I/O Redirection

Module 13

Overview

- In this module:
- Redirecting I/O to and from Files
- Using Device Files

I/O Model

- Every process has three I/O 'channels'
- standard input, or stdin
  - or fd (file descriptor) 0
- standard output, or stdout
  - or fd 1
- standard error, or stderr
  - or fd 2

I/O Defaults

- Normally:
- stdin – keyboard
- stdout – screen
- stderr – screen
- These can be redirected to files or other programs

Redirecting I/O

- To direct output to a file:
- command > file
- To direct errors to a file:
- command 2> file
- To direct input from a file
- command < file

Examples

- Examples
- who > userlist
- ls -l > files
- Multiple operators can be used in the same line
- cat < input-file > output-file
- ls -l /root /usr 2> errors

Appending Output

- The > operator opens the file as new
- It truncates any existing file data to zero
  - In other words, **overwrites** the file
- To append output, use the >> operator
- $ prog >> existing_file
  - output is appended to existing_file

Redirecting stdout and stderr

- Suppose you want to redirect stdout & stderr to the same place
- What happens here?
- $ prog > output_file 2> output_file
- Question:
- Who opens the output_file?

Redirecting stdout and stderr

- To redirect stdout and stderr to the same place
- $ prog > output_file 2>&1
  - This says “send channel 2 to wherever channel 1 is going”
  - It copies channel 1 to channel 2
- Bash 4 has a shortened form
- $ prog &> output_file

Appending stderr

- Standard error can also be appended
- $ prog 2>> error_log
  - This same syntax works for any other file descriptor

Using Device Files

- In Linux, a device **is** a file
- /dev/null
  - Black hole: sent data gets junked; reading data returns EOF
- /dev/tty
  - The current terminal
- /dev/zero
  - An infinite source of zeroes