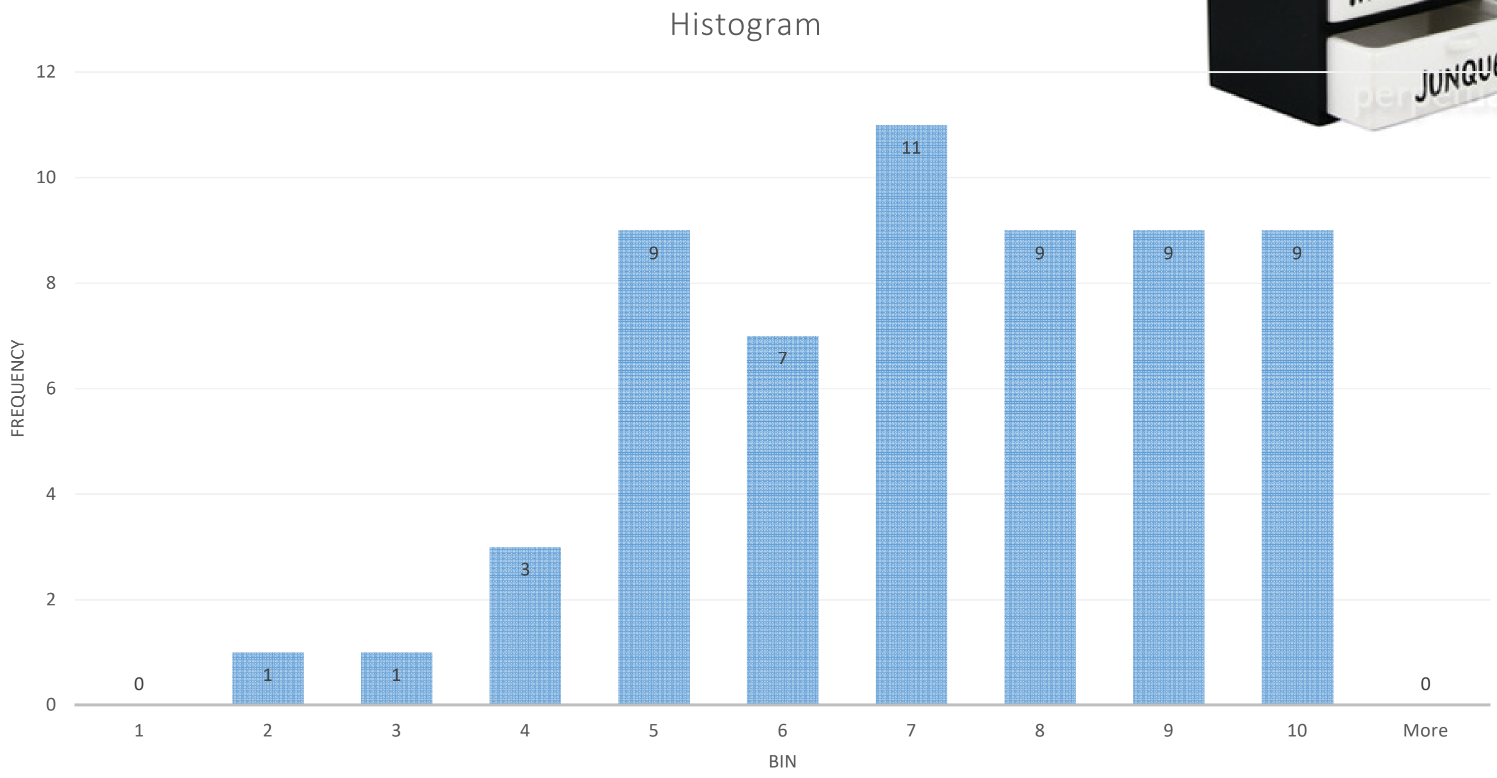
Quick Announcements

- GQ-01 results are in!



Quick Announcements

IE1

- In-class on Wednesday 9/14/2022 @ usual class time
- Bring laptop + power plug
- Taken on canvas, proctored by Honorlock
- Bring your student ID!
- Covers everything so far, including what we will discuss about module M2 before the exam
- Slides allowed + Notes
- Ubuntu VM allowed

Quick Announcements

PA1

- To help you prepare / review for IE1
- Consider it a Q&A session with hands-on exercises
- Bring laptop + power plug
- Bring any questions you might have about the study material so far

PAs / IEs / Final Exam tentative schedule

Week #	Date	MON	WED
4	9/12	PA1	IE1
8	10/10	PA2	IE2
12	11/7	PA3	IE3
15	11/28	PA4	TBD
16	12/5	Final Exam	n/a

Previously On... CIS4930

- Basics of CLI filesystem navigation
  - pwd cd mkdir rmdir rm
  - dirs popd pushd
- Basics of CLI process management
  - ^C ^Z
  - fg bg jobs
  - Signals
- Getting help
  - Manpages structures & related tools

The basic toolkit to survive CLI

Module

M02
Serious CLI

Menu for this module

T1	Globbing	Bash allows you to use so-called meta-characters to build expressions allowing you to designate sets of filenames or folder names on which you may apply all sorts of CLI tools
T2	Shell Quoting & Escaping	One of the most interesting topics when learning Bash; the syntax allowing you to control the interpretation of the above-mentioned meta-characters or even substitute the result of executing code in an expression.
T3	Bash Environment, Variables, & Options	We then examine Bash options & variables.
T4	Bash Initialization Files	Finally, we are going to look at how we may configure the Bash shell for your user accounts. We will consider individual configuration files first, then system-wide ones.

Topic

M2T1 Globbing, Glob-Patterns, Filename Substitutions

Reading Assignment:

https://ryanstutorials.net/linuxtutorial/wildcards.php

List of globbing meta-characters

Meta character = character w/ special meaning to the shell

- Filename with . at start, or . after /, or just / → matched as is
- `*` → matches anything but dot as 1st character
- `?` → matches any 1 character
- `[…]` → single character alternatives
- `[^…]` → negation of the above
- `{…,…,…}` → multi-characters alternatives
- Begins with `~` → shorthand for homedir
- `!(…)` → negate the enclosed globbing pattern

The Globbing Challenge

Use `touch` to create the following files:

file1	fileAB
file10	filea
file11	fileA
file2	fileAAA
File2	notAFile
File3	ThisOneEither5
file33	woohoo

https://linux-training.be/funhtml/ch17.html#idp54066976

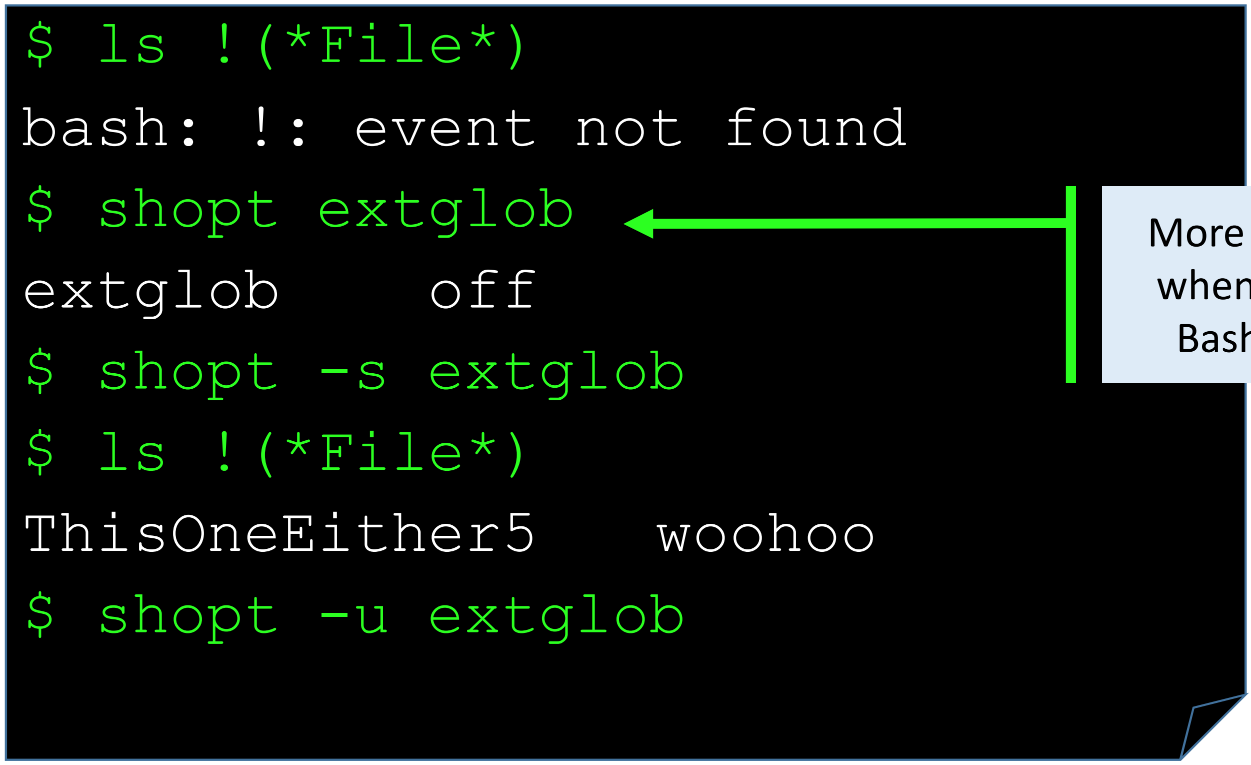
1	List (with ls) all files starting with file	
2	List all files containing File in their name	
3	List (with ls) all files starting with file and ending in a number.	
4	List (with ls) all files starting with file and ending with a lower case letter	
5	List (with ls) all files starting with File and having a digit as fifth character.	
6	List (with ls) all files starting with File and having a digit as fifth character and nothing else afterward.	
7	List (with ls) all files starting with a lower case letter & ending w/ a digit.	
8	List (with ls) all files that have exactly five characters.	
9	List (with ls) all files that start with f or F and end with 3 or A.	
10	List (with ls) all files that start with f have i or R as second character and end in a digit.	
11	List all files that do not start with the letter F.	
12	List all files that do not have File in their name	

https://linux-training.be/funhtml/ch17.html#idp54066976

1	List (with ls) all files starting with file	ls file*
2	List all files containing File in their name	ls *File*
3	List (with ls) all files starting with file and ending in a number.	ls file*[0-9]
4	List (with ls) all files starting with file and ending with a lower case letter	ls file*[a-z]
5	List (with ls) all files starting with File and having a digit as fifth character.	ls File[0-9]*
6	List (with ls) all files starting with File and having a digit as fifth character and nothing else afterward.	ls File[0-9]
7	List (with ls) all files starting with a lower case letter & ending w/ a digit.	ls [a-z]*[0-9]
8	List (with ls) all files that have exactly five characters.	ls ?????
9	List (with ls) all files that start with f or F and end with 3 or A.	ls [fF]*[3A]
10	List (with ls) all files that start with f have i or R as second character and end in a digit.	ls f[iR]*[0-9]
11	List all files that do not start with the letter F.	ls [^F]*
12	List all files that do not have File in their name	ls !(*File*)

https://linux-training.be/funhtml/ch17.html#idp54066976

Wait! The last one is not working!



```
$ ls !(*File*)
bash: !: event not found
$ shopt extglob
extglob         off
$ shopt -s extglob
$ ls !(*File*)
ThisOneEither5    woohoo
$ shopt -u extglob
```

More about this when we cover Bash options

Want to read more about extended Globbing Patterns?

LINUX JOURNAL

https://www.linuxjournal.com/content/bash-extended-globbing

Examples of Extended Globbing

`?(pattern-list)`	Matches zero or one occurrence of the given patterns
`*(pattern-list)`	Matches zero or more occurrences of the given patterns
`+(pattern-list)`	Matches one or more occurrences of the given patterns
`@(pattern-list)`	Matches one of the given patterns
`!(pattern-list)`	Matches anything except one of the given patterns

Let’s try some of these!

- List all the JPEG and GIF files that start with either "ab" or "def”:

[redacted]

- How would we do that without extglob?

[redacted]

- List all the .jpg files that start with ab followed by one or more occurrences of the digit 2 or one or more occurrences of the digit 3

[redacted]

- How would we do that without extglob?

[redacted]

Let’s try some of these!

- List all the JPEG and GIF files that start with either "ab" or "def”:

```
ls +(ab|def)*+(.jpg|.gif)
```

Actually, the above is more accurate. e.g., ababab.jpg @(ab|def) would be more in lines with the globbing

- How would we do that without extglob?

```
ls ab*.jpg ab*.gif def*.jpg def*.gif
```

- List all the .jpg files that start with ab followed by one or more occurrences of the digit 2 or one or more occurrences of the digit 3

```
ls ab+(2|3).jpg
```

- How would we do that without extglob?

Nope :)

* Globbing is GREEDY

- list all the files that aren't JPEGs or GIFs

[redacted]

* Globbing is GREEDY

- list all the files that aren't JPEGs or GIFs

```
ls *!(.jpg|.gif)
```

- Doesn't work because the ".jpg" and the ".gif" of any file's name end up getting matched by the "*" and the *null string* at the end of the file name is the part that ends up *not* matching the "!(...)" pattern.

* Globbing is GREEDY

- list all the files that aren't JPEGs or GIFs

```
ls *!(.jpg|.gif)
```

- Doesn't work because the ".jpg" and the ".gif" of any file's name end up getting matched by the "*" and the *null string* at the end of the file name is the part that ends up *not* matching the "!(...)" pattern.

```
ls !(*.jpg|*.gif)
```

Topic

M2T2
Shell Quoting & Escaping

Reading Assignment:

https://ryanstutorials.net/linuxtutorial/wildcards.php

Bash Meta-Characters and Backslash Escaping

Trivial meta-character: SPACE → separates things in the CLI

- `touch filewith onespaceinitsname`
- `ls -l`

- `touch filewith\ onespaceinitsname`
- `ls -l`

It may mess w/ AUTOCOMPLETION

- `ls filewith [TAB]` → the space messes up the auto-completion
- `ls filewith\ [TAB]` → this works much better

…Works but tedious if we have many spaces…

(we'll see better later)

https://youtu.be/c457F9p7Gsw

Another silly example: \n meta-char

- echo hello world [ENTER]
- echo hello world \[ENTER]

- Useless?
- Useful for multi-lines typing (convenience)

Escaping the \

  - echo this is just a \\ in the command line

Escaping globbing meta-chars

- Setup
  - `touch COP2512 COP2513 COP4610 COP4931`
- Creating weird file or touch-ing the above folders?
  - `touch COP*`
  - `touch COP\*` → if I want a file with that weird name
- Same for removing
  - `rm COP*` → the COP* is erased, folders are safe but tried
  - `rm COP\*` → better; only file affect is COP*

Weird Case

What would happen

```
touch COP*something
```

- We expect that expansion / substitution would lead no results
    - Does that mean error?
    - Or we touch COPsomething?
    - Or we touch `[nothing at all]`

Weird Case

```
touch COP*something
```

- We expect that expansion / substation would lead no results

→ Because we have no results for the filename substitution
we keep the string COP*something **as is**

Why?

- By default, Bash expands a glob-pattern that matches nothing into itself

How to change this bash behavior?

- shopt -s nullglob

Other (related) bash options of interest

dotglob

- *If set, Bash includes filenames beginning with a '.' in the results of filename expansion.*
- *The filenames '.' and '..' must always be matched explicitly, even if* dotglob *is set.*

failglob

- *If set, patterns which fail to match filenames during filename expansion result in an expansion error.*

nocaseglob

- *If set, Bash matches filenames in a case-insensitive fashion when performing filename expansion.*

nullglob

- *If set, Bash allows filename patterns which match no files to expand to a null string, rather than themselves.*

https://www.gnu.org/software/bash/manual/html_node/The-Shopt-Builtin.html