

CSE 374 Lecture 3

I/O Redirection and Scripts



Hw0

- Handed in last night
- Goal: ensure that you are able to use the fundamental tools we need for this course
 - ◆ If you had issues, follow up to correct them ASAP.
- Debugging: in this course you need to work more independently. Use all the resources you have to find answers.
 - ◆ Ex: scp
- If you added the course late: You will have two days from when you get your klaatu account to complete the exercise before it uses a 'late day'. I'll grant an amnesty on the Canvas discussion until Sunday, 11:59 pm

I/O Streams

- All bash commands have three streams
 - 0- stdin [keyboard]
 - 1- stdout [screen]
 - 2- stderr [screen]
- Can redirect streams
 - < yourInput
 - > yourOutput
 - >> appendYourOutput
 - 2> yourError
 - &> yourOutput&Error
 - And more...
- Special File /dev/null
 - Is EOF if input
 - Data is discarded if output
- Can combine one cmd to the next
 - Cmd1 | cmd2 - pipe output of cmd1 into input of cmd2
 - Cmd1; cmd2 - do one after another
 - Cmd1 `cmd2` - use output of cmd2 as input to cmd1
- Can use cmd logic
 - Cmd1 || cmd2 - do cmd2 if cmd1 fails
 - Cmd1 && cmd2 - do cmd 2 if cmd1 succeeds

Special Characters

! > < & | * ~ [] “ ‘ ` \$ /

\ is escape
character

“string”

‘string’



What do they all
mean?

Would substitute
things like \$VAR

Suppresses
substitutions

Shell Behavior

All redirection & string expansion or substitutions are done by the shell, before the command.

Command only sees resulting I/O streams.

Bash Language

- Bash acts as a language interpreter
 - Commands are subroutines with arguments
 - Bash interprets the arguments & calls subroutine
 - Bash also has its own variables and logic

Towards Scripts

- Shell has a state (working directory, user, aliases, history, streams)
- Can expand state with variables
- ‘Source’ runs a file and changes state
- Can run a file without changing state by running script in new shell.

Okay, lets make a script!

1. First line of file is `#!/bin/bash` (avoids problem of 'source' by running)
2. Make file executable (`chmod u+x`)
3. Run a file `./myNewScript`
4. Shell sees the shell program (`/bin/bash`) and launches it to run the script
5. Can include
 - a. String tests (string returns true if non-zero length, `string < string`, etc.)
 - b. Logic (`&&`, `||`, `!`) - use double brackets
 - c. File tests (`-d` : is directory, `-f`: is file, `-w`: file has write permission etc.)
 - d. Math - use double parens

Script Arguments & Errors

Script refers to i^{th} argument at $\$i$; $\$0$ is the program

Use 'shift' to move arguments towards left ($\$i$ become $\$i-n$)

Exit your shell with 0 (normal) or 1 (error)

Variables & Alias

Define variable

```
i=15
```

Access variable

```
$i
```

Undefined variable is empty string

Alias cheer="echo yahoo\!"