find

- **find [directory...] [criteria...]**
- Searches directory trees in real-time
  - Slower but more accurate than **locate**
  - CWD is used if no starting directory given
  - All files are matched if no criteria given
- Can execute commands on found files
- May only search directories where the user has read and execute permission

redhat 13-5

Unlike **locate**, **find** will do a real time search of the machines file system to find files that match the criteria of the command line arguments. But realize that since **find** is looking at files in the file system as your user account, you must have read and execute permission on a directory to examine its content.

find requires an argument of what directory it should start finding files in. So if you only wanted to find in a user student's home directory you would give **find** a starting directory of `/home/student`. If you would like **find** to look over the entire system, you would provide a starting directory of `/`.

find has a huge amount of options that can be provided to describe exactly what kind of file should be found. You can search based on file name, file size, last modified time stamp, inode number, and many, many more.

The ability to locate files based on almost any combination of criteria is powerful, but it is only part of what makes **find** such a useful tool. **find** can also execute arbitrary commands on any of the files that it matches, greatly simplifying some otherwise mind-numingly repetitive tasks.

Commands can be run on found files by using the **-ok** and **-exec** options. If **-ok** is used, you will be prompted for each file before the specified command is run. If **-exec** is used, the command will be run on all matching files with no prompts for confirmation. When specifying the command to run, the name of the found file can be represented with `{}` and the command must be terminated with a `Space` followed by `\;`. For example, the following command would fix any files that are writable by the `other` group, which is considered a security risk.

```
[root@stationX ~]# find / -perm -002 -exec chmod o-w {} \;
```

Basic *find* Examples

- **find -name snow.png**
  - Search for files named snow.png
- **find -iname snow.png**
  - Case-insensitive search for files named snow.png, Snow.png, SNOW.PNG, etc
- **find -user joe -group joe**
  - Search for files owned by the user *joe* and the group *joe*

redhat 13-6

Finding files based on their name will, unlike **locate**, look for files that are named the exact string passed to the **find** command. That is to say, if a **find** command was used like:

```
[student@stationX ~]$ find / -name .png
```

findwould only return the files that were named **.png**, not files that contained in their name the string **.png**. Fortunately, you can use shell wild cards with **find**, but they must be quoted. As an example:

```
[student@stationX ~]$ find / -name "*.png"
```

would find all files on the system that have **.png** as the end of their name. Related to **-name** is the **-iname** option which works the same way as **-name** but performs a case-insensitive search.

The **-regex** option in **find** does not work quite the way one would expect. **-regex** applies the regular expression to the name of the file, including the absolute path to the file. So in the above example, the regular expression **".*W.*\.png"** will match all files that have a capital **W** and **.png** in their names. If we had instead used **"W.*\.png"**, we would get no results because the full path to our file name will always begin with a / and our provided regular expression indicates we are only interested in files whose full path name begins with a **W**, something that can never be true.

find and Logical Operators

- Criteria are ANDed together by default.
- Can be OR'd or negated with **-o** and **-not**
- Parentheses can be used to determine logic order, but must be escaped in bash.
  - **find -user joe -not -group joe**
  - **find -user joe -o -user jane**
  - **find -not \(-user joe -o -user jane \)**

redhat 13-7

By default, if multiple criteria are given to **find**, they must all apply to a file for it to be considered a match. For example, the following will only match files whose names end in `.png` AND are owned by the user `student`:

```
[student@stationX ~]$find / -name "*.png" -user student
```

This behavior can be overridden with the `-o` option. The following command will match files whose names end in .png OR are owned by `student`:

```
[student@stationX ~]$find / -name "*.png" -o -user student
```

find also includes a logical "not" operator. Options preceded with a **!** or the **-not** option will cause find to look for the opposite of the given criterion. So the following command will match files whose names end in `.png` and are NOT owned by `student`.

```
[student@stationX ~]$find / -name "*.png" -not -user student
```

With the addition of logical operators, the question of operator precedence is raised. Logical ANDs have a higher priority than logical ORs, and logical NOTs have the highest priority of all. To force precedence of an expression, you can enclose options that should be grouped together in parentheses. Make sure to put spaces after the \(and before the \) or find will not work as expected. The find man page has more details about this in the OPERATORS section.

find and Permissions

- Can match ownership by name or id
  - **find / -user joe -o -uid 500**
- Can match octal or symbolic permissions
  - **find -perm 755** matches if mode is *exactly* 755
  - **find -perm +222** matches if *anyone* can write
  - **find -perm -222** matches if *everyone* can write
  - **find -perm -002** matches if other can write

redhat 13-8

find can search for files based on their ownership or permissions. Useful options when searching by owner are:

-user and **-group**: Search by name of user or group

-uid and **-gid** Search by user ID or group ID

The **-perm** option is used to look for files with a particular set of permissions. Permissions can be described as octal values (some combination of 4, 2 and 1 for read, write and execute, respectively) or using symbolic notation (eg `u+w` for "user has write access"). Permissions should be preceded by a + or `-` sign. The meanings of these operators are slightly different depending on whether you are using numeric or symbolic notation.

A numeric permission preceded by + will match files that have at least one bit (user, group or other) for that permission set. So, for example, a file with permissions `r--r--r--` would not match +222, but one with `rw-r--r--` would. A `-` sign before a permission means that all three instances of that bit must be on, so neither of the previous examples would match but something like `rw-rw-rw-` would.

So, to use a more complex example, the following command would match any file for which the user has read, write and execute permissions, the group has read permissions and others have read-only access:

```
[student@stationX ~]$ find -perm 764
```

To match files for which the user has *at least*read, write and execute permissions, the group has *at least* read permissions and others have *at least* read access:

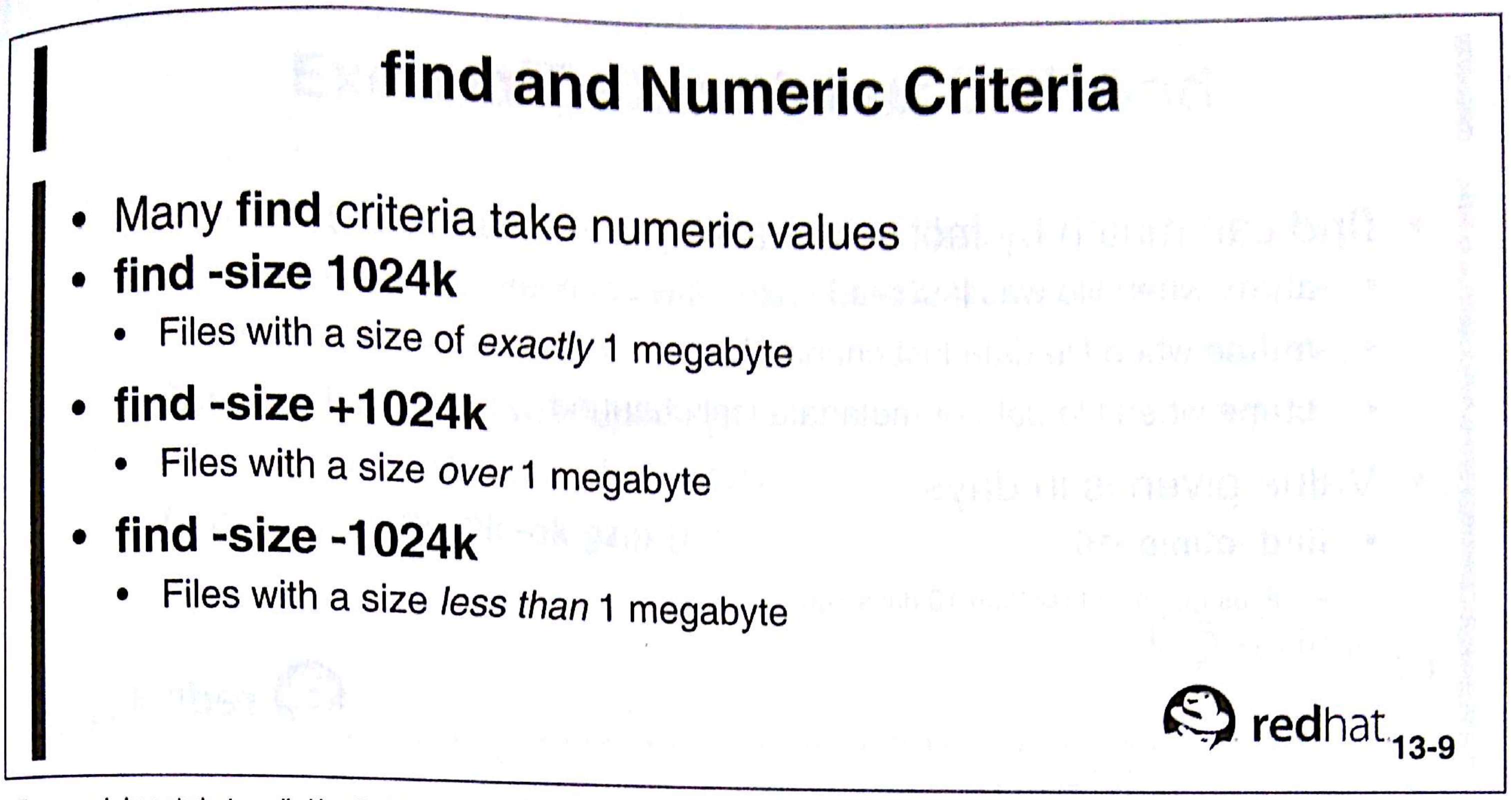
```
[student@stationX ~]$ find -perm -764
```

And to match files for which the user has read, write and execute permissions, *or* the group has at least read permissions *or* others have at least read access:

```
[student@stationX ~]$ find -perm +764
```

When used with + or `-`, a value of `0` works like a wildcard, since it means "a permission of at least nothing. Thus, the following command would match any file for which others have at least read access:

```
[student@stationX ~]$ find -perm -004
```



find and Numeric Criteria

- Many **find** criteria take numeric values
- **find -size 1024k**
  - Files with a size of *exactly* 1 megabyte
- **find -size +1024k**
  - Files with a size *over* 1 megabyte
- **find -size -1024k**
  - Files with a size *less than* 1 megabyte

find can search your system for files that comply with certain numeric criteria such as the size of the file (**-size**), the number of links to the file (**-links**) the date of the last change to the file's data, (**-mtime**), date of the last change to the file's metadata (**-ctime**) or the date of the last time the file was read (**-atime**). All of these criteria accept a numeric value. When you provide numeric values to find you can look for an exact match, more than the number, or less than the number. For example:

```
[student@stationX ~]$ find / -atime 5
```

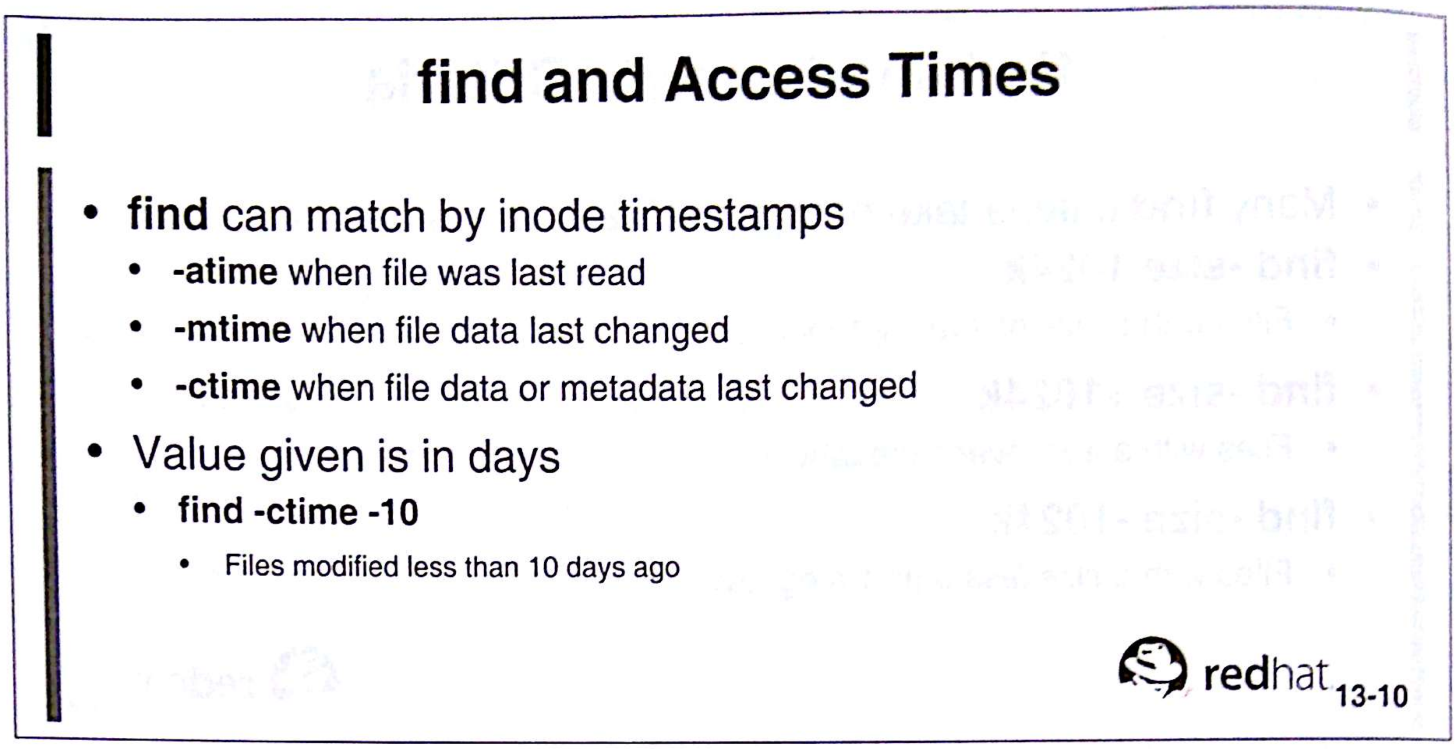
looks for files on the system whose last accessed time stamp (see the next slide for more information on **-atime**) is exactly five days ago, whereas:

```
[student@stationX ~]$ find / -atime +5
```

will find files whose last accessed time stamp is more than five days ago. Lastly:

```
[student@stationX ~]$ find / -atime -5
```

will print the list of files whose last accessed time stamp is less than five days ago.



find and Access Times

- **find** can match by inode timestamps
  - **-atime** when file was last read
  - **-mtime** when file data last changed
  - **-ctime** when file data or metadata last changed
- Value given is in days
  - **find -ctime -10**
    - Files modified less than 10 days ago

redhat 13-10

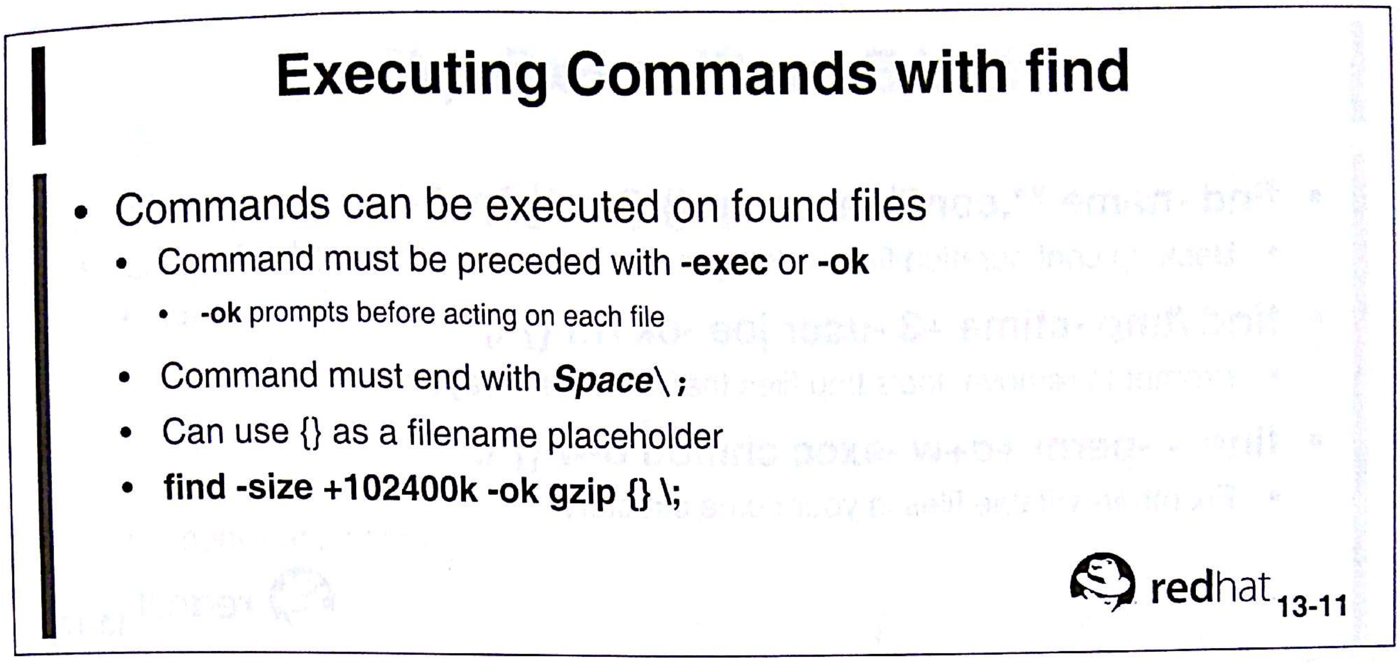
Examples of **find** timestamp-matching criteria are given in the notes of the previous slide. While the values passed to **-atime**, **-ctime** and **-mtime** are measured in days, there are also corresponding criteria that perform searches in minutes: **-amin**, **-cmin** and **-mmin**. You can even match access times relative to the timestamps of other files using **-anewer**, **-cnewer** and **-newer**, which tests mtime. For example:

```
[student@stationX ~]$ find -newer recent_file.txt
```

Would list all files with mtimes more recent than that of `recent_file.txt`. Note that there is no **-older** argument. To match files older than `recent_file.txt` you would simply negate the **-newer** criteria:

```
[student@stationX ~]$ find -not -newer recent_file.txt
```

Remember that the metadata, including all three timestamps, for a file can be manually examined using the **stat** command.



Executing Commands with find

- Commands can be executed on found files
  - Command must be preceded with **-exec** or **-ok**
    - **-ok** prompts before acting on each file
  - Command must end with ***Space*\ ;**
  - Can use {} as a filename placeholder
  - **find -size +102400k -ok gzip {} \;**

Using the **-exec** or **-ok** options with find will cause **find** to execute a command once for each file that matches the given criteria. This is commonly used for things like removing files or renaming files to have a certain extension. Be extremely careful when using **-exec** because it may perform the action on many files (remember that find recurses through subdirectories) and it does not ask for confirmation. Remember that running the search without **-exec** will list all matches, thus allowing you to preview which files will be acted upon. Alternately you can use the **-ok** option, which causes **find** to ask for each file.

The reason that the commands given with **-exec** and **-ok** must end in a **\;** is because find uses **;** as the delimiting character. Unfortunately **;** is also a delimiting character for the shell so we must prevent bash from interpreting it. When a character is prepended with a backslash (\), **bash** is instructed to treat it literally, so typing **\;** at the **bash** command prompt will send **;** to **find** after **bash** has done its interpretation.

find Execution Examples

- **find -name "*.conf" -exec cp {} {}.orig \;**
  - Back up configuration files, adding a `.orig` extension
- **find /tmp -ctime +3 -user joe -ok rm {} \;**
  - Prompt to remove Joe's tmp files that are over 3 days old
- **find ~ -perm +o+w -exec chmod o-w {} \;**
  - Fix other-writable files in your home directory

redhat 13-12

You can use **find** to execute any command you would like, and if you use the {} as filename place holders, the **find** command will put the file name of a file it has found in place of {} and execute the command.