Command Line Scripting

Introduction

- BS, MS, Computer Science
- Worked jointly with University of Tennessee and Oak Ridge National Lab
- Three (3) years at University of Tennessee
- Seven (7) years at Iowa State University
- Unix/Linux administrator since 2004

Goals

- Input and Output Streams
- File Descriptors
- Input Redirection
- Output Redirection
- Pipes
- Pipe Chaining

Input/Output

- Input Streams
- Output Streams
- Error Streams
- File Descriptors

Shell

A command-line ‘interpreter’ that provides a method of running programs on a computer

Streams

- Standard Input:
  - Abbreviated **STDIN**
- Standard Output:
  - Abbreviated **STDOUT**
- Standard Error:
  - Abbreviated **STDERR**

Stream Examples

- Input Streams:
  - Typing something into the terminal
  - *Redirecting* input from another file
- Output Streams:
  - Output from a program
- Error Streams:
  - Output from program errors...

File Descriptors

- Input (0)
- Output (1)
- Error (2)

Why is this important?

Redirection

- Use input from somewhere else
- Put output somewhere else
- Put any errors somewhere else

Input Redirection

- Use input from somewhere else

Uses the “<” operator

Example:

$> sort < unsortedfile

Output Redirection

- Put output somewhere else

Uses the “>” operator

Example:

$> sort unsortedfile > sortedfile

Error Redirection

- Put output somewhere else

Uses the “>” operator and specifies the “error” file descriptor (2).

Example:

$> cat ./non-existant-file 2> errors

Multiple Redirection

- Use input, output, and/or error to/from somewhere else

Example:

$> sort < unsortedfile > sortedfile 2> errorfile

Multiple Redirection

- Use input, output, and/or error to/from somewhere else

Example:

$> sort < unsortedfile > sortedfile 2> errorfile

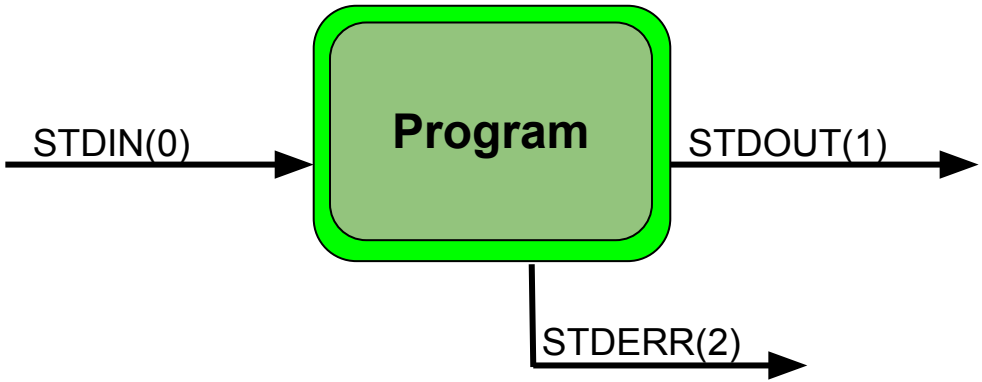
Multiple Redirection

Example:

sort	Program to run
< unsortedfile	Inputfile to read
> sortedfile	Output to write
2> errorfile	Errors to write

Examples

- **Input:**
  - read
- **Output:**
  - ls
- **Error:**
  - cat file-that-doesnt-exist



File Descriptor Visualization

Pipes

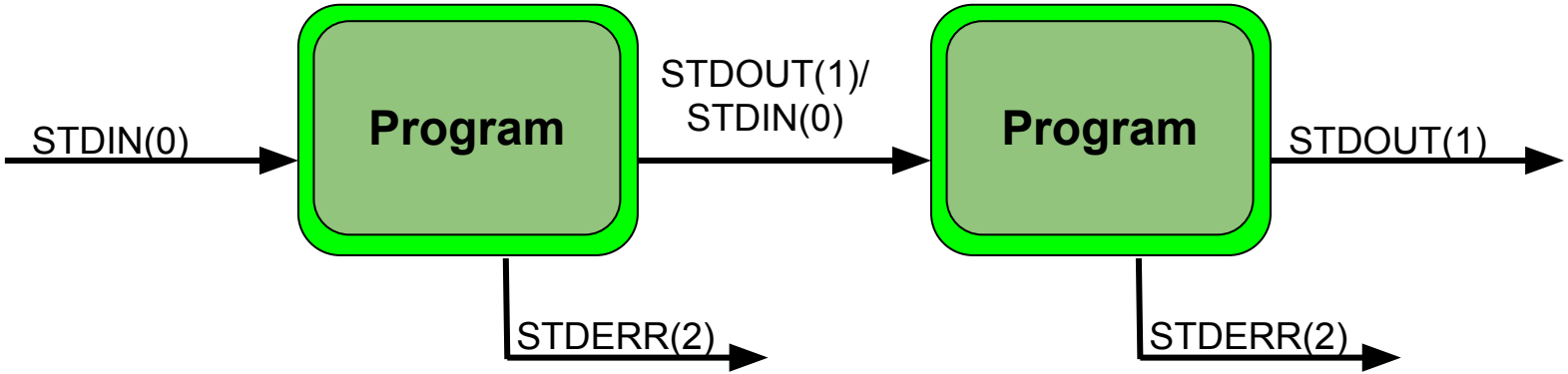
- Can connect multiple commands together.
- Output of one command becomes the input of another command

Pipes

$> sort some-file | uniq

$> cat username password | grep -v “root” | sort

$> head -n 20 numberfile | tail -n 10



Pipe Visualization

Commands

Common Filters

- sort
- uniq
- grep
- cat / tac
- head / tail
- tr

Advanced Filters

- grep
- sed
- awk

sort

- Sorts lines of text files
- Reads a file, or reads from **STDIN**
- Can do:
  - Dictionary order (-d)
  - Numeric order (-n)
  - “General” numeric order (-g)
- Caution, numeric order and “generic” numeric order are VERY different

uniq

- Reports (or omits) repeated lines
- Can also count repeated lines
- Can read from a file, or from **STDIN**
- Writes to **STDOUT**

grep

- Pattern searching
- Usage:
  - *grep <string> [filename]*
- Reads from files or **STDIN**
- Will output matches to **STDOUT**

cat / tac

- Concatenate files and print to **STDOUT**.
- tac concatenates files and prints them in reverse
- If no filename is given, reads input from **STDIN**

head / tail

- Outputs the first (head) or last (tail) part of files
- By default, prints lines to **STDOUT**
- Can specify the number of lines to print

tr

- “Transposes” a character into another character.
- Reads from a file or from **STDIN**.

grep / sed / awk

- More advanced, but we will discuss these in a future seminar.

Examples