Find, Locate, and Grep

==========

FIND

Find is a linux search tool that can be used to find a variety of files based on the given criteria.

Basic find

find targetdirectory -name stringtofind

```
find / -name hello.txt
```

wildcards

```
find / -name "*.txt"
```

Discard errors (ie permission denied)

```
find / -name "*.txt" 2>/dev/null
```

Find files with specific permissions

Find files with read,write, and execute permissions

```
find / -perm +rwx
```

Find files by size

Find files that are greater than 2MB:

```
find / -size +2M
```

Find files that are less than 2MB:

```
find / -size -2M
```

Find files that are exactly 2MB:

```
find / -size 2M
```

Find files by owner/group

Find files based on group

```
find / -group groupname
```

Find files based on owner

```
find / -user username
```

==========

LOCATE

Locate is similar to find in its functionality, however, there are some important distinctions:

- Locate keeps it's own database of files on a system
- Locate is less disk IO intensive since it doesn't have to scan the whole hard drive for files, instead it references it's own database.
- Locate isn't installed by default on most systems, the package name is mlocate. Once installed, you'll need to run the updated command to update locate's database - this should really be updated every time you use the command

---

Locate file based on name

```
locate test.txt
```

==========

GREP

-i - case insensitive

As with all of Linux, grep is case sensitive. The -i flag can be passed to ignore case type.

-v -exclude string

The -v flag is used to exclude a string from an output.

| - or statement

The below will search for string1 or string2 within a file, if both are found then both will be outputted.

```
grep 'string1|string2' filename
```

-r - search files recursively

```
grep -r hello /
```

-A & -B (before and after)

Sometimes you might want to search for a string, and see the lines before/after that string.

The below will show the 2 lines following the string

```
grep -A2 string filename
```

The below will show the 2 lines before the string

```
grep -B2 string filename
```

You can also combine these to see the lines before and after

```
grep -A2 -B2 string filename
```

==========

-E or egrep - extended regular expression

Regular expression is essentially the methodology that we can use to manipulate grep to find advanced string patterns.

Special characters

When trying to grep for special characters, you need to make sure to 'escape' those characters, this is done by proceeding special characters with a \:

```
grep -E year\'s
```

Beginning and end of line

line begins with

The below example would show any lines beginning with the character '1'

```
grep -E "^1" filename
```

line ends with

The below example would show any lines ending with the character '1'

```
grep -e "1$" filename
```

Ranges

grep interprets ranges that are defined through square brackets [].

line begins with 1 and is followed by numbers in the 0-2 range:

```
grep -E "^1[0-2]" filename
```

We can also search for ranges of letters

The below command would search for the letter b, proceeded by any letter in the specified range, followed by the letter g:

```
grep -E "b[aeiou]g" filename
```

We can also search a range of letters like this:

```
grep -E "b[a-z]g" filename
```

You can also combine ranges

```
grep -E "b[a-z,A-Z]g" filename
```

Wildcards

There are a number of wildcard options available to use in egrep.

. - any single character

```
grep -e "c.t" filename
```

- Matches: "cat", "cot", "cut", etc.
- Does not match: "ct", "caat"

* - matches zero or more occurrences of the preceding character

```
grep -e "g*d" filename
```

- Matches: "gd", "god", "good", "goood"
- Does not match: "go", "goooo", "gdo", "goddy"

.* - match zero or more of any character

```
grep -e "a.*b" filename
```

- Matches: "ab", "acb", "axyzb", "a123b"
- Does not match: "a b", "ab ", "acbd", "a"

Revision #6
Created 2024-05-24 19:00:03 UTC by Daniel
Updated 2024-05-26 20:13:18 UTC by Daniel