



## Linux Essentials

Dr. Hatem Yousry

# Agenda



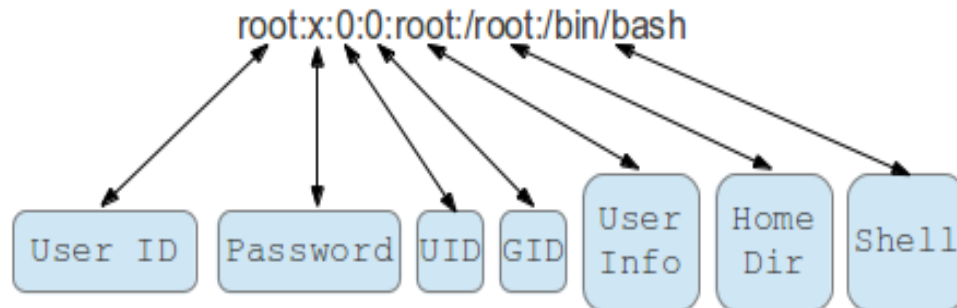
- **Directory Commands.**
- **Managing Users and Groups.**

```
test@test-VirtualBox: ~/Desktop/files
File Edit View Search Terminal Help
test@test-VirtualBox:~/Desktop/files$ grep phoenixNAP sample
```

*string to match filename*

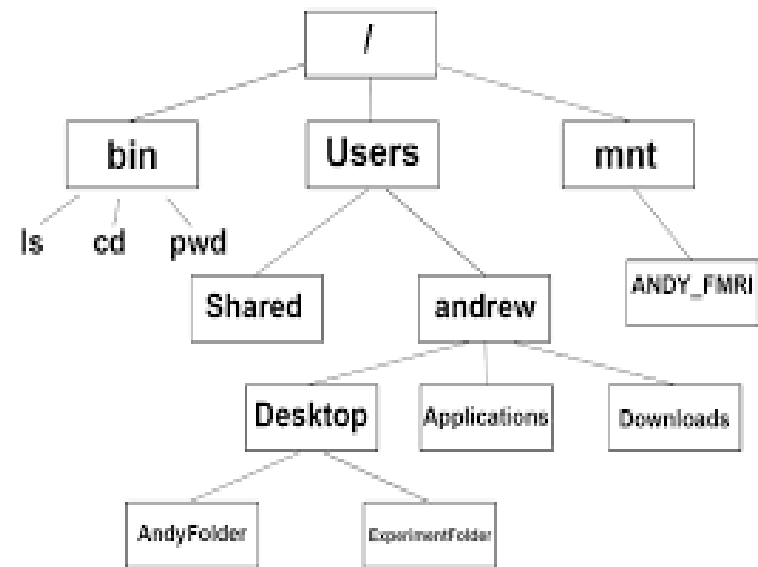
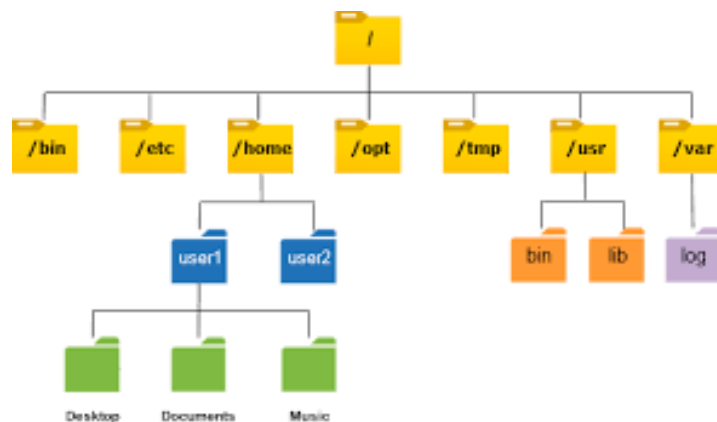
```
rishabh@rishabh: ~
rishabh@rishabh:~$ tree -a GFG
GFG
├── demo1
├── demo2
│   └── sample.txt
└── demo3

3 directories, 1 file
rishabh@rishabh:~$
```



# The Linux Directory Structure

- / — The Root Directory.
- /bin — Essential User Binaries.
- /boot — Static Boot Files.
- /cdrom — Historical Mount Point for CD-ROMs. ...
- /dev — Device Files.
- /etc — Configuration Files.
- /home — Home Folders.
- /lib — Essential Shared Libraries.



# Directory Commands

- The Current Directory: `cd` & Co.
- You can use the **`cd`** shell command to change the current directory: Simply give the desired directory as a parameter:
- **`$ cd letters`** Change to the letters directory
- **`$ cd ..`** Change to the directory above
- If you do not give a parameter you will end up in your home directory:
- **`$ cd`**
- **`$ pwd`**
- **`/home/joe`**
- You can output the absolute path name of the current directory using the **`pwd`** current directory (“print working directory”) command.

# Directory Commands Summary

- **cp file1 file2**            copy file1 and call it file2
- **mv *file1 file2***            move or rename file1 to file2
- **rm file**            remove a file
- **rmdir *directory***        remove a directory
- **cat file**            Display or concatenate a file
- **less file**            display a file a page at a time
- **head file**            display the first few lines of a file
- **tail file**            display the last few lines of a file
- **grep 'keyword' file** search a file for keywords
- **wc file**            count number of lines/words/characters in file

```
carbon@workstation:~$ ls
activity_detail.txt          df-commercial-tools.txt     logcat_radio.txt
'Andriller Outputs'         docker_command_android.txt  logcat_system.txt
Android                     Documents                   logfile1
AndroidStudioProjects       Downloads                   logfile13.txt
batterystats.txt            examples.desktop             logs
bugreport-NRD90M-2020-01-06-09-25-31.zip first.txt                    Music
bug_report.txt              foxy.txt                    myscript
clues.txt                   full_dump.txt               old-andriller-master
cybrary-link.txt            'GFG article'              package_list.txt
DEADJOE                     logcat_events.txt           payouts.txt
Desktop                     logcat_main.txt             Pictures
carbon@workstation:~$
```

# Displaying the contents of a file on the screen

- **less**
- The command less writes the contents of a file onto the screen **a page at a time**. Type less science.txt
- Press the space bar if you want to see another page, type q if you want to quit reading. As you can see, less is used in preference to cat for long files.
- **head**
- The head command **writes the first ten lines of a file to the screen**. First clear the screen then type head science.txt
- Then type head -5 science.txt
- What difference did the -5 do to the head command?
- **tail**
- The tail command writes the **last ten lines of a file to the screen**. Clear the screen and type tail science.txt
- How can you view the last 15 lines of the file?

# Less Command

- To have the lines of the text file numbered for you, use the -N (line numbers) option.
- `less -N Dr-Jekyll-and-Mr-Hyde-001.txt`

```
dave@howtogeek:~$ less -N Dr-Jekyll-and-Mr-Hyde-001.txt
```

- Using Piped Input with Less**
- The `dmesg` command displays the kernel ring buffer messages. We can pipe the output from `dmesg` into `less` using the following command:
- `dmesg | less`

```
dave@howtogeek:~$ dmesg | less
```

```
421
422 "You will not find Dr. Jekyll; he is from home," replied Mr. H
423 yde,
424 blowing in the key. And then suddenly, but still without looki
425 ng up,
426 "How did you know me?" he asked.
427
428 "On your side," said Mr. Utterson "will you do me a favour?"
429
430 "With pleasure," replied the other. "What shall it be?"
431
432 "Will you let me see your face?" asked the lawyer.
433
434 Mr. Hyde appeared to hesitate, and then, as if upon some sudde
435 n
436 reflection, fronted about with an air of defiance; and the pai
437 r stared
438 at each other pretty fixedly for a few seconds. "Now I shall k
439 now you
440 again," said Mr. Utterson. "It may be useful."
```

```
[ 0.000000] Linux version 5.0.0-31-generic (build@lgw01-amd64-046)
[ 0.000000] (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1-18.04.1)) #33-18.04.1-Ubunt
[ 0.000000] SMP Tue Oct 1 10:20:39 UTC 2019 (Ubuntu 5.0.0-31.33-18.04.1-generic
[ 0.000000] S.0.21)
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.0.0-31-generic
[ 0.000000] root=UUID=4a143d08-8695-475b-8243-b13b56050fc2 ro quiet splash
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Hygon HygonGenuine
[ 0.000000] Centaur CentaurHauls
[ 0.000000] [Firmware Bug]: TSC doesn't count with P0 frequency!
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
[ 0.000000] point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers
[ 0.000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
[ 0.000000] x86/fpu: Enabled xstate features 0x7, context size is 8
:
```



# less

- **less is a basic screen reader and it lets you view the contents of a file in a scrollable format.**
- To exit back to the terminal, **press q.**
- Examples
- To view the contents of a file called file1:
- **less file1**
- To view the results of a command (e.g. dmesg) in less, use the pipe character “|” followed by less:
- `dmesg | less`

# cat

- cat is short **for “concatenate”** and it can be used in a variety of ways, including **linking files together** or simply **viewing the contents of a file on screen**.
- Examples
- To output the contents of a file called file1 on the terminal screen:
- **cat file1**
- To output the contents of multiple files on the terminal screen:
- cat file1; cat file2; cat file3
- **To combine two files (file1 and file2) into one file (file3):**
- cat file1 file2 > file3

# head and tail Command

- **head -n 5 state.txt**
- Andhra Pradesh
- Arunachal Pradesh
- Assam
- Bihar
- Chhattisgarh
- **head -n 20 state.txt | tail -10**
- Jharkhand
- Karnataka
- Kerala
- Madhya Pradesh
- Maharashtra
- Manipur
- Meghalaya
- Mizoram
- Nagaland
- Odisha

```
$ cat state.txt
```

1. Andhra Pradesh
2. Arunachal Pradesh
3. Assam
4. Bihar
5. Chhattisgarh
6. Goa
7. Gujarat
8. Haryana
9. Himachal Pradesh
10. Jammu and Kashmir
11. Jharkhand
12. Karnataka
13. Kerala
14. Madhya Pradesh
15. Maharashtra
16. Manipur
17. Meghalaya
18. Mizoram
19. Nagaland
20. Odisha
21. Punjab
22. Rajasthan
23. Sikkim
24. Tamil Nadu
25. Telangana
26. Tripura
27. Uttar Pradesh
28. Uttarakhand
29. West Bengal

# Finding For Files

- We can use the find command to find for files.
- Syntax
- find [where to start searching from]
- [expression determines what to find] [-options] [what to find]
- Consider the following tree hierarchy :

```
rishabh@rishabh: ~  
rishabh@rishabh:~$ tree -a GFG  
GFG  
├── demo1  
├── demo2  
│   └── sample.txt  
└── demo3  
  
3 directories, 1 file  
rishabh@rishabh:~$
```

- Search a file with specific name.
- **\$ find ./GFG -name sample.txt**
- It will search for sample.txt in GFG directory.

- Output :

```
rishabh@rishabh: ~  
rishabh@rishabh:~$ find ./GFG -name sample.txt  
./GFG/demo2/sample.txt  
rishabh@rishabh:~$
```

# Find Example

- How to find and delete a file with confirmation.
- **\$ find ./GFG -name sample.txt -exec rm -i {} \;**
- When this command is entered, a prompt will come for confirmation, if you want to delete sample.txt or not. if you enter 'Y/y' it will delete the file.
- Output :

```
rishabh@rishabh: ~  
rishabh@rishabh:~$ find ./GFG -name sample.txt -exec rm -i {} \;  
rm: remove regular file './GFG/demo2/sample.txt'? y  
rishabh@rishabh:~$ find ./GFG -name *.txt  
rishabh@rishabh:~$
```

# Find Example

- **find .** Find files and folders recursively in current directory.
- **find . -name \*.png -exec cp '{}' ~/images \;** Copy all PNG files to ~/images folder.
- **find . -name \*.txt -exec mv '{}' ./txt \;** Move all TXT files to ./txt folder.
- **find . -name .svn -prune -exec rm -r '{}' \;** Delete all .svn folders.
- **find . -type f -exec file '{}' \;** Run files.

# dir

- The command “dir” stands for directory and it is used to display the list of all directories or folder in the current directory.
- Syntax: **dir**
- Example:

```
haya@ubuntu: ~
File Edit View Search Terminal Help
haya@ubuntu:~$ dir
Desktop      Downloads      Music      Public      Templates
Documents    examples.desktop  Pictures    snap      Videos
haya@ubuntu:~$
```

# whereis

- The command “whereis” is self-explanatory, as it **displays the path** where the package for specific built-in Linux command locates.
- Syntax: **whereis command\_name**

- Example:
- **whereis zip**
- **whereis help**
- **whereis cat**

```
haya@ubuntu: ~/Desktop
File Edit View Search Terminal Help
haya@ubuntu:~/Desktop$ whereis zip
zip: /usr/bin/zip /usr/share/man/man1/zip.1.gz
haya@ubuntu:~/Desktop$ whereis help
help: /usr/share/help
haya@ubuntu:~/Desktop$ whereis cat
cat: /bin/cat /usr/share/man/man1/cat.1.gz
haya@ubuntu:~/Desktop$
```

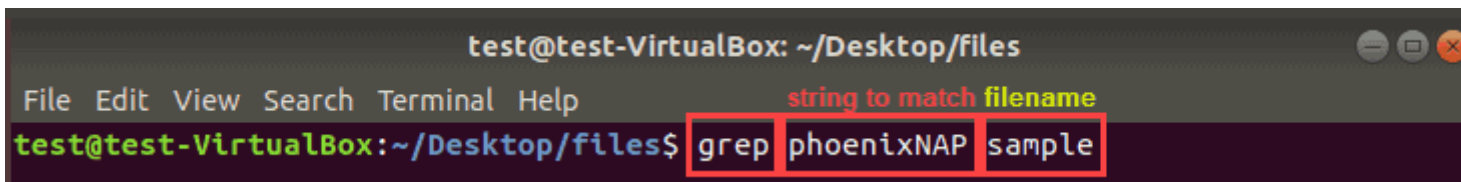


# grep

- grep stands **for “Global Regular Expression Print”** and it lets **you search for strings of text inside files**. You can think of it as the Google for your filesystem, and it becomes extremely powerful when you combine it with regular expressions.
- Examples
- **To search for the string “Hello” in the file called greetings in the current directory:**
- `grep "Hello" greetings`
- To recursively search for the string “Error” in all files and folders under the directory ~/projects:
- **`grep -r "Error" ~/projects`**
- To recursively (-r) search for strings under ~/projects with the word “error”, case insensitive (-i), and also show the line number where the string appears (-n):
- `grep -rin "error" ~/projects`

# grep

- Grep is a Linux / Unix command-line tool used to **search for a string of characters in a specified file**. The text search pattern is called a regular expression. When it finds a match, it prints the line with the result. The grep command is handy when searching through large log files.
- The grep command consists of **three parts** in its most basic form. The first part starts with grep, followed by the pattern that you are searching for. After the string comes the file name that the grep searches through.
- The simplest grep command syntax looks like this:
- **grep [options] pattern [files]**



```
test@test-VirtualBox: ~/Desktop/files
File Edit View Search Terminal Help
test@test-VirtualBox:~/Desktop/files$ grep phoenixNAP sample
```

# Grep Options Description

- **-c** : This prints only a count of the lines that match a pattern
- **-h** : Display the matched lines, but do not display the filenames.
- **-i** : Ignores, case for matching
- **-l** : Displays list of a filenames only.
- **-n** : Display the matched lines and their line numbers.
- **-v** : This prints out all the lines that do not matches the pattern
- **-e exp** : Specifies expression with this option. Can use multiple times.
- **-f file** : Takes patterns from file, one per line.
- **-E** : Treats pattern as an extended regular expression (ERE)
- **-w** : Match whole word
- **-o** : Print only the matched parts of a matching line, with each such part on a separate output line.

# grep Example

- Consider the below file as an input.
- **\$cat > geekfile.txt**

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

# 1. Case insensitive search

- The -i option enables to search for a string case insensitively in the give file. It matches the words like “UNIX”, “Unix”, “unix”.
- **\$grep -i "UNix" geekfile.txt**

unix is great os. unix is opensource. unix is free os.  
 learn operating system.  
 Unix linux which one you choose.  
 uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

- **Output:**
- unix is great os. unix is opensource. unix is free os.
- Unix linux which one you choose.
- uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

## 2. Displaying the count of number of matches

- We can find the number of lines that matches the given string/pattern
- **\$grep -c "unix" geekfile.txt**

unix is great os. unix is opensource. unix is free os.  
 learn operating system.  
 Unix linux which one you choose.  
 uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

- **Output:**
- 2

### 3. Display the file names that matches the pattern

- We can just display the files that contains the given string/pattern.
- `$grep -l "unix" *`
- or
- `$grep -l "unix" f1.txt f2.txt f3.txt f4.txt`

unix is great os. unix is opensource. unix is free os.  
 learn operating system.  
 Unix linux which one you choose.  
 uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

- **Output:**
- `geekfile.txt`

## 4. Checking for the whole words in a file

- By default, grep matches the given string/pattern even if it found as a substring in a file. The -w option to grep makes it match only the whole words.
- `$ grep -w "unix" geekfile.txt`

```
unix is great os. unix is opensource. unix is free os.  
learn operating system.  
Unix linux which one you choose.  
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- **Output:**
- unix is great os. unix is opensource. unix is free os.
- uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.



## 5. Displaying only the matched pattern

- By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.
- `$ grep -o "unix" geekfile.txt`

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- **Output:**
- unix
- unix
- unix
- unix
- unix
- unix

## 6. Show line number while displaying the output using grep -n

- show the line number of file with the line matched.
- `$ grep -n "unix" geekfile.txt`

```
1.  unix is great os. unix is opensource. unix is free os.
2.  learn operating system.
3.  Unix linux which one you choose.
4.  uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a
    powerful.
```

- **Output:**
- 1:unix is great os. unix is opensource. unix is free os.
- 4:uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

## 7. Inverting the pattern match

- You can display the lines that are not matched with the specified search string pattern using the -v option.
- `$ grep -v "unix" geekfile.txt`

unix is great os. unix is opensource. unix is free os.  
 learn operating system.  
 Unix linux which one you choose.  
 uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

- Output:**
- learn operating system.
- Unix linux which one you choose.

## 8. Matching the lines that start with a string

- The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern.
- `$ grep "^unix" geekfile.txt`

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- **Output:**
- unix is great os. unix is opensource. unix is free os.

## 9. Matching the lines that end with a string

- The \$ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern.
- `$ grep "os$" geekfile.txt`

## 10.Specifies expression with -e option

- Can use multiple times :
- `$grep -e "Agarwal" -e "Aggarwal" -e "Agrawal" geekfile.txt`

# 11. -f file option Takes patterns from file, one per line.

- `$cat pattern.txt`
- Agarwal
- Aggarwal
- Agrawal
- `$grep -f pattern.txt geekfile.txt`

# wc (word count)

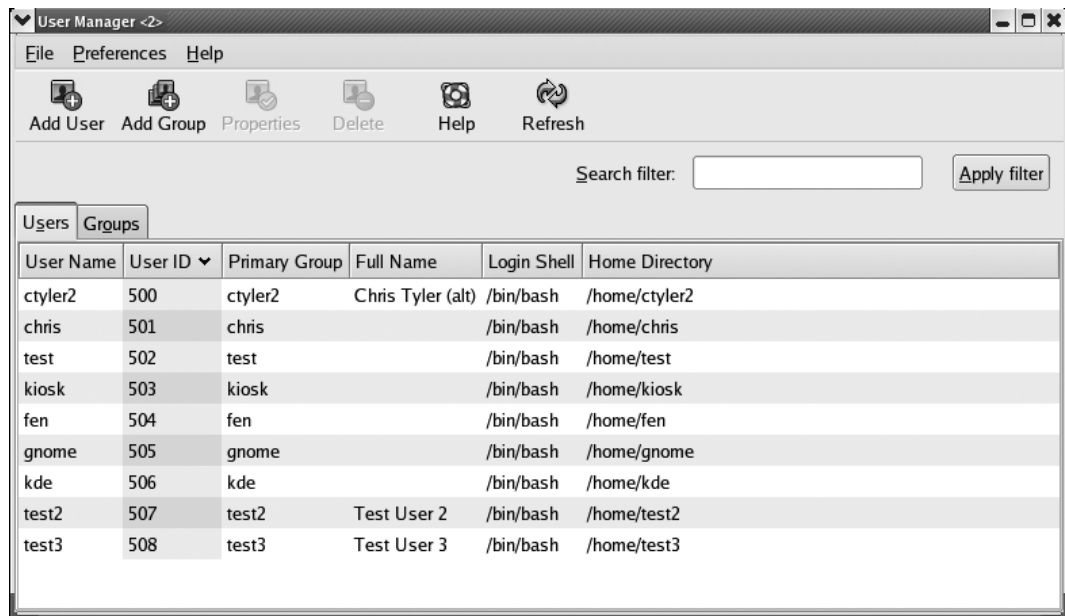
- A handy little utility is the *wc* command, short for word count. To do a word count on **science.txt**, type
- **wc -w science.txt**
- To find out how many lines the file has, type
- **wc -l science.txt**
- To find out how many characters the file has, type
- **wc -m science.txt**



# Linux System Administration:

## Managing Users and Groups

- Linux is a multiuser OS, meaning that it provides features to help multiple individuals use the computer. Collectively, these features constitute accounts.
- **Understanding accounts.**
- **Using account tools.**
- **Working as root.**

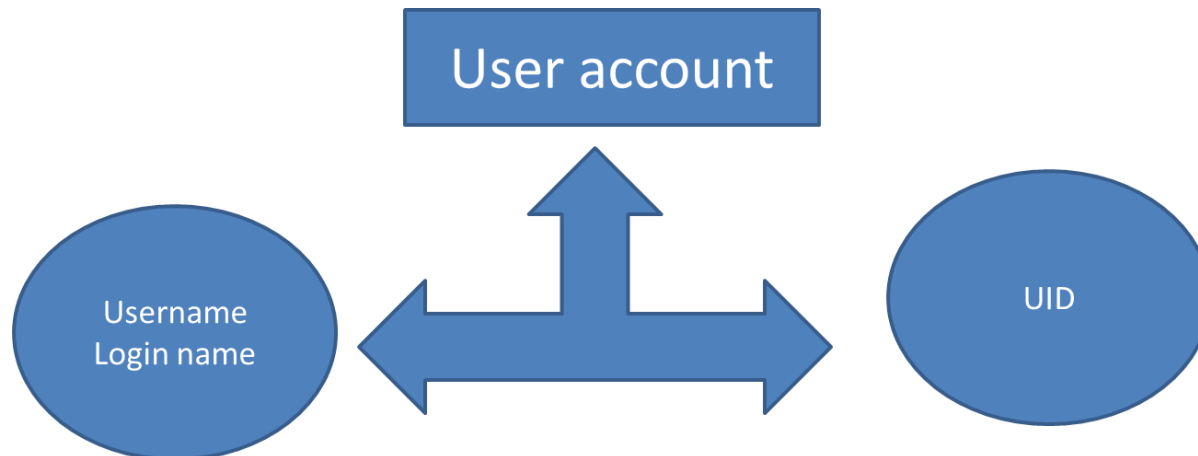


# Understanding Accounts

- Accounts enable multiple users to **share a single computer** without causing each other too much trouble.
- They also **enable system administrators to track who is using system resources and, sometimes, who is doing things that they shouldn't be doing**. Account features help users use a computer and administrators administer it. Understanding these features is the basis for enabling you to manage accounts.
- Some account features help you **identify accounts and the files and resources associated** with them. Knowing how to use these features will help you track down account-related problems and manage the computer's users.

# What is a user account?

- A user account is a systematic approach to track and monitor the usage of system resources. Each user account contains two unique identifiers; **username and UID**.
- When a user account is created, its username is mapped to a unique UID.

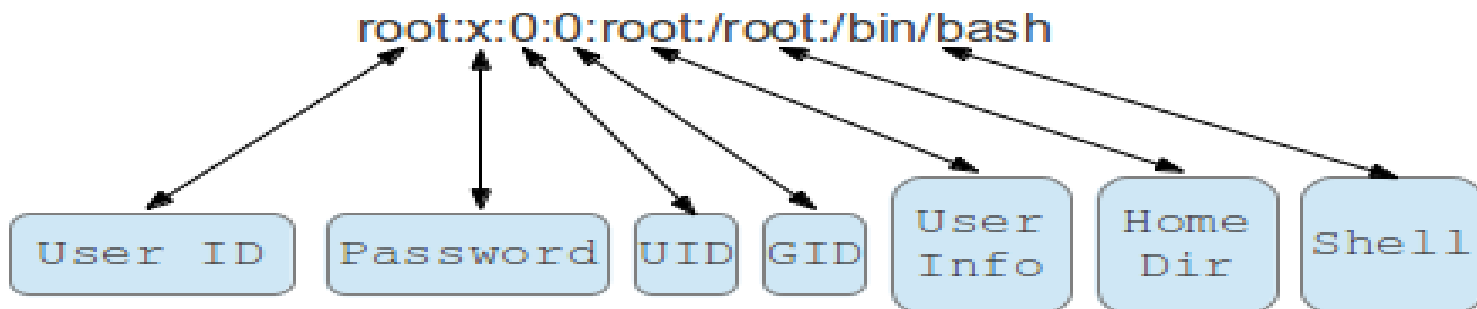


# What is a user account?

- Username is flexible. It can be changed as per requirement.
- Regardless it is selected first time or changed later; it must be unique in system. **Two users can't use the same username.**
- **UID is fixed.** It cannot be changed. Once assigned, it always remains the same for that user account.
- Username is used to access the user account. Username is also known as login name. **UID is used to authenticate**, track and monitor the activity of user account. Username is used by the user while the UID is used by the system.

# Understanding Account Features

- Most account features are defined in the `/etc/passwd` file, which consists of colon-delimited lines, with each line (or record) defining a single account. An entry might resemble the following:
- **rich:x:1003:100:Rich Blum:/home/rich:/bin/bash**
- `<username>:<password>:<UID>:<GID>:<GECOS>:<home directory>:<shell>`
- The information contained in the fields of this record includes the following:
- **Username, Password, UID, GID, Comment Field, and Home Directory.**



# rich:x:1003:100:Rich Blum:/home/rich:/bin/bash

- **Username:** An account's username is its most relevant feature.
- **Password:** User accounts are typically protected by a password, which is required to log into the computer. The password is stored as a salted hash.
- **UID:** The computer uses a **user identification (UID)** number to track accounts.
- **GID:** Accounts are tied to one or more groups, which are similar to accounts in many ways; however, a group is a collection of accounts. One of the primary purposes of groups is to enable users to give certain users access to their files, while preventing others from accessing them. Each account is tied directly to a primary group via a **group ID (GID)** number (100 in the preceding example). Accounts can be tied to other groups by inclusion in the group's definition.
- **Comment Field:** The comment field normally holds the user's full name (Rich Blum in this example), although this field can hold other information instead of or in addition to the user's name.
- **Home Directory:** User accounts, and some system accounts, have home directories(/home/rich in this example). A home directory is an account's "home base." Normally, ownership of an account's home directory belongs to the account.

# user account's UID

- You can find out a user account's UID, the primary and secondary groups and the corresponding GIDs by means of the `id` program:
- *\$ id*
- uid=1000(joe) gid=1000(joe)  
groups=24(cdrom),29(audio),44(video), 1000(joe)
- *\$ id root*
- uid=0(root) gid=0(root) groups=0(root)

# Thank You



# Linux

**Dr. Hatem Yousry**  
**E-mail: [Hyoustry@nctu.edu.eg](mailto:Hyoustry@nctu.edu.eg)**





