aws re/start

Working with Files

Linux Fundamentals

Welcome to Working with Files.

What you will learn

At the core of the lesson

You will learn how to:

- Describe the use of the `hash`, `cksum`, `find`, `grep`, and `diff` commands
- Differentiate hard links from symbolic links
- Compare the `tar`, `gzip`, and `zip` commands

aws re/start

In this lesson, you will learn how to:

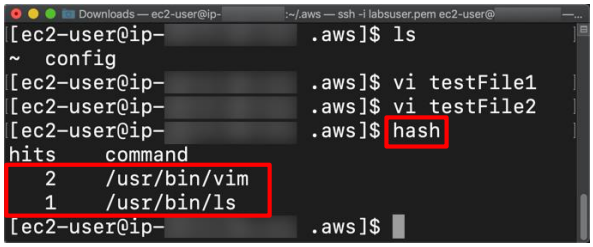
- Describe the use of the hash, cksum, find, grep, and diff commands
- Differentiate hard links from symbolic links
- Compare the tar, gzip, and zip commands

Important commands: `hash`, `cksum`, `find`, `grep`, and `diff`

You begin by defining important commands that are used when working with files.

The hash command

- Displays a list of recently run programs, their location, and the number of times they have run
- Information is maintained by the command in a **hash table**
- Can be used to reset or modify the hash table
- Location information includes the program's full path name
- Syntax: `hash [options] [-p pathName] [options] [commandName ...]`



aws re/start

The `hash` command can be used to view recently run programs, their location in the file system, and the number of times they were run.

Specifically, the `hash` command displays or modifies the remembered location of the program associated with a given command and how many times the command was run. It stores this information in a hash table and provides options to display, reset, or delete the table's content. A program's location information consists of its full path name in the file system.

Syntax: `hash [-lr] [-p pathName] [-dt] [commandName ...]`

Some command options are as follows.

- `-d`: Deletes the location for `commandName` from the hash table
- `-l`: Displays output in a format that can be used as input to another command
- `-p`: Sets `pathName` as the the full path location for `commandName`
- `-r`: Empties the hash table
- `-t`: Displays the location of `commandName`

The `cksum` command

- This command generates a checksum value for a file or stream of data.
- It is used to see whether the file was corrupted during transfer.
- The `cksum` command displays a cyclic redundancy check (CRC) value and the byte count for a file.
- If the file's CRC value is the same before and after a transfer, the file was not corrupted.
- Syntax: **cksum [FileName]**

```
Downloads — ec2-user@ip-          :~ — ssh -i labsuser.pem ec2-user@
[ec2-user@ip-                ~]$ ls -l UserData.txt
-rw-r--r-- 1 root root 553 Jul  8  2020 UserData.txt
[ec2-user@ip-                ~]$ cksum UserData.txt
23850951 553 UserData.txt
[ec2-user@ip-                ~]$
```


aws re/start

The `cksum` command generates a checksum value for a file or stream of data that the user can use to see whether the file was corrupted during transfer. Specifically, it computes and displays a cyclic redundancy check (CRC) value for the file and shows its size in bytes. The CRC value is derived from the contents of the file and is used to validate the file's integrity. If a file's checksum value is the same before and after a transfer, the file was not corrupted.

The `find` command

- The `find` command searches a designated directory for files that match specific criteria
- Can search by:
  - Owner
  - File name
  - File size
  - File modification date
- Can specify which directories to search to narrow the scope
- Can specify an action to take when the file is found
- Syntax: `find [directory to start from][options][what to find]`
  - Example: `find /home/student01 -name fileA.txt`

aws re/start

The `find` command searches a directory and any of its subdirectories for files that match specific criteria. Search criteria can include file name, file type, file size, file owner, and file modification date, and you can search by logical expressions. When searching by name, wildcard characters can be used to match based on a character pattern. In addition, you can specify an action to take on matching files, such as delete or run a specified command on them. The `find` command can also be used with pipe (|) to input the findings into another program.

In the example that is shown, the `find` command returns all files with a name of `fileA.txt` in the `/home/student01` directory and its subdirectories.

`find` options

Some common options for the `find` command include:

Option	Description
`-name <file name>`	Searches by file name
`-iname <file name>`	Searches by file name but ignores case
`-user <user name>`	Searches by file owner
`-type <file type>`	Searches by file type

aws re/start

You can use the options for the `find` command to specify the type of search to perform: for example, by name, owner, or file type. You can also specify actions to take on the returned file matches.

Actions that are used with the `find` command

Write the location of searched files to an output file

Run command on searched files

Delete searched files

Option	Description
`-fprint`	Write output to a file
`-exec`	Run a command
`-delete`	Delete the file

aws re/start

The `find` command can perform actions on the files that match the search criteria. Some examples are as follows.

- `-fprint` *`fileName`*: Writes the output of the command to the specified file name
- `-exec` *`commandName`*: Runs the specified command on the returned file or files
- `-delete`: Deletes the returned file or files

Examples of the `find` command

Example 1:	`find . -iname fileA.txt`
Example 2:	`find /home/student01 -user student01`
Example 3:	`find /home/student01 -name *.jpg`
Example 4:	`find /etc -iname "*.conf" -mtime 7`

aws re/start

This slide shows examples of the `find` command:

- The first example shows how to search for a file that is named `fileA.txt` and starts in the current directory. The search should ignore the case of the letters in the file name pattern.
- The second example shows how to search for files that `student01` owns, which start in the `/home/student01` directory.
- The third example shows how to search for files with a file name extension of `.jpg` that start in the `/home/student01` directory.
- The fourth example shows how to search for files with a file name extension of `.conf` that start in the `/etc` directory. The search should be case insensitive and return only those matching files that were modified exactly 168 (**7** x 24) hours ago.

Demonstration: The `find` Command

The `find` command searches the file system to display files that you want to search for. You can search based on many criteria, including file name, owner, and file size. In this demonstration, the instructor will use the `find` command to display several files based on different criteria.

```
[ec2-user]$ touch CompanyA/myFile.csv
[ec2-user]$ find CompanyA/ -name *.csv
CompanyA/Management/Sections.csv
CompanyA/Management/Promotions.csv
CompanyA/Employees/Schedules.csv
CompanyA/Finance/Salary.csv
CompanyA/HR/Managers.csv
CompanyA/HR/Assessments.csv
CompanyA/SharedFolders/myFile.csv
CompanyA/myFile.csv
[ec2-user]$ find CompanyA/ -maxdepth 1 -name *.csv
CompanyA/myFile.csv
[ec2-user]$
```


aws re/start

In this demonstration, the instructor will show various forms of the `find` command. Some examples are as follows.

- Find files that have a specific name extension:
  `find <startingDirectoryName> -name <*.extension>`
- Find files that have a specific name extension up to a subdirectory depth of 1:
  `find < startingDirectoryName > -maxdepth 1 -name <*.extension>`

The `grep` command

- Searches the contents of a file for a particular text pattern or string and displays each occurrence
- Can also search files in a directory
- Provides options to control the search behavior and output
- Syntax: `grep <text pattern or string> <where to search>`

Option	Description
`-i`	Ignore case
`-r`	Recursive searches
`-l`	List only file names
`-n`	Display line number
`-c`	Count of matching lines
`--files-with-matches`	Names of files that contain selected lines are written to standard output

aws re/start

The `grep` command searches a file or a directory for a particular text pattern or string and displays each occurrence. You can control the search behavior and output through various options.

Example: The `grep` command

Matches patterns of text to file contents

Example: `grep fail /var/log/secure` matches entries in the `/var/log/secure` log file that contain ***fail***

```
[ec2-user]$ sudo grep fail /var/log/secure
Aug 23 07:45:25 ip-10-0-10-69 sshd[6540]: error: AuthorizedKeysCommand /opt/aws/
bin/eic_run_authorized_keys ec2-user SHA256:UhBsc766SvcR9/3AqyunB0luvyViHgSO4vvP
7Y1YUC0 failed. status 22
Aug 23 07:45:25 ip-10-0-10-69 sshd[6540]: error: AuthorizedKeysCommand /opt/aws/
bin/eic_run_authorized_keys ec2-user SHA256:UhBsc766SvcR9/3AqyunB0luvyViHgSO4vvP
7Y1YUC0 failed. status 22
Aug 23 08:02:17 ip-10-0-10-69 sudo: ec2-user : TTY=pts/0 ; PWD=/home/ec2-user ;
USER=root ; COMMAND=/bin/grep fail /var/log/secure
Aug 23 08:48:48 ip-10-0-10-69 sshd[3492]: error: AuthorizedKeysCommand /opt/aws/
bin/eic_run_authorized_keys ec2-user SHA256:UhBsc766SvcR9/3AqyunB0luvyViHgSO4vvP
7Y1YUC0 failed. status 22
Aug 23 08:48:48 ip-10-0-10-69 sshd[3492]: error: AuthorizedKeysCommand /opt/aws/
bin/eic_run_authorized_keys ec2-user SHA256:UhBsc766SvcR9/3AqyunB0luvyViHgSO4vvP
7Y1YUC0 failed. status 22
```


aws re/start

In this example, the `grep` command matches the pattern text *fail* to the file contents.

`find` and `grep` comparison

	`find`	`grep`
Description	Searches files based on given criteria	Searches file content for a given string or text pattern
Use	Is used to locate files	Is used to find a string in a file
Output	Returns file names	Returns occurrences of a searched string

aws re/start

The `find` and `grep` commands have one main difference. The `find` command is used to locate *files* that match specified criteria, and the `grep` command is used to find a *string* in a file.

The `diff` command

- Compares files line by line and displays the differences
- Valuable for comparing two files
- Output is called a **diff**
- Syntax: `diff [options] File1 File2`

```
[ec2-user]$ echo "LA, NYC, Tokyo, Taipei" > cities.txt
[ec2-user]$ echo "LA, NYC, Tokyo, Taipei, Paris" > cities2.txt
[ec2-user]$ diff cities.txt cities2.txt
1c1
< LA, NYC, Tokyo, Taipei
---
> LA, NYC, Tokyo, Taipei, Paris
[ec2-user]$
```


aws re/start

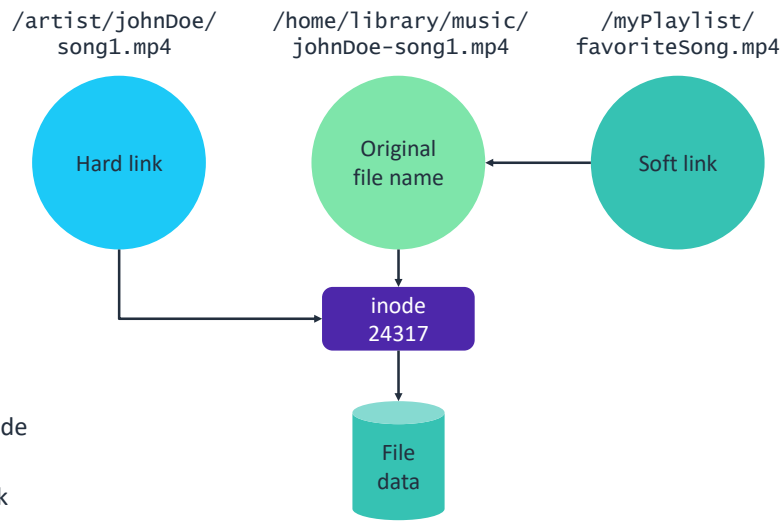
The `diff` command compares two files line by line and displays the differences between the two files.

Links

Links are another way to refer to files.

Links and inode

- You can use links to refer to the same file by using different names.
- You can use links to access the same file from more than one location in the file system.
- Every file has an **inode** object that uniquely identifies its data location and attributes.
- Two types of links:
  - **Hard** link: Points to a file's inode
  - **Symbolic** link: Points to the original file name or a hard link



aws re/start

You can use a link to refer to a given file by using different names. By creating multiple links for the same file, you can access it from more than one location in the file system.

Linux has two types of links:

- Hard link: Every file has an object that is called inode, which stores the file's disk block locations and attributes. An inode is identified with a unique number. A hard link is a pointer to a file's inode.
- Symbolic link: Also known as a soft link or symlink, a symbolic link points to the original file name or a hard link.

In the example, a music file was originally created with a complete path name of `/home/library/music/johnDoe-song1.mp4` and an inode that was identified as `24317`. A hard link was then created to allow access to the file through a directory structure that categorized the song by its artist's name: `/artist/johnDoe/song1.mp4`. A soft link was also created to identify it as the favorite song in a playlist directory: `/myPlaylist/favoriteSong.mp4`.

Hard link

- Points to the original file's **inode**
- Cannot reference a directory
- If the original file is deleted, its data still exists until the hard link is deleted
- Syntax for creating a hard link: `ln [options] [originalFileName] [linkName]`
  - Example: `ln /home/userA/dev-project.txt /devprojects/dev-data.txt`

```
[ec2-user]$ ls
CompanyA  Documents
[ec2-user]$ cat Documents/file1
I am file1
[ec2-user]$ ln Documents/file1 fileLink
[ec2-user]$ ls -l
total 4
drwxr-xr-x 8 ec2-user root     115 Aug 23 08:54 CompanyA
drwxrwxr-x 2 ec2-user ec2-user  45 Aug 23 08:38 Documents
-rw-rw-r-- 2 ec2-user ec2-user  11 Aug 23 08:59 fileLink
[ec2-user]$ cat fileLink
I am file1
[ec2-user]$
```


aws re/start

Some important things to keep in mind with hard links include the following:

- No visual difference exists between a hard link and a file in a directory listing. When you use the `ls` command to list a directory, no special indicators show that an entry is a hard link.
- If the original file is deleted, its contents still exist until the hard link is deleted.

The screen capture shows how to create a hard link that is named `fileA` for the file that is named `file1`.

Symbolic link

- Points to an **original file name** or a **hard link**
- Can point to a directory
- If the original file is deleted, the soft link is broken until you create a new file with the original name
- Syntax for creating a symbolic link: `ln -s [options] [originalFileName] [linkName]`

```
[ec2-user]$ ln -s Documents/file1 sym-fileLink
[ec2-user]$ ls -l
total 4
drwxr-xr-x 8 ec2-user root     115 Aug 23 08:54 CompanyA
drwxrwxr-x 2 ec2-user ec2-user  45 Aug 23 08:38 Documents
-rw-rw-r-- 2 ec2-user ec2-user  11 Aug 23 08:59 fileLink
lrwxrwxrwx 1 ec2-user ec2-user  15 Aug 23 09:02 sym-fileLink -> Documents/file1
[ec2-user]$ cat sym-fileLink
I am file1
[ec2-user]$
```


aws re/start

Some important things to keep in mind with symbolic links include the following:

- Technically, you can also make a symbolic link point to a hard link. A hard link is like the original file name because it points to the file's inode.
- A visual difference exists between a symbolic link and a file in a directory listing. When you use the `ls` command to list a directory, a symbolic link points to its original file name.
- If the original file is deleted, the symbolic link is broken and has no value. If you create a new file with the original name, the symbolic link will work as expected again.

The screen capture shows how to create a symbolic link that is named `sym-fileA` for the file that is named `fileA`.

Demonstration: Links

Sometimes it is useful to be able to access file content from two different locations. In this demonstration, the instructor will use a soft link to achieve this goal.

```
[ec2-user]$ touch CompanyA/SharedFolders/myFile.csv
[ec2-user]$ ln -s CompanyA/SharedFolders/myFile.csv sym-myfile
[ec2-user]$ echo "I can use the link to access the file" > sym-fileLink
[ec2-user]$ cat sym-fileLink
I can use the link to access the file
[ec2-user]$
```


aws re/start

In this demonstration, the instructor will show you how to create a soft link by using the link command: `ln –s [originalFileName] [linkName]`

Linux compression utilities

Linux has a number of utilities that you can use to bundle files together.

The `tar` command

- Bundles a collection of files into a single archive file for easier copying or downloading
- Created bundle is called a *tarball*
- Contents of an archive file can optionally be compressed
- The command is also used to unbundle an archive file
- For example:
  - To bundle and compress `file1`, `file2`, and `file3` into a file called `tarball.tar`:

    `tar -cvf tarball.tar file1 file2 file3`
  - To unbundle or extract files from the tarball:

    `tar -xf tarball.tar`

```
[ec2-user]$ tar -cvf tarFile.tar Documents
Documents/
Documents/file1
Documents/file2
Documents/file3
[ec2-user]$ ls
CompanyA  Documents  tarFile.tar
[ec2-user]$
```

```
[ec2-user]$ rm -rf Documents/
[ec2-user]$ ls
CompanyA  tarFile.tar
[ec2-user]$ tar -xf tarFile.tar
[ec2-user]$ ls
CompanyA  Documents  tarFile.tar
[ec2-user]$ ls Documents/
file1  file2  file3
[ec2-user]$
```


aws re/start

The `tar` command is used for the following:

- File downloads
- Large numbers of files to copy or move
- Internet downloads, including software

Common `tar` options

Option	Description
`-x`	Extracts the contents of a tarball
`-z`	Compresses the contents of a tarball by using the `gzip` utility
`-f`	Specifies the name of the tarball
`-v`	Produces verbose output by showing file names while the tarball is processed

aws re/start

Some `tar` options are mandatory when extracting.

- `-x`: Extracts a tarball (an archive file)
- `-z`: Uses `gzip` (see later in this lesson)
- `-f`: Provides the name of the tarball
- `-v`: Shows the progress as the tarball is created

The `gzip` command

- Compresses or decompresses files, including tarballs
- Examples:
  - To compress a tarball:

    ```
    gzip salesdata.tar
    ```
  - To decompress a tarball:

    ```
    gzip -d salesdata.tar.gz
    ```

```
[ec2-user]$ ls -l tarFile.tar
-rw-rw-r-- 1 ec2-user ec2-user 10240 Aug 23 09:07 tarFile.tar
[ec2-user]$ gzip tarFile.tar
[ec2-user]$ ls -l tarFile.tar.gz
-rw-rw-r-- 1 ec2-user ec2-user 242 Aug 23 09:07 tarFile.tar.gz
[ec2-user]$
```


aws re/start

Use the gzip command to compress or decompress files, including tarballs:

- Common with internet downloads
- Common with archiving log files
- Common with archiving old data

Files can be both tarred and gzipped, and files that are created with the gzip command will have the `.gz` extension.

The `zip` and `unzip` commands

- The `zip` command is used as a compression tool.
  - Syntax: `zip -r [FolderName]`
- The `unzip` command is used as an extraction tool.
  - Syntax: `unzip [FolderName].zip`

aws re/start

When a folder or directory is zipped, you can use the recursion option `-r` to include the contents of its subdirectories.

By default, the name of the created compressed file will have a `.zip` extension.

Checkpoint questions

Is it possible to find the files that a specific user owns by using the `find` command?

Which command bundles many files into one file?

You've been asked to ensure that a file is transferred to another user with an absolute guarantee that it has not been damaged or altered in the transfer.
How can you accomplish that task by using the information in this lesson?

aws re/start

Answers:

1. Yes, by using the `-user` option
2. The `tar` command
3. Before transferring the file, run the `cksum` command on it and record the result. After transmission, run `cksum` again, and compare the new result with the original one. If the results match, the file was not altered during transfer.

Key takeaways

Several commands make it easier to work with files in Linux. You learned the following commands:

- `hash`: Is used to see a history of programs and commands that are run from the command line
- `cksum`: Verifies that a file has not changed
- `find`: Searches for files by using criteria such as the file name, the size, and the owner
- `grep`: Searches a file's contents for a text pattern
- `diff`: Is used to quickly see the difference between two files
- `ln`: Creates pointers to a given file
- `tar`: Bundles multiple files into one file
- `gzip`: Compresses a file's size
- `zip`: Compresses the contents of a file
- `unzip`: Decompresses the contents of a file

aws re/start

The commands introduced in this lesson are summarized as follows:

- `hash:` Is used to see a history of programs and commands that are run from the command line
- `cksum:` Verifies that a file has not changed
- `find:` Searches for files by using criteria such as the file name, the size, and the owner
- `grep:` Searches a file's contents for a text pattern
- `diff:` Is used to quickly see the difference between two files
- **`ln:`** Creates pointers to a given file
- `tar:` Bundles multiple files into one file
- `gzip:` Compresses a file's size
- `zip:` Compresses the contents of a file
- `unzip:` Decompresses the contents of a file

Thank you

aws re/start

Thank you.