and exits. Init, startup, and executed shell scripts start non-interactive shells.

.bash_profile

- The .bash_profile file is a hidden script file with custom configurations for a user terminal session.
- The file automatically executes in bash interactive login shells.
- When running an interactive login shell, the system reads the following configuration file first:
    - /etc/profile - Stores global configurations for login shells. The configurations apply to all users.
    - Next, the Bash shell searches for specific user configuration files in the following order:
        - ~/.bash_profile
        - ~/.bash_login
        - ~/.profile
    - The first found file is read and executed.