csce215 — UNIX/Linux Fundamentals
Spring 2022 — Lecture Notes: Where are my keys?

This document contains slides from the lecture, formatted to be suitable for printing or individual reading, and with some supplemental explanations added. It is intended as a supplement to, rather than a replacement for, the lectures themselves — you should not expect the notes to be self-contained or complete on their own.

(6.1) *Last time*

Last time we learned about a tool called **make**, whose job is to help keep files up-to-date when their dependencies change.

Today, we will learn about some tools for **finding things**.

- Several commands to **look for files** with certain properties, including `find` and `locate`.
- Some more details about the `grep` command, whose job is to **look within files** for certain patterns.
- **Regular expressions**, which can be used in many contexts to describe patterns to search for in text.

(6.2) *Finding files*

`find` 
Search for and act on files in or below a directory.

By default, `find` lists all files and directories.

Example: *Find files and directories in* `/usr/share/dict`.

```
$ find /usr/share/dict
/usr/share/dict
/usr/share/dict/cracklib-small
/usr/share/dict/words
/usr/share/dict/README.select-wordlist
/usr/share/dict/words.pre-dictionaries-common
/usr/share/dict/american-english
/usr/share/dict/british-english
```

(6.3) *Find: Tests*

We can add **tests** to filter out only certain files and directories.

Use `-name` to look for files and directories whose names contain a certain string or wildcard pattern.

Example: *Find files and directories in* `/usr/share/dict` *whose name contains '*`english`*'.*

```
$ find /usr/share/dict -name *english*
/usr/share/dict/american-english
/usr/share/dict/british-english
```

Use `-type d` to look for only directories.

Example: *Find directories (not files) in* `/usr/share`*. Show only the first 5.*

```
$ find /usr/share -type d | head -n 5
/usr/share
/usr/share/zoneinfo
/usr/share/zoneinfo/Etc
/usr/share/zoneinfo/Arctic
/usr/share/zoneinfo/Indian
```

Use `-type f` to look for only files.

Example: *Find files (not directories) in* `/usr/share/dict`*.*

```
$ find /usr/share/dict -type f
/usr/share/dict/cracklib-small
/usr/share/dict/README.select-wordlist
/usr/share/dict/american-english
/usr/share/dict/british-english
```

There are lots of other tests: modification times, file sizes, permissions, etc. See the `man` page for find.

(6.4) *Find: Actions*

We can add **actions** to run commands on each thing that is found.

Use `-print` to print the name of each match.

Example: *Find files in* `/usr/share/dict` *and print their names.*

```
$ find /usr/share/dict -type f -print
/usr/share/dict/cracklib-small
/usr/share/dict/README.select-wordlist
/usr/share/dict/american-english
/usr/share/dict/british-english
```

If you don't give an action, `find` assumes `-print` by default.

Use `-delete` to delete each matching file.

Example: *Find files in* `/usr/share/dict` *and* (*try to*) *delete them.*

```
$ find /usr/share/dict -type f -delete
find: cannot delete /usr/share/dict/cracklib-small: Permission denied
find: cannot delete /usr/share/dict/README.select-wordlist: Permission denied
find: cannot delete /usr/share/dict/american-english: Permission denied
find: cannot delete /usr/share/dict/british-english: Permission denied
```

Use `-exec` to execute a command for each match.

- `{}` for the name of the match
- `\;` to end the command

Example: *Find files in* `/usr/share/dict` *and show the first line of each one.*

```
$ find /usr/share/dict -type f -exec head {} -n1 \;
007bond
Please use 'select-default-wordlist' superuser script
A
A
```

There are a few other actions, mostly for printing information or running commands. See the `man` page.

(6.5) *Regular expressions*

A **regular expression** ('regex') is

- a **sequence of characters** that
- describes a **set of strings**.

Main idea: A regular expression is a **pattern** that **matches** certain strings.

For example, the regular expression

```
[rR]\d-..
```

matches the string

```
R2-D2
```

(and many other strings), but does not match the string

```
C3-PO
```

(6.6) *Who cares?*

Regular expressions appear in a number of different places.

In tools you've seen already:

- In `vim`, the `/` command for search (and replace) accepts a regular expression.
- In `grep`, the pattern we search for can be a regular expression.

Your favorite programming language supports regular expressions too!

- In Python, use the `re` module.
- In Perl, use the `=~` operator.
- In Java, use the `java.util.regex` package.
- In C++, use the `regex` library.

(6.7) *A quick clarification*

Remember **wildcards** and **brace expansion**?

- `??.*`
- `*V*7*`

These play a **somewhat similar role** to regular expressions, but are used in **different contexts**.

They also **work differently** and are generally **less powerful**.

(6.8) *Regular expressions in* `grep`

We've seen `grep`, which looks for lines that match a given pattern, before.

`grep` 
Find lines that match a pattern.
`-i` case insensitive
`-v` find lines that *don't* match
`-E` interpret special characters in patterns as regex operators 
`-r` search recursively in subdirectories 

When the pattern is more than just one specific string, use `-E` and use **single quotes** around the pattern.

Example: *See words with a* `q` *followed by something other than a* `u`.

```
$ grep -E 'q[^u]' /usr/share/dict/words
Chongqing
Chongqing's
Compaq's
Iqaluit
Iqaluit's
Iqbal
Iqbal's
Iraq's
Iraqi
Iraqi's
Iraqis
Qiqihar
Qiqihar's
Urumqi
Urumqi's
```

(6.9) *Locating files*

`locate`
List files on the system matching a pattern. `--regex` interpret the pattern as a regular expression

Example: *Find* `-dev` *files for Python 3.4, 3.6, or 3.8:*

```
$ locate --regex 'python3.[468]-dev'
/usr/share/doc/libpython3.8-dev
/usr/share/doc/python3.8-dev
/usr/share/lintian/overrides/libpython3.8-dev
/var/lib/dpkg/info/libpython3.8-dev:amd64.list
/var/lib/dpkg/info/libpython3.8-dev:amd64.md5sums
/var/lib/dpkg/info/python3.8-dev.list
/var/lib/dpkg/info/python3.8-dev.md5sums
```

(6.10) *The simplest regular expressions*

Most characters match themselves.

Example: *Match the word* `the`.

```
the
  ⇓
When the frost is on the punkin
```

A dot matches any character.

Example: *Match the letters* `fo`, *followed by any two characters, followed by the letters* `er`.

```
fo..er
  ⇓
and the fodder's in the shock
```

(6.11) Character classes

Use square brackets to give a **character class**.

Example: *Match the letter* `y` *followed by two vowels.*

```
y[aeiouAEIOU][aeiouAEIOU]
  ⇓
And you hear the kyouck and gobble
```

Use a dash inside the brackets to ask for any character in a given **range**.

Example: *Match* `u`, `v`, `w`, `x`, `y`, *or* `z`.

```
[u-z]
  ⇓
of the struttin' turkey-cock
```

Use a caret inside the brackets to **negate** the class.

Example: *Match any character that is not a vowel.*

```
[^aeiouAEIOU]
  ⇓
And the clackin' of the guineys
```

(6.12) *Repetition*

Use a question mark for **zero or one** of the previous.

Example: *Match either* `the` *or* `he`.

```
t?he
  ⇓
and the cluckin' of the hens
```

Use a star for **zero or more** of the previous.

Example: *Match an* `r`*, followed by zero or more* `o`*'s.*

```
ro*
  ⇓
And the rooster's hallylooyer
```

Use a plus for **one or more** of the previous.

Example: *Match one or more instances of* `t` *followed by any two characters.*

```
(t..)+
  ⇓
as he tiptoes on the fence
```

(6.13) *Groups*

Use parentheses to form groups, which are useful when combined with repetition operators.

Example: *Match one or more instances of vowel followed by non-vowel.*

```
([AEIUOaeiuo][^AEIUOaeiou])+
  ⇓
O, it's then's the times a feller
```

(6.14) *Either or*

Use a pipe for either of two choices.

Example: *Match either one or more* `e`*'s or the word* `his`.

```
e+|his
  ⇓
is a-feelin' at his best
```

(6.15) *Anchors*

Use a caret to match the start of the line.

Example: *Match one or more letters at the start of a line.*

```
^[A-Za-z]+
  ⇓
With the risin sun to greet him
```

Use a dollar sign to match the end of the line.

Example: *Match one or more letters at the end of a line.*

```
[A-Za-z]+$
  ⇓
from a night of peaceful rest
```

(6.16) *Not just for searching*

sed
Stream editor for filtering and transforming text.
`-e's/pattern/replacement/'` Search and replace using a regular expression.

Example: *Replace* `h` *followed by any three characters, followed by* `o` *with* `goodbye`.

```
$ echo hello, world | sed -e's/h...o/goodbye/'
goodbye, world
```

(6.17) *But wait, there's more!*

These are the most important elements, common to most regex implementations. But there are loads of other features that can be used in regular expressions.

To learn more:

`https://regexone.com/`

To test and experiment with regular expressions:

`https://regex101.com/`

(6.18) *Sample final exam questions*

1. Suppose the regular expression

```
en*
```

is used to search the text

> These violent delights have violent ends

How many matches will be found for this pattern in this text?

A. 3
B. 2
C. 7
D. 0

2. Suppose the regular expression

```
...low
```

is used to search the text

> below bellow follow

Which of the words in the text will match this regular expression?

A. only `below`
B. all three of `bellow`, `follow`, and `below`
C. both `bellow` and `follow`, but not `below`
D. only `follow`

3. Suppose the regular expression

```
^[A-Za-z]+
```

is used to search the text:

Anakin: I have brought security to my new empire!
Obi-Wan: Your new empire?
Anakin: Don't make me kill you...
Obi-Wan: Anakin, my allegiance is with the Republic!
Anakin: If you're not with me then you're my enemy!
Obi-Wan: Only a Sith deals in absolutes.
Obi-Wan: I will do what I must.
Anakin: You will try.

Which of these substrings is matched by this pattern in this text?

A. try.
B. Obi-Wan
C. Republic!
D. Anakin

4. Suppose the regular expression

```
ice
```

is used to search the text

> Some say the world will end in fire,
> Some say in ice.
> From what I've tasted of desire
> I hold with those who favor fire.
> But if it had to perish twice,
> I think I know enough of hate
> To say that for destruction ice
> Is also great
> And would suffice.

How many matches will be found for this pattern in this text?

A. 4
B. 3
C. 2
D. 1

5. Which option can be used with the `grep` command to tell `grep` to search recursively in subdirectories?

A. `-i`
B. `-E`
C. `-v`
D. `-r`

6. Within a regular expression, we use _____ to form groups.

A. parentheses
B. double quotes
C. curly brackets
D. square brackets

7. Which option can be used with the `grep` command to tell `grep` to interpret special characters in patterns as regex operators?

A. `-E`
B. `-i`
C. `-v`
D. `-r`

8. Within a regular expression, we use _____ to match the end of the line.

A. `\e`
B. `$`
C. `\`
D. `^`

9. Within a regular expression, we use a _____ to choose between either of two choices.

A. backslash
B. semicolon
C. pipe
D. forward slash

10. Within a regular expression, the question mark character (`?`) indicates _____.

A. any one single character
B. zero or more of the previous
C. zero or one of the previous
D. one or more of the previous

11. Suppose the regular expression

```
(v..)+
```

is used to search the text

> These violent delights have violent ends

How many matches will be found for this pattern in this text?

A. 3

B. 7

C. 0

D. 2