M1 – More Linux Shell

CS 136L F23 – LEC 3

Yiqing Irene Huang

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

Main Points

Learn more about the features, commands, utilities and tools within Linux Shell

1. Globbing
    - Filters file names
2. Pipes
    - Connects stdout of program 1 to stdin of program 2
3. Embedding Commands
    - Makes program 1 output an argument of program 2
4. Regular expression with egrep
    - Searches through text
5. File permissions

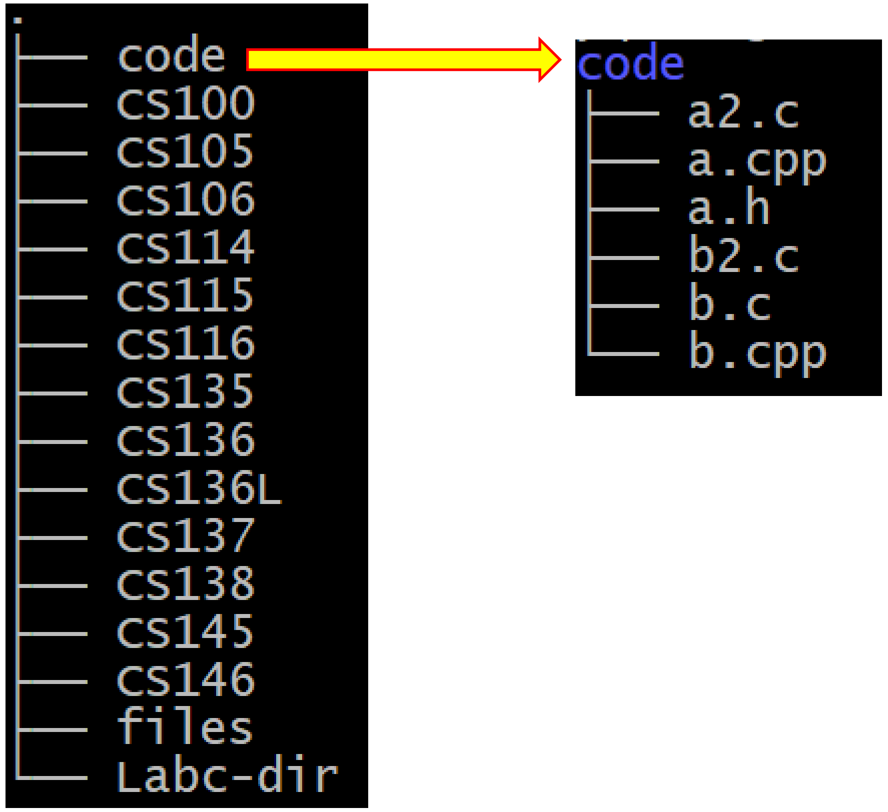
Globbing

- Globbing is also known as wildcard expansion
  - The process of matching expressions containing wildcards to filenames
  - It is a feature of shell itself (i.e. not of individual commands)
  - It is different from regular expressions
- It is the shell that
  - interprets the globbing pattern,
  - applies the pattern on the files in the current directory, and
  - substitutes the globbing pattern with filenames that match the pattern

Globbing

- How to filter files based on their names?
  - list all .c files
  - list all .c or .h files
  - list all files that are not .c or .h

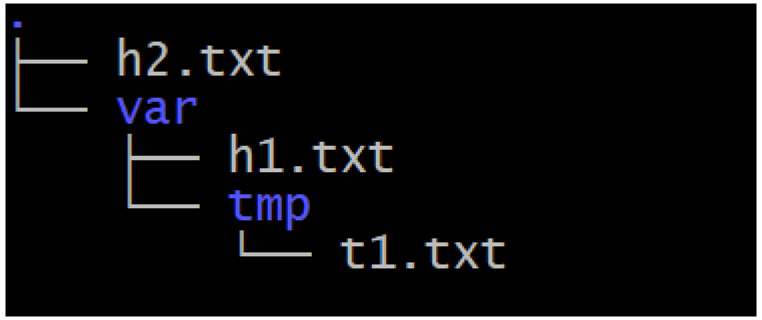
Symbol	
?	CS13?
*	CS13*
[]	CS13[5-7], *.[ch]
[!]	CS13[!6]
{}	*.{c,h}, *.{cpp,h}



```
.
├── code
├── CS100
├── CS105
├── CS106
├── CS114
├── CS115
├── CS116
├── CS135
├── CS136
├── CS136L
├── CS137
├── CS138
├── CS145
├── CS146
├── files
└── Labc-dir
```

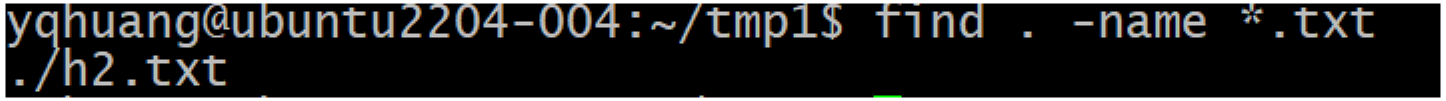
```
code
├── a2.c
├── a.cpp
├── a.h
├── b2.c
├── b.c
└── b.cpp
```

Globbing – Quotes

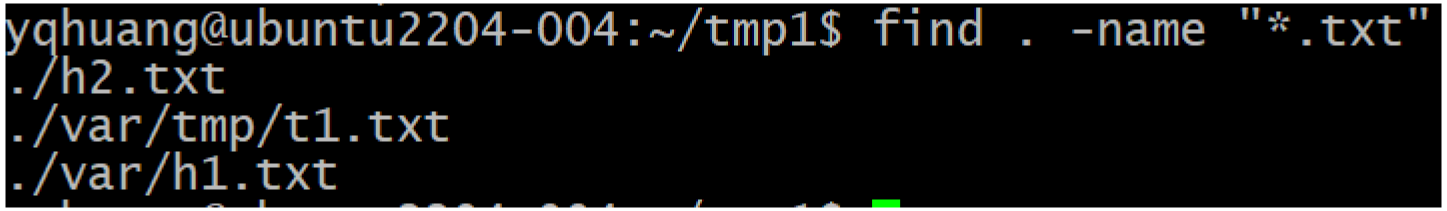


```
.
├── h2.txt
└── var
    ├── h1.txt
    └── tmp
        └── t1.txt
```

- Suppress shell's globbing pattern ability
  - Single or double quotes around the pattern string
- Use together with the "find" or "grep" command
  - You want to suppress the pattern and let "find" expand the pattern
- Examples
  - `find . –name *.txt`
  - `find . –name "*.txt"`



```
yqhuang@ubuntu2204-004:~/tmp1$ find . -name *.txt
./h2.txt
```



```
yqhuang@ubuntu2204-004:~/tmp1$ find . -name "*.txt"
./h2.txt
./var/tmp/t1.txt
./var/h1.txt
```

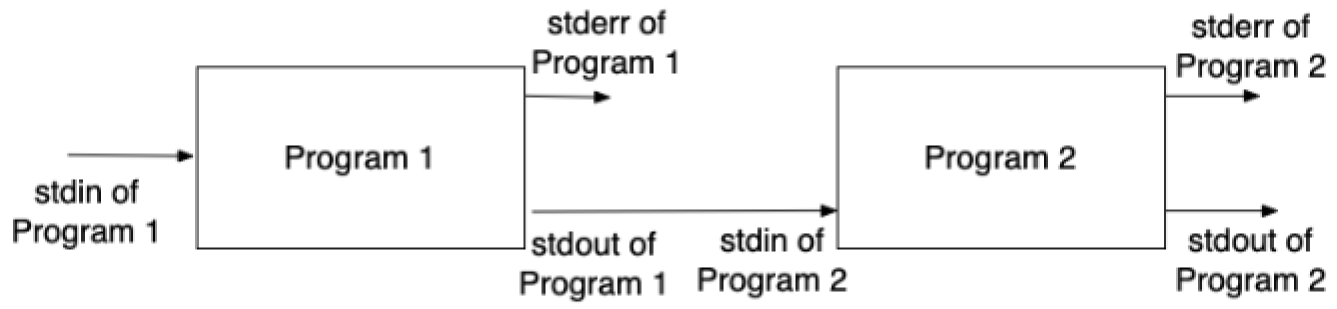
Standard Input vs Input Arguments

- Standard input is what you feed to the running program after the program starts to run. Usually it is through keyboard, but it can be re-directed by using a file.
- Input arguments is what you feed to the program before the program executes.

Commands	Stdin	Notes	Cmd Arg	
ls	N/A		-a, <dir name>	ls –a, ls dir1
echo	N/A		<string>	echo “hi”
cat	Yes	Print stdin to stdout	<file name>	cat file.txt
wc	Yes	Until you hit Ctrl-D	<file name>	wc file.txt

Pipes

- Redirect the output stream from Program 1 to the input stream of Program 2



- Construct command pipelines with two or more stages

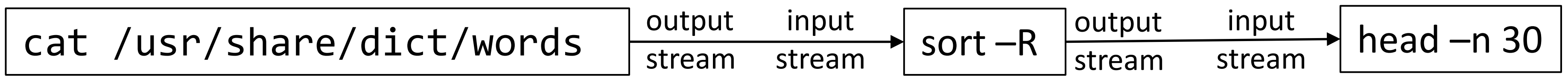
```
ls *.c > output.txt
wc -l < output.txt
```

```
ls *.c | wc -l
```

The Command Pipeline

Output 30 random words to the screen without using "shuf"

- Combine the commands and put them into a pipeline
  1. Output the dictionary /usr/share/dict/words
  2. Sort the words randomly (man sort)
  3. Get the first 30 words (man head)



```
cat /usr/share/dict/words | sort -R | head -n 30
```

Embedding Commands

- Embed the output of a command in a string
- Use the output string as an argument to another command
- Example: show all processes from the user
  - ps –ef | grep $(whoami)
  - ps –U `whoami`
- Embedding command is also known as command substitution
  - $() , POSIX compatible, modern, preferred
  - `` , backticks, deprecated, a lot less typing, so still used
  - Not good at nesting http://mywiki.wooledge.org/BashFAQ/082

Standard Input vs Argument

- Suppose the file path.txt contains a single line:

```
/u/cs136l/pub/lab0/start.txt
```

  How can we print the contents of start.txt to the screen?

- Most commands take both standard input and input arguments
  - `cat path.txt` prints the contents of the file path.txt. Here path.txt is an argument to our program and the output is a stream (i.e. stdout)
  - `cat < path.txt` redirect the input from the path.txt. So it is as if we call cat without any argument and then type in the contents in the path.txt
  - `cat $(cat path.txt)` will feed the contents of the file path.txt as the argument to the outer cat command. It is equivalent to

```
cat /u/cs136l/pub/lab0/start.txt
```

Regular Expression with egrep

- Regular Expression (RE) is a pattern that describes a set of strings
  - BRE: Basic RE
  - ERE: Extended RE
  - PCRE: Perl-compatible RE
- The grep utility

Globbing vs Regular Expression

Metacharacters	Globbing Matches	Example	Regex Mathes	Example
.	.		any one char	a..d
*	0 or more chars	*k	zero or more the thing before it	[0-9]*
?	any one char		zero or one the thing before it	A ?[0-]*
+	+		1 or more the thing before it	[a-zA-Z]+
|	N/A		or	A|B
\	escape thing after	*	escape thing after it	\?, \.
[*set*]	any char in *set*	[abc]	any char in *set*	[a-zA-Z_0-9]
[!*set*]	any char not in *set*	[!a-zA-Z0-9_]	N/A, ! is not a metacharacter	[!a-z]
[^*set*]	same as [!*set*] in bash		any char not in *set*	[^!@#]
{}	or	*.{cpp,hpp}	{N}, N times the thing before it	[0-9]{4}
^	N/A		Begin with the thing after it	^[0-9]+
$	N/A		End with the thing before it	[0-9]$

Exercise - regex

- Which of the following are recognized by `a...e`
  a) apse
  b) apple
  c) apply
  d) applesauce
  e) red delicious apple

File Permissions

- Use `ls -l` to print permissions
- A file can have read(r), write(w) or execute(x) permissions.
- Permission groups:
  - user(u), owner of the file
  - group(g), a group of users who have access to the file
  - other(o), users other than u and g
  - all(a), everyone
- Permissions are shown as rwx bits

```
-rwxr-xr-x
```

Change the Permission

- Use `chmod` command (ownership + operator + permission)

Ownership	
u	owner/user
g	group
o	other
a	all

Operator	
+	add permission
-	revoke permission
=	set permission exactly

Permission	
r	read
w	write
x	execute

```
chmod u+x myprogram
chmod g+rx myprogram
chmod o-x myprogram
chmod a=rx myprogram
```

```
chmod 700 myprogram
chmod 750 myprogram
chmod 711 myprogram
chmod 555 myprogram
```

Dec	Bin	Dec	Bin
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Warnings About Permissions

Source: https://memegenerator.net/instance/68424931

Acknowledgement

- Slides by courtesy of Carmen Bruni and Anton Mosunov
- Demo notes from Nomair Naeem
- Demo lectures by Carmen Bruni and Nomair Naeem

References

- CS 136L edX notes at https://online.cs.uwaterloo.ca/