Lec-03-1 Searching, globbing and Regular Expressions

Dr Syed Faisal Hasan and Dr. Hymie Latif
Computing and Information Technology
College of Enterprise and Development
Otago Polytechnic
Dunedin, New Zealand

Bachelor of Information Technology
IN616 – Operating Systems Concepts
Semester 1, 2020

Schedule

- Recap on last week
- Searching
  - Using the `find` command and globbing
- Advanced searching
  - Using the `grep` command and regular expressions

The **`find`** command

- Command syntax:

```
find <location> <options> <expression>
find /etc –maxdepth 1 –name “host”
find . –name “.*”
```

- **`find`** arguments:
  - Arguments include: location, options and expression
  - All arguments are **optional**

The `find` command

- Default behavior:
  - Defaults to **current directory** as path
  - Defaults to **search recursively**
  - Default to **search files and directories**
- **`find`** options:
  - Not many useful options
  - **`-maxdepth`**
  - **`-mindepth`**
  - Power lies in the expression

The `find` command: Expressions

- `find ~ –name “*.jpg”`
  - The `-name` searches for **filenames**
  - The * wildcard could be any number of characters
  - **So what does this command achieve?**

- `find ~ –iname “*.png”`
  - The `-iname` searches for **case insensitive filenames**
  - **So what does this command achieve?**

The `find` command: Expressions

- `find /home/user/images -not –name “*.jpg”`
  - The `-not` inverts the search
  - **So what does this command achieve?**

- `find ~/website ! –name “*.php”`
  - The `!` inverts the search
  - **So what does this command achieve?**

The `find` command: Expressions

- Multiple expressions can be used
- It helps to read longer commands in sequence

- `find ~/web -name "login*" ! -name "*.html"`
  - **So what does this command achieve?**

- `find ~/web -name "*.php" -o -name "*.html"`
  - The `-o` is an OR operator!
  - **So what does this command achieve?**

The `find` command: Expressions

- There are many other examples of expressions:

```
-executable
-size
-perm
-user
-group
-atime
-mtime
-size
```

The **find** command: Expressions

- Boolean operators:
  - Requires to be wrapped in parentheses: **( )**

    -a (-and), **-o** (-or), **!** (-not)
  - Parentheses needs to be escaped

    \(expression \)
  - Whitespace required between each element

```
find ~/web \( –name “*.php” –o –name “*.html” \)
```

  - **So what does this command achieve?**

```
find ~/web –name “*.php” –o –name “*.html”
```

The `find` command: Expressions

- `find /etc –type f`
  - The `–type f` searches for **files**
  - **So what does this command achieve?**
- `find /etc –type f –name “host*”`
  - **So what does this command achieve?**
- `find /etc –type d`
  - The `–type d` searches for **directories**
  - **So what does this command achieve?**

The `find` command: Expressions

- You can provide more than one *location*
- `find ~/BITY1 ~/BITY2 –type f –name “*.doc”`
  - **So what does this command achieve?**
- `find ~/BITY1 –type f –name “.gitignore”`
  - Remember, the `.` before a filename is hidden in Linux
  - **So what does this command achieve?**

Globbing

- We can glob find commands
- We have already looked at examples…

Basic globbing

Character	Description
`*`	Matches any character zero or more times, except for `/`.
`**`	Matches any character zero or more times, including `/`.
`?`	Matches any character one time
`[abc]`	Matches any of the specified characters (in this case, `a`, `b` or `c`)

Globbing Limitations

- Globbing is somewhat limited
  - But still powerfull
  - Gets everyday use
- What about complex scenarios? Matching:
  - Email address
  - Matching IP address
  - Matching website URL

The `grep` command

- **G**lobally search a **R**egular **E**xpression and **P**rint
  - grep likes file content
  - This does not limit it to text-based files

- Like find, grep can be used with arguments/expressions

```
grep <expression> <file/s>
grep “andrew” studentlist.txt
```

https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux

The `grep` command

THE FOLLOWING EXAMPLES ALL USE THE GPL LICENSE FILE

`/usr/share/common-licenses/GPL-3`

- `grep "GNU" GPL-3`
  - **So what does this command achieve?**
- `grep -i "License" GPL-3`
  - The `-i` performs a case insensitive search
  - **So what does this command achieve?**
- `grep -n "freedom" GPL-3`
  - The `-n` lists the line number of the file
  - **So what does this command achieve?**

Function	Globbing	Regular Expressions
Matching single character	?	.
Matching single character from alternatives	[xyz]	[xyz]
Matching character from alternatives repeatedly	[xyz][xyz] (repeating pattern)	[xyz]+ (arbitrary) [xyz]\{3\} (3 times) [xyz]\{3,\} (at least 3 times) [xyz]\{3,5\} (3 to 5 times)
Matching groups of characters repeatedly	+(xyz) – requires extended globbing	(xyz)\{2\} (2 times), see above
Character repetition		123* (matches 12, 123, 1233, 12333, etc.)
Arbitrary number of arbitrary characters	*	.*
Range	[0-9], [0-9a-z]	[0-9], [0-9a-z]
Excluding characters	[!ab]	[^ab]
Excluding ranges	[!0-6]	[^0-6]

Function	Globbing	Regular Expressions
Specific symbol at beginning	Pattern begins with desired symbol	^T
Specific symbol at end	Pattern ends with desired symbol	T$
Word boundaries	“ the “	\\bthe\\b – alternative: \\<the\\> (matches ‘the’, but not ‘them’ or ‘Athens’) – depends on Regex dialect Example includes leading backslashes to escape terms

The **grep** command: And regex

- `grep -n "^GNU" GPL-3`
  - The `^` symbol searches for the term at the start of the line
  - **So what does this command achieve?**

- `grep -n "and$" GPL-3`
  - The `$` symbol searches for the term at the end of the line
  - **So what does this command achieve?**

The **grep** command

- `grep -n "..cept" GPL-3`
  - The `.` symbol replaces a single character
  - **So what does this command achieve?**

- `grep -n "t[wo]" GPL-3`
  - The [] symbol contains a list of potential characters
  - **So what does this command achieve?**

The `grep` command

- `grep -n "[^c]ode" GPL-3`
  - This is like the expression `.ode`
  - But will not match "code" – why not?
  - What could it match?
  - Mode, mode, Code
  - [^] Matches anything except

- `grep –n "^[A-Z]" GPL-3`
  - **So what does this command achieve?**

The **grep** command and piping

- `find /etc | grep “host”` **Piping**
  Command 1 **Command 2**
- `cat file.txt | grep “the”` **Piping**
  Command 1 **Command 2**

- The pipe “|” character redirects content
- Output from one command as input to the other

The `grep`, piping and content

- These two commands do the same thing:
  - `cat file.txt | grep “kittens”`
  - `grep “kittens” file.txt`

  - **What do they do??**

The `grep`, piping and filenames

- These two commands **do not** do the same thing:
  - `ls /etc | grep “host*”`
  - `grep -r “host*” /etc`

  - **What do each do??**
  - **Why are they not the same??**

`grep` VS `find`

- `find` is primarily for file system entries
  - file names, directory names
- `grep` is for content
  - well, it can be used for anything
  - content, file names, directory names

Lab-03-1

- **TOPICS:**
  - **Find and Grep**

Regular Expressions (regex)

- Can perform complex matching strategies
  - Can match **partial or complete strings**
  - Can search **explicit repetitions**
  - Can search **alternative substrings** (groups)

- Regex is used extensively in many UNIX tools:
  - `grep`
  - `sed`
  - `awk`

- https://www.digitalocean.com/community/tutorials/an-introduction-to-regular-expressions

Regular Expressions (regex)

```
^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$

[\w-]+@([\w-]+\.)+[\w-]+

^.+@[^\.].*\.[a-z]{2,}$

^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-
  9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-
 ]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$
```

Exploring Regular Expressions (regex)

- ^[0-9][a-zA-Z]*
  - Starts with number, followed by arbitrary number of characters (case-insensitive)

- ^[^0-9][a-z].*\.conf$
  - Must not start with a number, can have arbitrary further characters/numbers and ends with '.conf'

- ([0-9]\{2\})\|([A-Z]\{2,\}).*
  - Contain either 2 numbers or 2 or more capital characters + further characters

Note the escaping of \, { and }

Exploring Regular Expressions (regex)

- 2559 6953 4962 6464
  - Extract the credit card number

- Fruits: Apple, Banana, Ananas, Pineapple
  - Extract all of the fruits

Regex Resources

- Comprehensive Overview/Cheat sheet
  - http://www.rexegg.com/regex-quickstart.html
- Interactive Regex Evaluator
  - http://www.regexr.com/
- Another evaluation tool
  - http://www.regexpal.com/

Lab-03-1 – Finish

- **TOPICS:**
- **Grep and searching the package manager**
- **Grep and regular expressions**

Class_03-1 – Homework

- **Digital ocean tutorial : find**
  - https://www.digitalocean.com/community/tutorials/how-to-use-find-and-locate-to-search-for-files-on-a-linux-vps
- **Digital ocean tutorial : grep**
  - https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux
- **Digital ocean tutorial : regular expressions**
  - https://www.digitalocean.com/community/tutorials/an-introduction-to-regular-expressions
- **Finish the lab**