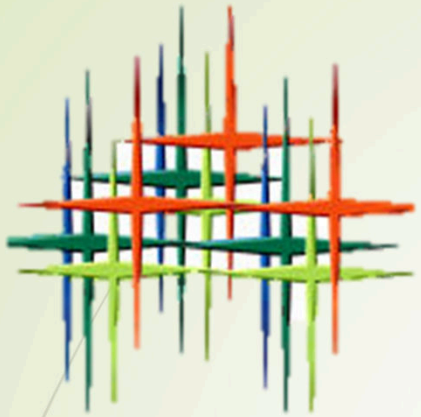


ABC IT Education

WE'LL TAKE YOU FROM ZERO TO HERO IN A SNAP

Linux Systems Administration

Linux Intermediate 3

Scheduling Repeated Jobs with Cron, Switching Users and Running Commands as Others, Shell History and Tab Completion, Installing Software

Scheduling Repeated Jobs with Cron

- This lesson covers the cron service, the crontab or cron table format, as well as the crontab command.
- If you need to repeat a task on a schedule, you can use the cron service.
- The cron service is a time based job scheduling system that starts when the system boots.
- Every minute the cron service checks to see if there are any scheduled jobs to run and if so, it runs them.
- Cron jobs are often used to automate a process or perform routine maintenance.
- You schedule cron jobs using the crontab command.
- A crontab or a cron table is a configuration file that specifies when commands are to be executed by cron.

Crontab Format

```
* * * * * command
| | | | |
| | | | +-- Day of the Week   (0-6)
| | | +---- Month of the Year (1-12)
| | +------ Day of the Month  (1-31)
| +-------- Hour              (0-23)
+---------- Minute            (0-59)
```

Example Crontab Entry

```
# Run every Monday at 07:00.
0 7 * * 1 /opt/sales/bin/weekly-report
| | | | |
| | | | +-- Day of the Week   (0-6)
| | | +---- Month of the Year (1-12)
| | +------ Day of the Month  (1-31)
| +-------- Hour              (0-23)
+---------- Minute            (0-59)
```

Redirecting Output

```
# Run at 02:00 every day and
# send output to a log file.
0 2 * * * /root/backupdb > /tmp/db.log 2>&1
```

Example Crontab Entries

```
# Run every 30 minutes.
0,30 * * * * /opt/acme/bin/half-hour-check

# Another way to do the same thing.
*/2 * * * * /opt/acme/bin/half-hour-check

# Run for the first 5 minutes of the hour
0-4 * * * * /opt/acme/bin/first-five-mins
```

- Each line in a crontab represents a job and contains two pieces of information: 1) when to run and 2) what to run.
- The time specification consists of five fields.
- They are minutes, hour, day of the month, month, and day of the week, and after the time specification, you provide the command to be executed.
- This example crontab entry runs every Monday at 7 AM.
- The command will only be executed when all of the time specification fields match the current date and time.
- You can specify that a command be run only once, but this is not the typical use case for cron.
- Typically, one or more of the time specification fields will contain an asterisk (*) or star, which matches any time or date for that field.

- This job will run only when the minute is zero, the hour is 7 and the day of the week is 1.
- In the day of the week field - 0 represents Sunday, 1 is Monday, etc.
- This job will run on any day of the month and during any month since the asterisk was used for those two fields.
- If any output is generated by the command on a cron job, it is mailed to you.
- You can check your local mail with the mail command, but if you would prefer not to get email, you can redirect the output of the command as in this example.
- You can provide multiple values for each of the fields, so if you would like to run a command every half-hour, you could use '0,30' in the minute field or '*/2'.

- You can even use ranges with a dash.
- If you want to run a job every minute for the first 5 minutes of the hour, you can use '0-4' and the minutes field.
- That job will get executed at the top of the hour, 0 minutes, then 1 minute, 2 minute, 3 minute and at the fourth minute, running a total of 5 times every hour.
- There are several implementations of the cron scheduler and some of those implementations allow you to use shortcuts and keywords in your crontabs.
- There are some common shortcuts, but these may or may not work on your Linux distribution, so the best thing to do is to man cron and check to see if these will work for you.

Crontab Shortcuts

@yearly	0 0 1 1 *
@annually	0 0 1 1 *
@monthly	0 0 1 * *
@weekly	0 0 * * 0
@daily	0 0 * * *
@midnight	0 0 * * *
@hourly	0 * * * *

- The '@yearly' shortcut means to run once a year at midnight on the morning of January, 1st.
- Annually is just the exact same thing as yearly.
- Monthly means run once a month at midnight on the morning of the first day of the month.
- Weekly, run once a week at midnight on the morning of Sunday.
- Daily, run everyday at midnight, and Midnight is the same thing as daily.
- And hourly means to run once an hour at the beginning of the hour or at the top of the hour.
- The crontab command manipulates cron jobs.
- You can use 'crontab file' to install the new crontab from the contents of that file.

Using the Crontab Command

`crontab file`	Install a new crontab from file.
`crontab -l`	List your cron jobs.
`crontab -e`	Edit your cron jobs.
`crontab -r`	Remove all of your cron jobs.

- To list your jobs, run 'crontab -l' and to edit them, use 'crontab -e'. 'crontab –e' will invoke the editor that's specified in the editor environment variable and 'crontab -r' removes all of your cron jobs.
- If we were in the crontab -l command, it should list our cron jobs, and in this case I do not have any cron job scheduled, let's create one.

- Edit a file with your favorite editor vi. Any line that begins with an octothorpe, or pound sign or hash or whatever you call that, is a comment and will be ignored in shell scripts, it will be ignored by cron, etc.
- Let's create a time specifications so, 0 minutes, 7 for the hour, and 1 represents Monday.
- And we'll put the command we want to run ‘cat ~/<file>'.
- Save that to a file and we can install that file by running crontab in the file name.
- Now when you list our cron jobs, you'll see that we have one cron job.
- Let’s look at the editor environment variable, if it is not set, let's set editor to vi.

- Edit the crontab by running crontab -e.
- If I want to disable this cron job for instance, I could just comment it out.
- Another way I can do that is run 'crontab -r' to delete my cron jobs and you can see that - that removed them.
- Remember the cron service runs scheduled jobs and you can manipulate those jobs by the crontab command.

Switching Users & Running Commands as Others

- We will now see how to switch to other accounts and how to run commands and programs as other users.
- One way to start a session as another user on the system is to use the 'su' command.
- If no arguments are supplied to 'su', it assumes you want to become the superuser or root account, so executing 'su' without any options is the same thing as executing 'su root'.
- Your current environment is passed to the new shell unless you specify a hyphen (-) and in that case, su creates an environment like you would expect to see had you logged in as that user.

- If the command you want to execute is more than one word in length, then you need to surround it with quotes.

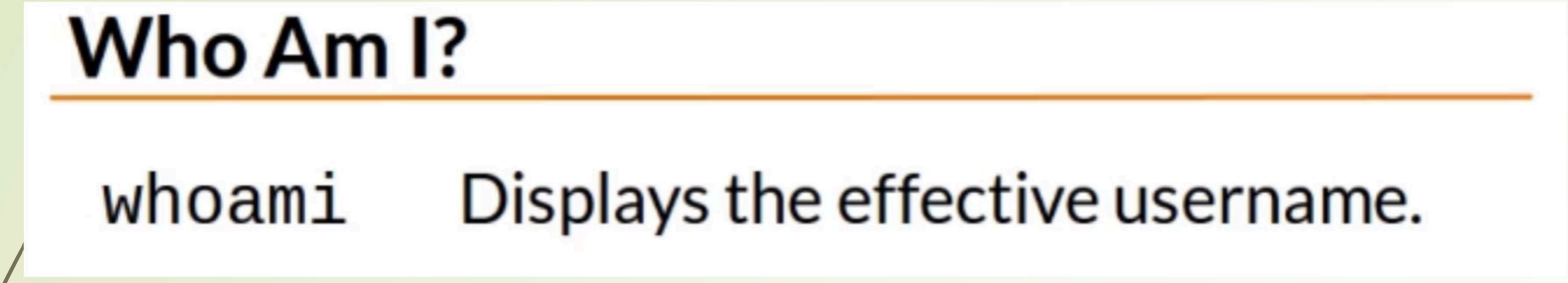
The su Command

`su [username]`	Change user ID or become superuser

su Options

-	A hyphen is used to provide an environment similar to what the user would expect had the user logged in directly.
`-c command`	Specify a command to be executed.

- If you want to know what user you are working as, run the 'whoami' command.
- If you run 'whoami', it should return your current account name and after you su, or switch to another user, ‘whoami’ should return that user’s account name.



- Let's set an environment variable say LOC to test with.
- $ export LOC=Bowie ; echo $LOC, it shows Bowie
- Now let's use the su command to switch to the root user. Notice that that environment variable LOC still exists under root

- Let's use su with a dash (hyphen) to get homeuser's environment.
- You can see that our environment variable of LOC was not passed on when we 'su -' to the homeuser user.
- Also we ended up in the homeuser's home directory.
- Let's pass a command, say 'echo $PWD'.
- And nothing was returned, we didn't specify that we wanted homeuser 's environment, rather we used our environment.
- Let's use the dash meaning get homeuser's environment, now you can see that that environment variable that exists in the user’s account is accessible to homeuser.

- Another way to switch users or execute commands as others is to use the ‘sudo’ command.
- Sudo allows you to run programs with the security privileges of another user.
- Like su, if no username is specified it assumes you are trying to run commands as the superuser (root).
- This is why sudo is referred to as super user do.
- It is commonly used to install, start, and stop application that require root privileges.

Sudo - Super User Do

sudo	Execute a command as another user, typically the superuser.

- One of the advantages of using sudo over the su command is that you don't need to know the password of the other user.
- This can certainly eliminate some issues that arise from using shared passwords and/or generic accounts.
- When you execute the sudo command you are prompted for your password, and if the sudo configuration permits access, that command is executed.
- sudo configuration is typically controlled by the system administrator and requires root access to change.
- On your personal system, you have access to the root account so you are additionally the system administrator.

- To see the commands that are available for you to run or sudo, use 'sudo -l'.
- To run a command as a root user, just run sudo, space the command you want execute.
- You can specify a user with '-u', so 'sudo -u root command' is the exact same thing as 'sudo command'.
- However if you want to run a command as another user, you need to specify that with '-u‘ like 'sudo -u homeuser command' for example.
- You can use 'sudo su' to switch to the super user account.
- If you use 'sudo su -' and hit enter, you get switched to the super user account with the roots environment.

Using sudo

`sudo -l`	List available commands.
`sudo command`	Run command as root.
`sudo -u root command`	Same as above.
`sudo -u user command`	Run as user.

- 'sudo su - username' would switch to that username's account with an environment like you would expect to see had you logged in as that user.
- Another way to switch to another account with sudo is to use the '-s' option, so running 'sudo -s', that will start a shell as root, or you can specify a user with a '-u' option.
- We can use the 'sudo -l' command to see what commands we can run with sudo. Let's try it **$ sudo -l**

Using sudo

`sudo su`	Switch to the superuser account.
`sudo su -`	Switch to the superuser account with root's environment.
`sudo su - username`	Switch to the username account.

Using sudo

`sudo -s`	Start a shell
`sudo -u root -s`	Same as sudo -s
`sudo -u user -s`	Start a shell as user

- In this case it looks like you can run any command and I don't even need to specify a password.
- Example
- Let’s create a new user using sudo to execute a command as root.

  $ sudo useradd –c “Home User” –d /home/homeuser –s /bin/bash –m homeuser
- You can also run sudo with '-u' to run programs as other users besides root, so let's run something as the user yamba.

  $ sudo –u homeuser cat /etc/passwd
- And that worked and the command was run as home user

- Let's switch to the msaks account.

  $ sudo su - msaks

- Do a ‘whoami’ and you can see the user is msaks.
- Now back to the adminuser account. $ exit
- I can also use sudo with a '-s'. $ sudo –s to change to the root account.
- Do a ‘whoami’ and you can see the user is root and in the home directory of adminuser.
- Now back to the adminuser account. $ exit
- Let's do this another way by switching to the root account with 'sudo su‘ or even 'sudo su - ' to get root environment.
- Do a ‘whoami’ and you can see the user is root.

Changing the sudo Coniguration

`visudo`	Edit the /etc/sudoers file

- If you need to modify the sudo configuration, use the command 'visudo'.
- 'visudo' starts the vi editor and edits the etc sudoers file.
- The visudo command has to be executed with the group privileges.
- This means that you either need to switch to the root account and run visudo, for example 'su root' then run visudo, or run 'sudo visudo' as your own account.
- The sudoers file contains a list of users and the commands those users can run and ask what users those commands can be run as.

- There are many options to the sudoers file, but the most simple format and one that you'll see quite often is a line that gives a specific user a set of commands that he or she can run.
- The specification starts with the user that is getting the sudo privilege, followed by the host on which that user can execute those privileges on.
- This allows system administrators to use one sudoers configuration file and distribute that to multiple servers.
- The user listed in parenthesis is the user that the commands will run as, then a list of commands is provided that the user can run.
- You can separate multiple commands with a coma.

- Also if you don't require that a password be entered when using sudo, you can supply the no password decorator in front of the commands.
- **Summary**
- Use the su command to switch users.
- To get the user's environment that you're switching to, remember to use a hyphen or a dash.
- The whoami command displays your current account name.
- The sudo command allows you to execute programs as other users.
- To switch users with sudo, use 'sudo -s' or 'sudo su'.
- If you need to change the sudoers configuration, use the visudo command to edit the sudoers file.

Shell History and Tab Completion

- In this section, we will cover the shell history, how to repeat commands or portions of commands with the exclamation mark syntax and how to use auto completion.
- Each command you enter into the shell is recorded in your shell history, so having access to your shell history is extremely useful.
- You can search through the shell history, repeat commands you previously entered and even recall previous commands and change them slightly before executing them again.
- Not only can they save you time and keystrokes, but it can prevent you from making mistakes by running a previously known good command.
- Some shells like bash keep their history in memory and only write the history to a file on exit.

- In this section, we will cover the shell history, how to repeat commands or portions of commands with the exclamation mark syntax and how to use auto completion.
- Each command you enter into the shell is recorded in your shell history, so having access to your shell history is extremely useful.
- You can search through the shell history, repeat commands you previously entered and even recall previous commands and change them slightly before executing them again.
- Not only can they save you time and keystrokes, but it can prevent you from making mistakes by running a previously known good command.
- Some shells like bash keep their history in memory and only write the history to a file on exit.

- Common history files include - '.bash_history', '.history' and '.histfile‘, and these files are stored in your home directory.
- The history command displays the commands in your shell history, and precedes each one of the commands with a number that can be used to reference that command at a later time.

history Command

`history`	Displays the shell history.
`HISTSIZE`	Controls the number of commands to retain in history.

```
export HISTSIZE=1000
```

- By default, bash retains 500 commands in your shell history, and is controlled by the 'HISTSIZE' environment variable.

- If you want to increase this number, add 'export HISTSIZE=some number‘ and place that in your personal initialization files.

! Syntax

`!N`	Repeat command line number N.
`!!`	Repeat the previous command line.
`!string`	Repeat the most recent command starting with "string."

- If you want to rerun command number 3 in your history, you would type, '!3' enter.
- Sometimes the exclamation mark is abbreviated or called bang, so if I were to say 'bang 3', we know that that means the exact same thing as '!3'.

- If you want to repeat the previous command, you can use 'bang bang' (!!) enter.
- If you want to repeat a command that starts with a certain letter or string, you can enter bang followed by that string.
- Let's say you want to repeat the cat command executed a few moments earlier, you can simply enter 'bang c' enter.
- In addition to executing entire commands in your history, you can pull out parts of a command line like '!:number'.

More ! Syntax

```
!:N <Event> <Separator> <Word>
```

`!` Represents a command line (or event).

- ! = The most recent command line.
- ! = !!

`:N` Represents a word on the command line.

- 0 = command, 1 = first argument, etc.

- The bang represents an event.
- You can use the bang syntax we just talked about, i.e. bang number or bang string or bang bang.
- When pulling out words from a previous command, you can abbreviate bang bang to just bang.
- The number in the syntax represents a word on the command line, where Zero represents the first word on the command line which is always going to be a command.
- One represents the second word on the command line.
- An easy way to think about this is that it represents the first argument to the command.
- Let's say you execute this command.
- $ 'head files.txt sorted_files.txt notes.txt'.

! Syntax Examples

```
$ head files.txt sorted_files.txt notes.txt
<Output from head command here>
$ !!
head files.txt sorted_files.txt notes.txt
<Output from head command here>
$ vi !:2
vi sorted_files.txt
<vi editor starts>
```

- The bang represents an event.
- You can use the bang syntax we just talked about, i.e. bang number or bang string or bang bang.
- When pulling out words from a previous command, you can abbreviate bang bang to just bang.

- To repeat that entire command, type '!!' enter.
- If you want to pull out the second argument to the previous command, use '!:2', so 'vi !:2' is the same as 'vi sorted_files.txt'.
- In this example, '!:0' is equivalent to head, '!:1' is files.txt, '!:2' is sorted_files.txt, and '!:3' is notes.txt.
- Here are two more exclamation mark syntax shortcuts.

Even More ! Syntax

```
!^   Represents the first argument.
     !^ = !:1
```

!$ Represents the last argument.

```
$ head files.txt sorted_files.txt notes.txt
```

!^ = files.txt

!$ = notes.txt

- The first is '!^’ and '!^' represents the first argument to a command meaning it is the same as '!:1'.
- The next shortcut is '!$‘ which represents the last word on the command line.
- In this example command line, '!^' equals files.txt and '!$' represents notes.txt.
- If you wanted to edit notes.txt, you could count the number of words on the command line and use 'vi !:3’ or save the counting and a character and simply type 'vi !$'.

- You can search for commands in your command history.

Searching Shell History

`Ctrl-r`	Reverse shell history search
Enter	Execute the command
Arrows	Change the command
Ctrl-g	Cancel the search

- To start a reverse shell history search, type 'ctrl r'.
- Your prompt changes to reflect that you are performing a search.
- Start typing in characters from the previous command that you're searching for.
- If the first result returned is not what you're looking for, type 'ctrl r' again to get the next previous match.

- To execute the command as it is simply press the enter key.
- If you want to modify the command before executing it, use the left or right arrow keys to start navigating the command line.
- Make changes and type enter to execute the command.
- To cancel your search and return to the prompt, type 'ctrl g'.
- **Tab Completion**
- Another way to increase your efficiency at the command line is by using tab completion.
- After you start typing a command, you can hit the tab key to invoke tab completion.
- Tab attempts to automatically complete partially typed commands.

- If there are multiple commands that begin with the same string that precedes tab, those commands will be displayed.
- You may need to hit the tab key twice in order to reveal the possible completion options.
- You can continue to type and press tab again and when there is only one possibility remaining, pressing the tab key will complete the command.
- Tab completion not only works on commands but also works on files and directories.
- You can even use tab completion for environment variables or usernames when using the tilde expansion syntax.
- Play with tab completion to become familiar with it and it will become your best friend on the command line.

- **Class Practice**
- The ‘history’ command displays a list of commands in your shell history preceded by a number.

  Type **$ history** <enter>
- To run commands with a number you can type '! and the number‘ so '!1' will run the first command in our history, which is history.
- Let's run another command here, ‘echo $HISTSIZE'.

  Rerun previous command, using '!' syntax. Type **$!!** <enter>
- That displays the command that's going to be executed, then executes the command and then it returns the output from that command.

- **Class Practice**
- Let's repeat the history command using the '!' and a string syntax. Type **$!h** for the most previous command that started with 'h'.
- To run the echo command we could use **'!2'** since it's number 2 in our history list and then again it displays what it's going to run, runs it and then provides the output.
- Looking at this 'ls' command, $ ls files.txt sorted_files.txt notes.txt`, to run the previous command over again type **'!!'**.
- To pull out portions of that command we can use the '!:' and the number syntax.
- Type **'echo !:2'** to get the second argument of the ls command, So 'ls' is 0, 'files' is 1, 'sorted_files' is 2, and 'notes' is 3.

- **Class Practice**
- 'echo !:2' should return 'echo sorted_files.txt‘, then it appends the command that it's going to run and then the output of that command.
- To repeat a command that starts with a particular letter, we could do that again with **'!l’**, let's pull that up.
- Thus we've run our 'ls' command again.
- If we want to echo the last thing on the previous command line, we can use **‘ echo !$‘** and it will 'echo notes.txt'.
- Let's run our 'ls' command again. **!l**
- If we want to echo the first argument to the previous command, we can use **'echo !^'.**

- **Class Practice**
- That pulls out files.txt since it was the first argument on this command line.
- Let's run some more commands here. **'echo DEL MA VA'**.
- $ echo !^ will pull the first argument of the last command which will be DEL.
- Then we can use the bang syntax to pull out the second argument from the command that began with 'l' meaning the 'ls', and that would be sorted_files and then we'll pull out the 'ls' command again take out the command itself

  $ echo !^ !l:2 !l:0

- And that gives 'echo DEL sorted_files.txt ls'.

- **Class Practice**
- To perform a reverse search we can type **'ctrl r‘** and you see that the prompt changes to say that you're performing a reverse search.
- This is going to search for echo, so start typing an **'e’** then **'c’**, and it pulls out the previous echo command.
- Type **'ctrl r'** again and we can get the next previous command, then **'ctrl r'** again and when we find the command we want, just hit enter to execute that command.
- Let's retry **'crtl r'** to go to a different command and if we want to modify this command before it executed, just press an arrow key and then edit the command line.
- So let's take off these items and then press enter.

- **Class Practice**
- If you're doing a reverse search, and you want to cancel it, that is done with **'ctrl g'**, and that brings you back to your prompt with nothing executed.
- Let's look how the help tab completion works.
- Let's type the word 'who‘ and then hit tab tab.
- There are two possible expansions here, who and whoami
- Type the letter 'a' and hit tab, now there's only one option and it completes that command. Hit enter and it runs that command.
- Let's do the same thing with file, type 'ls file', hit tab a couple of times and it shows our choices here.

- **Class Practice**
- So let's do 'files.txt'.
- Let's show how this can be pretty quick.
- So if I type 'ls sorted_files.txt notes.txt' and hit enter, that does not take long but if you use tab completion it could be a lot quicker.
- If I just typed 'ls so', 'tab', and then 'n', 'tab`, it did the file completion for us.
- You can also use tab completion for environment variables and we know the hist environment variable or the HISTSIZE environment variable controls how many commands are stored in our history.
- So let's type - H. I. S. T. and tab.

- **Class Practice**
- You may or may not see many options in your environment.
- If more than 1 found, just hit 's', and then press tab, and it completes it.
- You also can do the tab completion with usernames, with the tilde expansion syntax.
- Remember tilde (~) represents your home directory so if you don't give an argument, that is 'home/<username>'.
- Let's do 'ls /home' to see if there are any 2 username that begin with the letter, if not we will create the users.
- If we type in '~<letter(s)>' and hit tab a couple of times, we can see that there are two users that begin with the same letter on your system.

- **Class Practice**
- Now type the next letter that is in one name and not the other, then hit tab, it auto completes, we'll press enter and then that displays the home directory of that user.
- **Summary**
- We have talked about how shell history is stored, how the number of commands it saves can be manipulated by the HISTSIZE environment variable.
- Also talked about the exclamation mark syntax and how powerful it is and can be used to repeat entire commands from shell history or extract portions of previous commands.
- Using the tab key to perform auto completion can increase your efficiency, speed up your work flow and prevent typing mistakes.

Installing Software

- This topic covers packages, package managers and how to find, install and remove software for the most popular Linux distributions.
- When you install software on a Linux system, you do so with a package.
- A package is just a collection of files that make up an application, in addition, a package contains data about the application as well as any steps required to successfully install and remove that application.
- The data, or metadata, that is contained within a package can include such information as the description of the application, the version, and a list of dependencies or other packages that this particular application needs in order to function.

- A package manager is used to install, upgrade, and remove packages.
- The package manager uses a package's metadata to automatically install any required dependencies.
- Package managers keep track of what files belong to what packages, what packages are installed, and what versions of those packages are installed.
- Here's a list of distributions that are based on the RPM package format. RPM stands for RedHat Package Manager.
- Obviously the RedHat distribution uses RPMs as well as CentOS, Fedora, Oracle linux and Scientific Linux.
- The yum command line utility is a package management program for those Linux distributions that use the RPM packages.

- You can find packages to install with **'yum search'** and provide that a search string.
- You can also display information about a package or an application by using **'yum info'** and the package name.
- You can install packages with the **'yum install package'** command.
- yum will typically ask you to review your request and say 'yes' or 'no' if you want to continue.
- To automatically answer yes to yum's question, **use '-y'.**
- Also note that installing or removing software requires root or super user privileges.
- To remove a package, simply enter 'yum remove' and the name of the package.

yum

`yum search string`	Search for string
`yum info [package]`	Display info
`yum install [-y] package`	Install package
`yum remove package`	Remove package

- In addition to the yum command, you can use the 'rpm' command to interact with the package manager.
- **'rpm -qa'** lists all the installed packages.
- **'rpm -qf'** with a path to a file will tell you what package that file belongs to.
- Using **'rpm -ql package_name'** will list all the files that belong to that particular package.

- To install a package, use 'rpm -ivh' and the package file.
- To erase or uninstall a package, use 'rpm -e' with the package name.

rpm

`rpm -qa` List all installed packages.

`rpm -qf /path/to/file` List the file's package.

rpm

`rpm -ql package` List package's files.

`rpm -ivh package.rpm` Install package.

rpm

`rpm -e package` Erase (uninstall) package.

- Let's look for an application to install like inkscape.
- We'll do a search and we see 3 items returned.
- Now do yum info on each and see what they do.

  $ yum info inkscape-docs

- inkscape-docs says it's a tutorial and examples for you, so that's not what we're looking for,
- Let's look at the next one, inkscape.x86_64, and that says it's a graphics editor, so that is the package we want.
- So in order to install or remove software,
- To install or remove applications (pakages) we need root privileges, we can use sudo, if that's configured, or we can also just switch to the root user.
- Switch to root using su -, and enter the password for root for this server.

- Then we'll do 'yum install inkscape‘ which prompts is this okay to download this? Yes, it is, so enter 'y' <enter>.
- Package is now installed.
- You can also use 'yum install' with a '-y’, so you don't have to answer 'y' to the question here.
- We'll do this with another piece of software called, gimp.
- And you can see that this has several dependencies, so it not only installs the application, but it installs the other packages that are required for gimp to run properly.
- After the install, let's look at our applications lists or graphics, and you can see that Inkscape and gimp has been installed in our system.
- Let's remove some software with yum 'remove', we'll do this to gimp.

- Similar to install, you could use '-y' to answer yes automatically.
- Do this manually here and it says it has erased that package.
- Let's look applications, graphics, and gimp is no longer showing up at our menu.
- There may be some software that you want to install that's not included in the package manager.
- Let's look for Opera, so you can use yum to install Dropbox or at least directly so we'll go to dropbox.com and download the rpm to the Downloads directory
- We can install an rpm with '-i' for install, 'v' is verbose and 'h' says print some 'hash marks' to show our progress, so we'll just do that here and answer OK to the prompts.

- Dropbox was successfully installed. You can check applications and see there is Dropbox, so yes we installed Dropbox.
- You can also use rpm to do some queries.
- Let’s use 'rpm –qa’ to query all packages or to list all packages that are installed on the system, and these are listed in no particular order.
- We'll sort the list, so it makes easier for us to read 'sort' it and 'pipe' it to ‘less’, so we can look at one page at a time.
- Now we go, away to the end you can see yum, zlib, etcetera.
- You can also ask rpm what file a package belongs to, so the program 'which' is in /usr/bin/which.

- Now to find out which package it belongs to you do

  $ rpm –qf /usr/bin/which

- And it belongs to the which package
- You can also say - show me all the files, and we'll list all the files that are in that package but we'll just give it the package name.

  $ rpm –ql which

- You can see all the files contained in the package.

- Another popular package format is the Debian package format.
- In addition to Debian, distributions like Linux Mint and Ubuntu use their packages, and the Debian based distributions use a package manager called 'APT', the advanceed packaging tool.
- APT is comprised of a few small utilities with the two most commonly used ones being apt-cache and apt-get.

APT - Advanced Packaging Tool

```
apt-cache search string
```

Search for string.

```
apt-get install [-y] package
```

Install package.

- To search for packages, use 'apt-cache search' and the search string and to install the package, use 'apt-get install' and the package name.
- You can use '-y' to automatically answer yes to any of apt-get's questions.
- To remove a package, use 'apt-get remove'.

APT - Advanced Packaging Tool

```
apt-get remove package
```

Remove package, leaving configuration.

```
apt-get purge package
```

Remove package, deleting configuration.

- If there are any configuration files or configuration changes, those will stay on your system with apt-get remove, but if you want to get rid of the configuration in addition to the package, you can use 'apt-get purge package'.
- To show information about a particular package, use the 'apt-cache show' and the package name.
- In addition to the apt utilities, you can use the dpkg command to interact with a package manager.

dpkg

`dpkg -l`	List installed packages.
`dpkg -S /path/to/file`	List file's package.
`dpkg -L package`	List all files in package.
`dpkg -i package.deb`	Install package.

- 'dpkg -l' lists the installed packages.
- 'dpkg -S' and a path to a file will display the package that the file belongs to.
- 'dpkg -L' lists all the files in a package.
- And to install a package, use 'dpkg -i' and the 'deb' file.
- Let’s log into Ubuntu school
- Let's use apt to look for and install some software.
- Look for Inkscape **$ apt-cache search inkscape** and the first result returned there looks like what we want.
- Let's take a look at it with apt-cache show and get some information about the package, its dependencies, it's description.

- So we will install that with apt-get install.
- Remember, in Linux, installing and removing software requires root privileges and can get root privileges by using the sudo command. That's one way, but you can also switch to the root account.
- When the installation is done see if you can find it, sure thing it is there.
- You can also just answer yes to apt's question with a '-y'.
- So we'll go ahead and install gimp this way

  $ apt-get install –y gimp

- And gimp is installed without prompting, so let’s remove it

  $ apt-get remove –y gimp

- Do a ‘which gimp’ no found so gimp is gone

- Let's look at the installing software from a 'deb' file.
- Lets download the '.deb' for Opera.
- Use 'dpkg -i' to do the installation, and Opera is installed.
- We can use 'dpkg -l' to list all the installed packages.
- Also we use 'dpkg –S' and sew which package the 'which' command belongs to. **$ dpkg –S /usr/bin/which**
- It belongs to the Debian utils package.
- So we can look at all the files in the Debian utils package with the '-L' command. **$ dpkg –L debianutils**

Summary

- Packages are used to install software on Linux systems.
- You can manipulate packages with a package manager.
- Two of the most popular package formats are RPM and Debian.
- For RPM based distributions, use the yum and rpm commands to manage packages.
- For Debian based distributions, use apt and dpkg command to manage packages.