Greybeard's Guide to *NIX

or, how I learned to love the command line

Steve Roggenkamp

Institute for Biomedical Informatics
University of Kentucky
steve.roggenkamp@uky.edu
https://github.com/roggenkamps/GreybeardsGuide

13 Sep 2017

Why???

The command line is Ancient and Decrepit? Why waste time on it?

- Systems administration
- Resource constrained systems
  - Embedded systems
  - High performance systems
- It can do a lot for you

Why???

The command line is Ancient and Decrepit? Why waste time on it?

- Systems administration
- Resource constrained systems
  - Embedded systems
  - High performance systems
- It can do a lot for you

Why???

The command line is Ancient and Decrepit? Why waste time on it?

- Systems administration
- Resource constrained systems
  - Embedded systems
  - High performance systems
- It can do a lot for you

Why???

The command line is Ancient and Decrepit? Why waste time on it?

- Systems administration
- Resource constrained systems
  - Embedded systems
  - High performance systems
- It can do a lot for you

Why???

The command line is Ancient and Decrepit? Why waste time on it?

- Systems administration
- Resource constrained systems
  - Embedded systems
  - High performance systems
- It can do a lot for you

The Shell, a must have

- Parses your input and executes commands, think REPL
- Provides an environment in which you work
- Provides a scripting programming language to automate tasks

The Shell, a must have

- Parses your input and executes commands, think REPL
- Provides an environment in which you work
- Provides a scripting programming language to automate tasks

The Shell, a must have

- Parses your input and executes commands, think REPL
- Provides an environment in which you work
- Provides a scripting programming language to automate tasks

Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

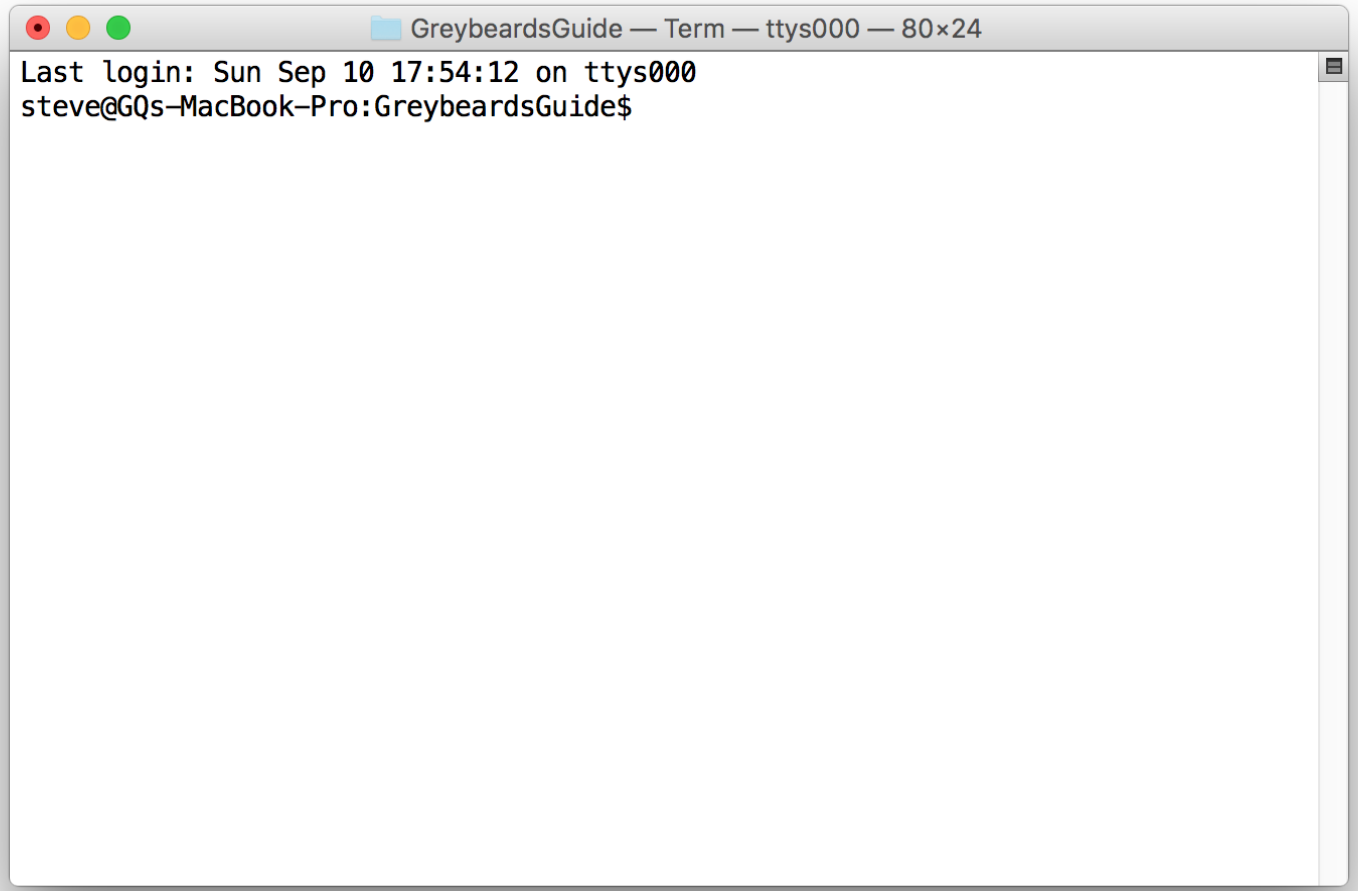
Shells - If you don't like one, try another

- `ash`, `bash`, `dash`
- Bourne shell, Korn shell, PD Korn, Bourne-again shell
- `csh`, `tcsh`, `zsh`
- Busybox (which normally uses `ash`)
- Each provides unique set of capabilities
- Today's talk focuses on Bourne-again shell, aka, `bash`
  - Default shell for most Linux and Mac OSX
  - Available for most *NIX system

Starting a shell

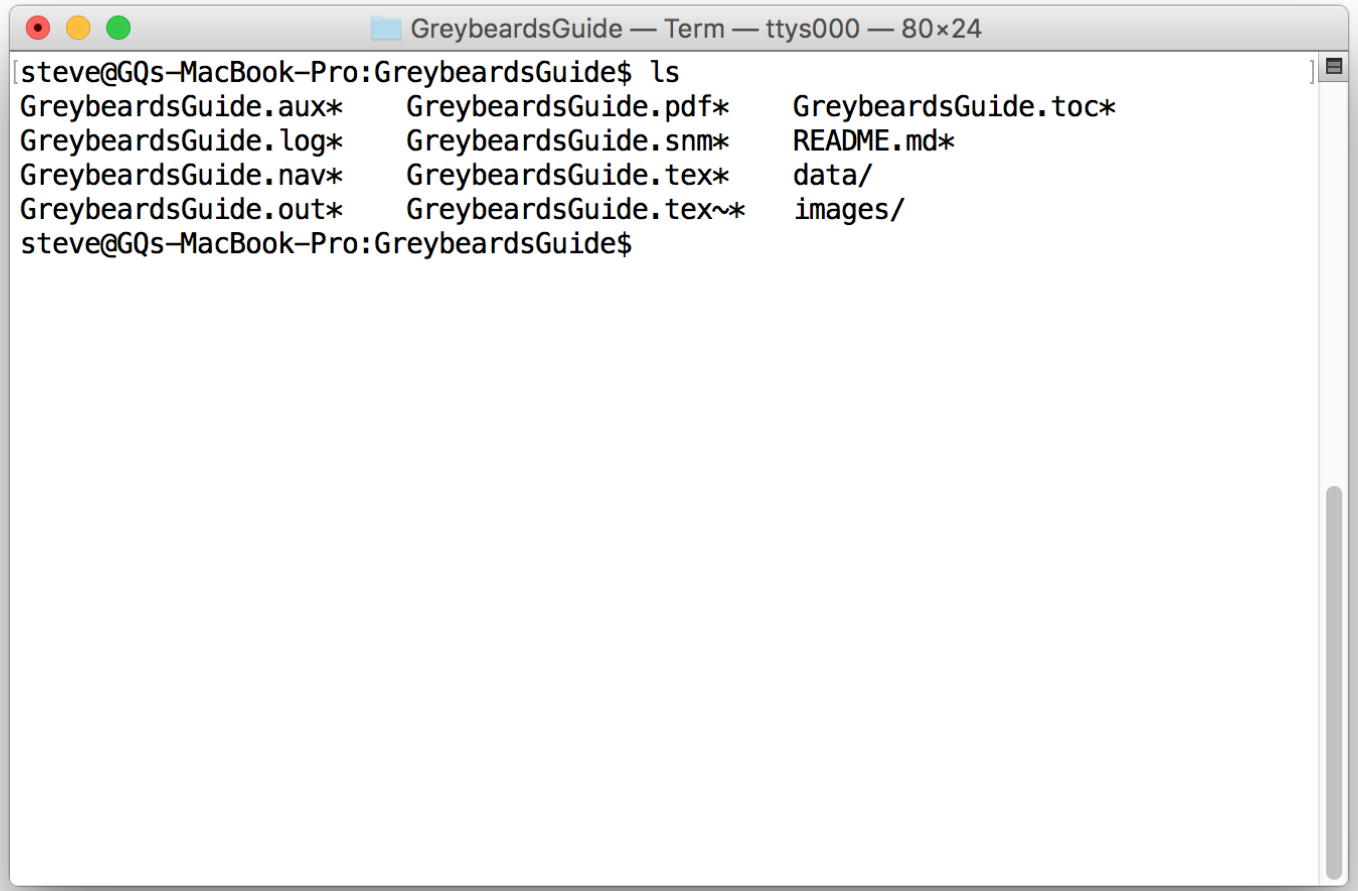
- Start a terminal in a GUI environment
- Remotely execute a shell via `ssh`
- Log into a system console

Sample session start



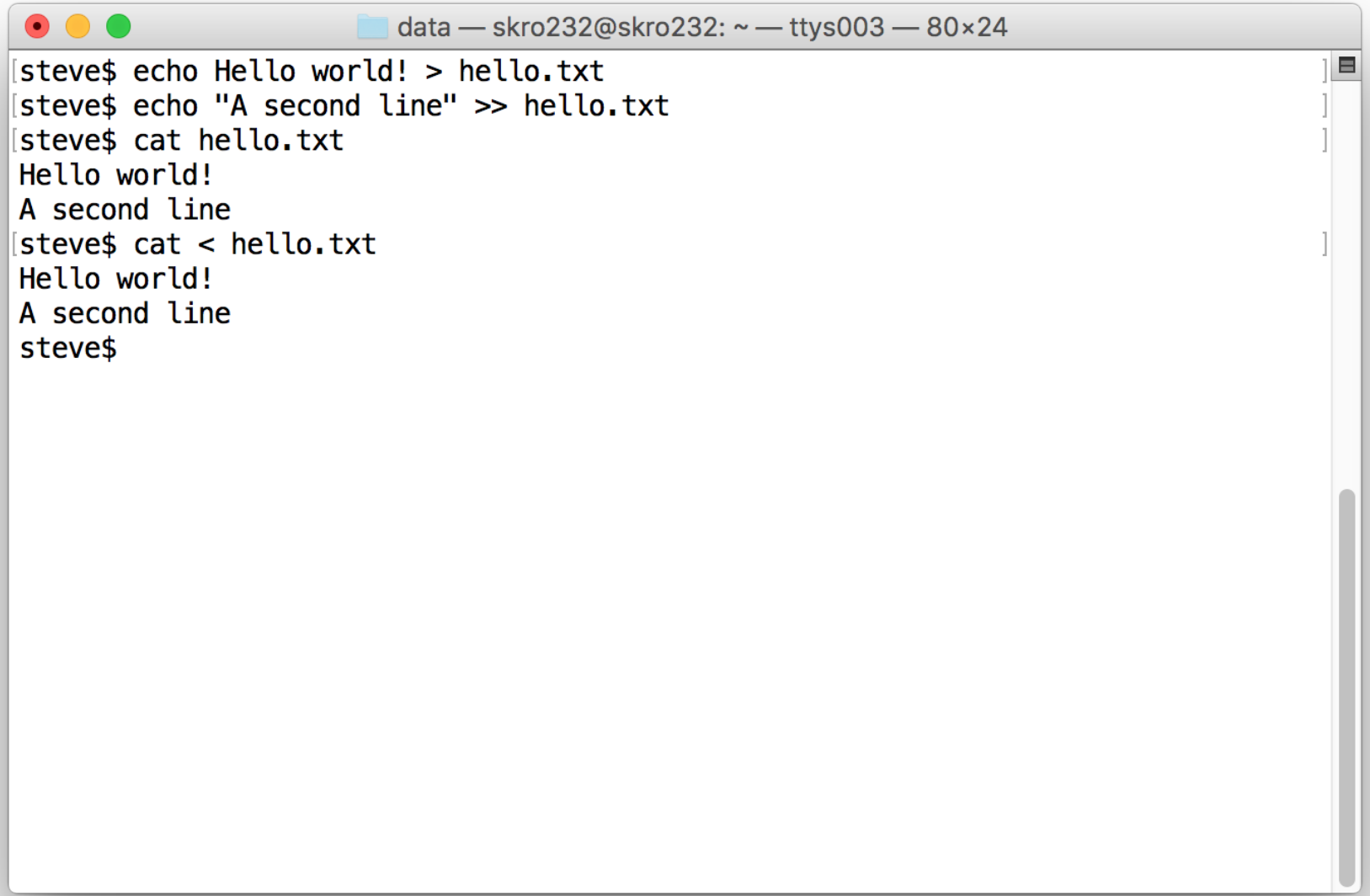
Now what?? Coder's Block!!!

Session - list current directory

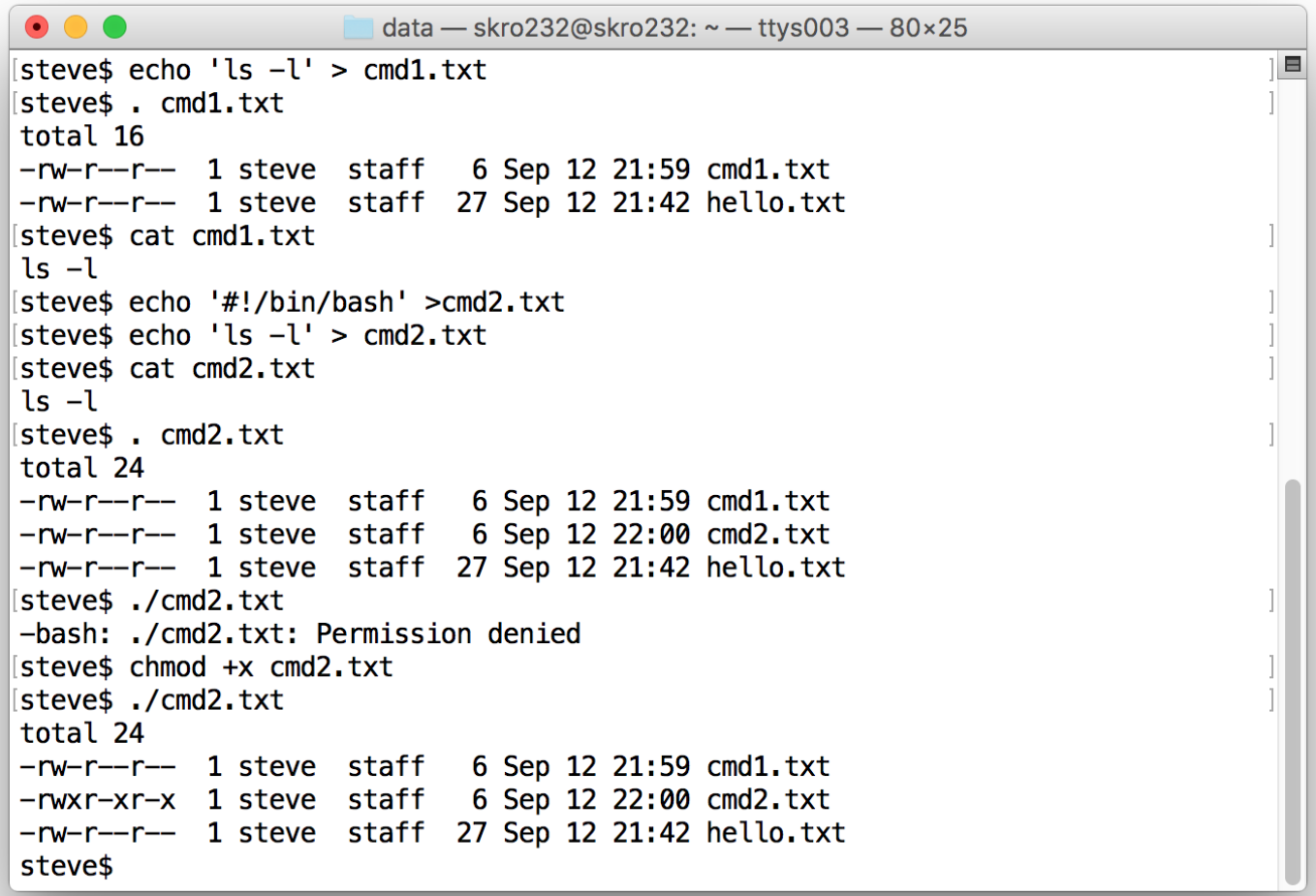


`ls` lists the contents of a directory

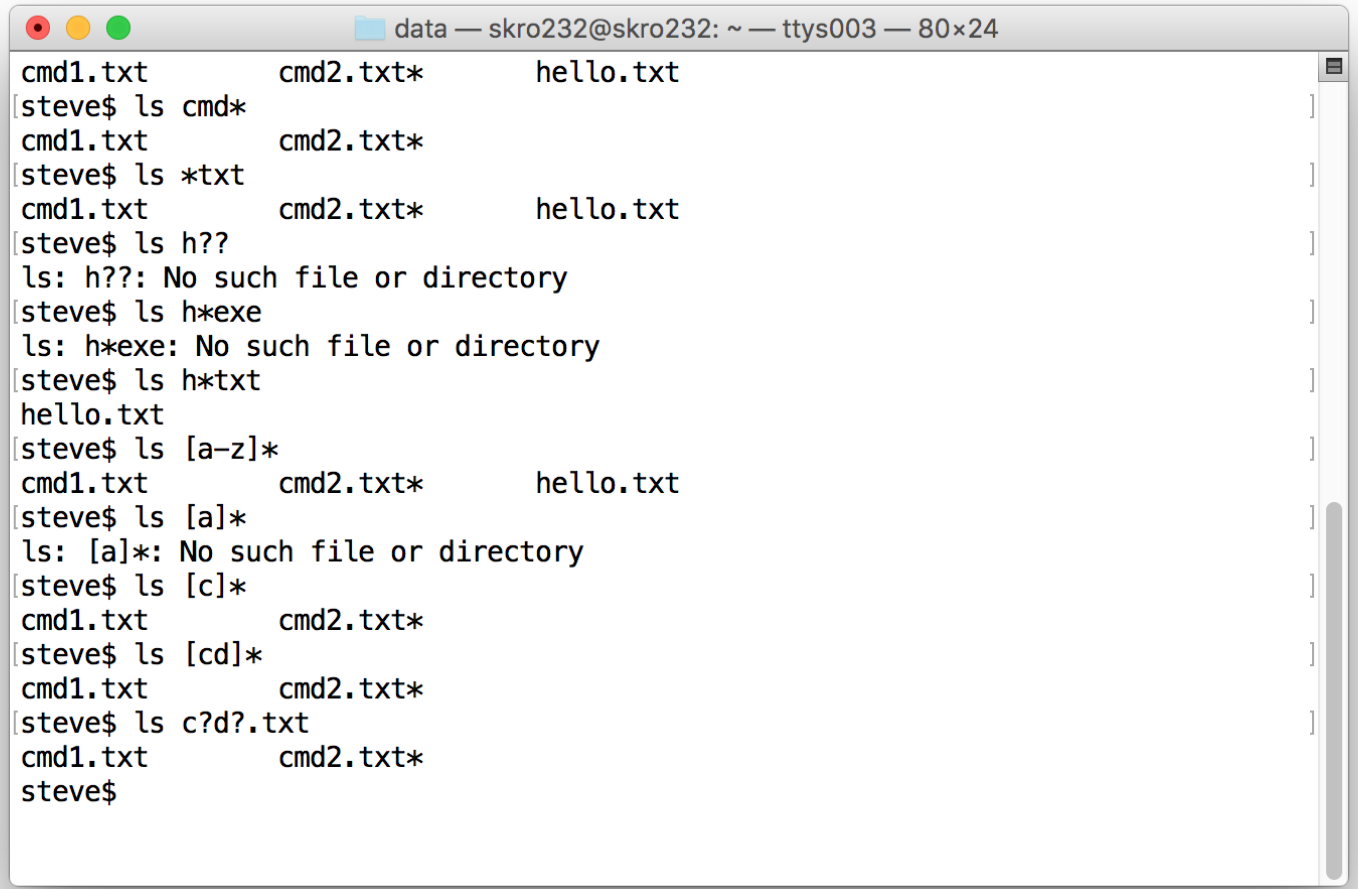
Shell - file redirection



Shell - executing scripts



Shell - filename patterns

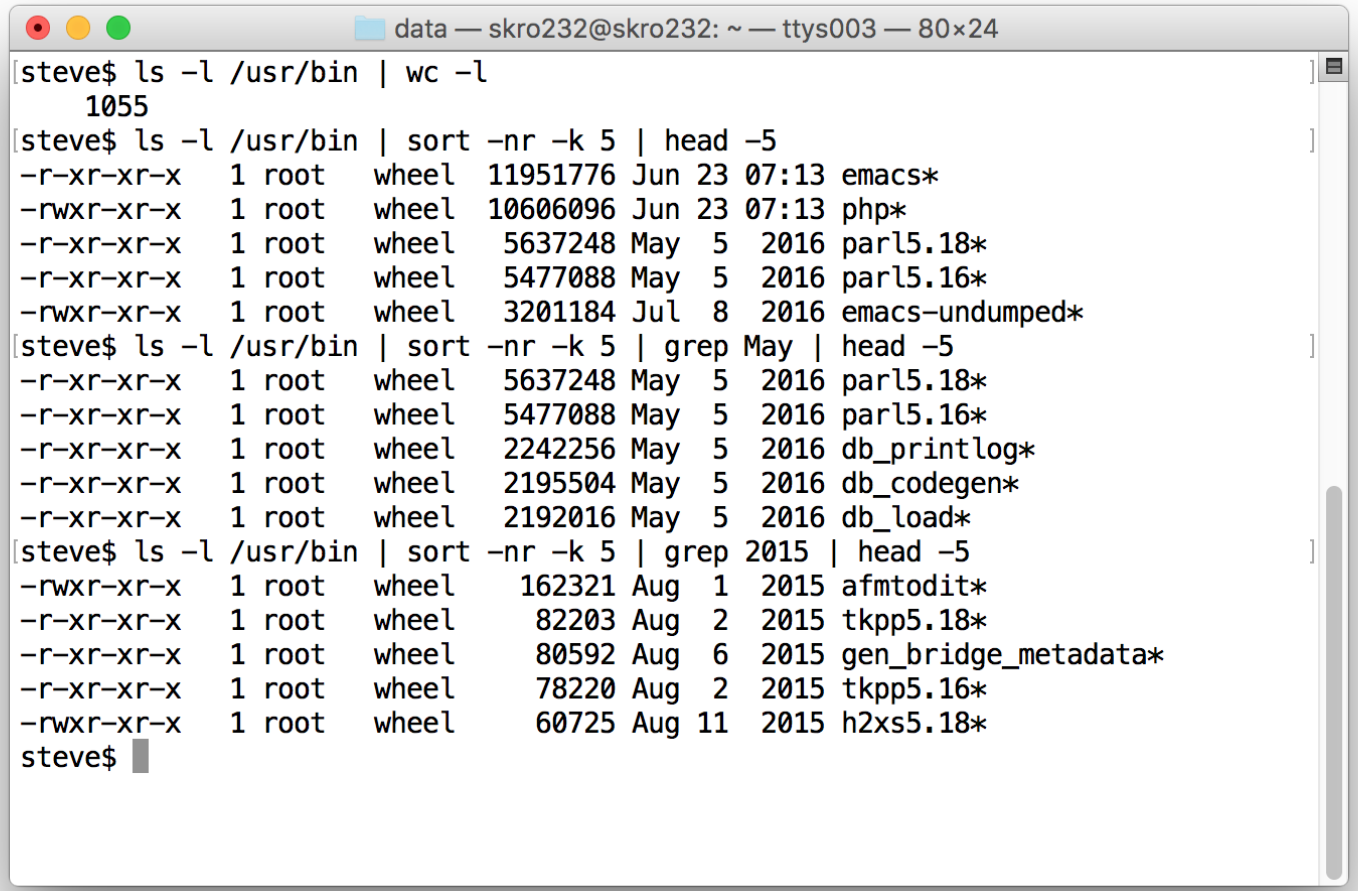


- 'file.out' r
- *NIX doe
- * matche
- ? matche
- [xy]match
- [x-z]matc

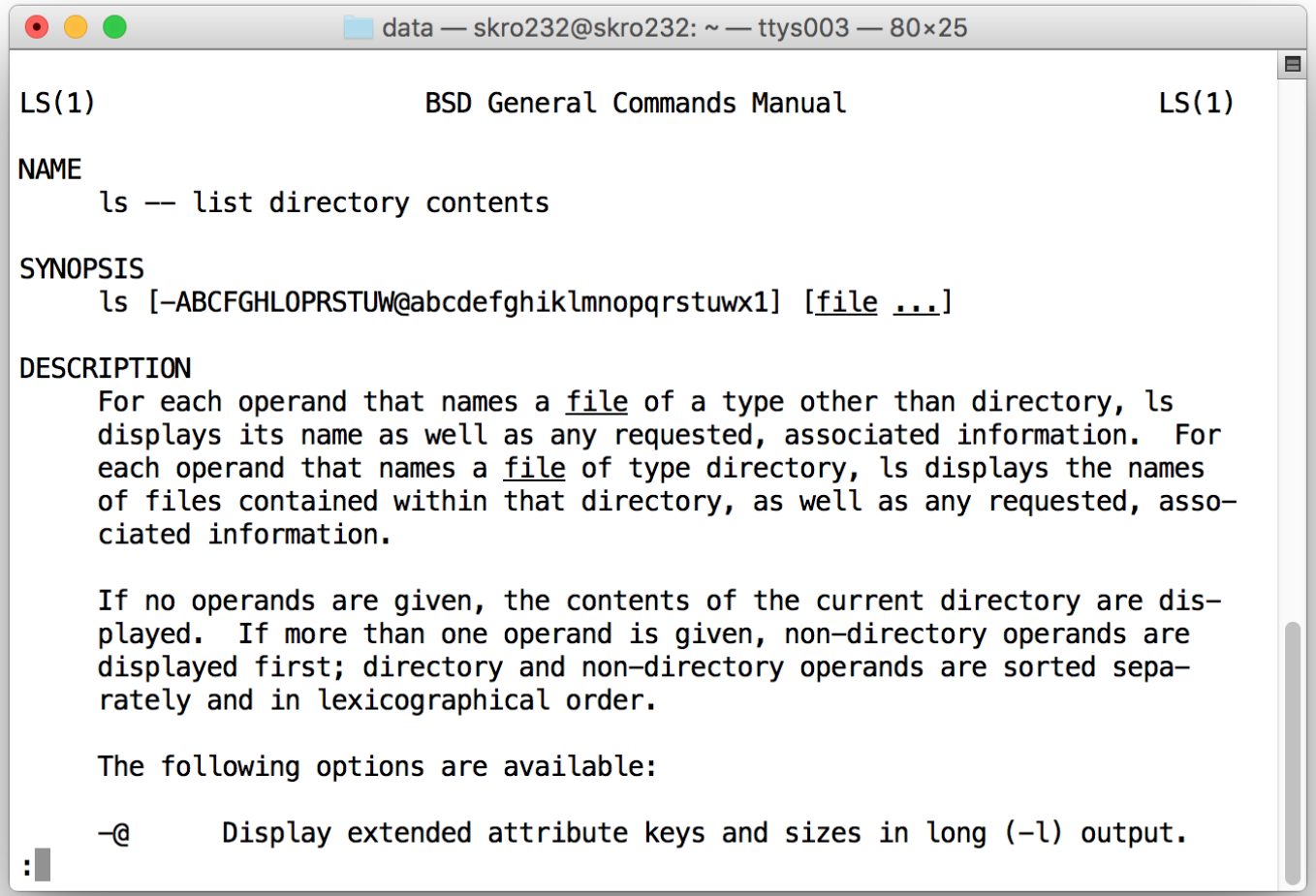
Pipes - Composing programs

- Pipes feed the output from a program to the input of another program
- Allow us to string programs together for “one-of” programs
- Very efficient
  - Creates one process per program
  - File buffer between programs
  - Signals and scheduler coordinate writing and reading data between programs - no extra files needed

Pipe examples



What to type for a given command



- `man` *command* provides the manual page for *command*

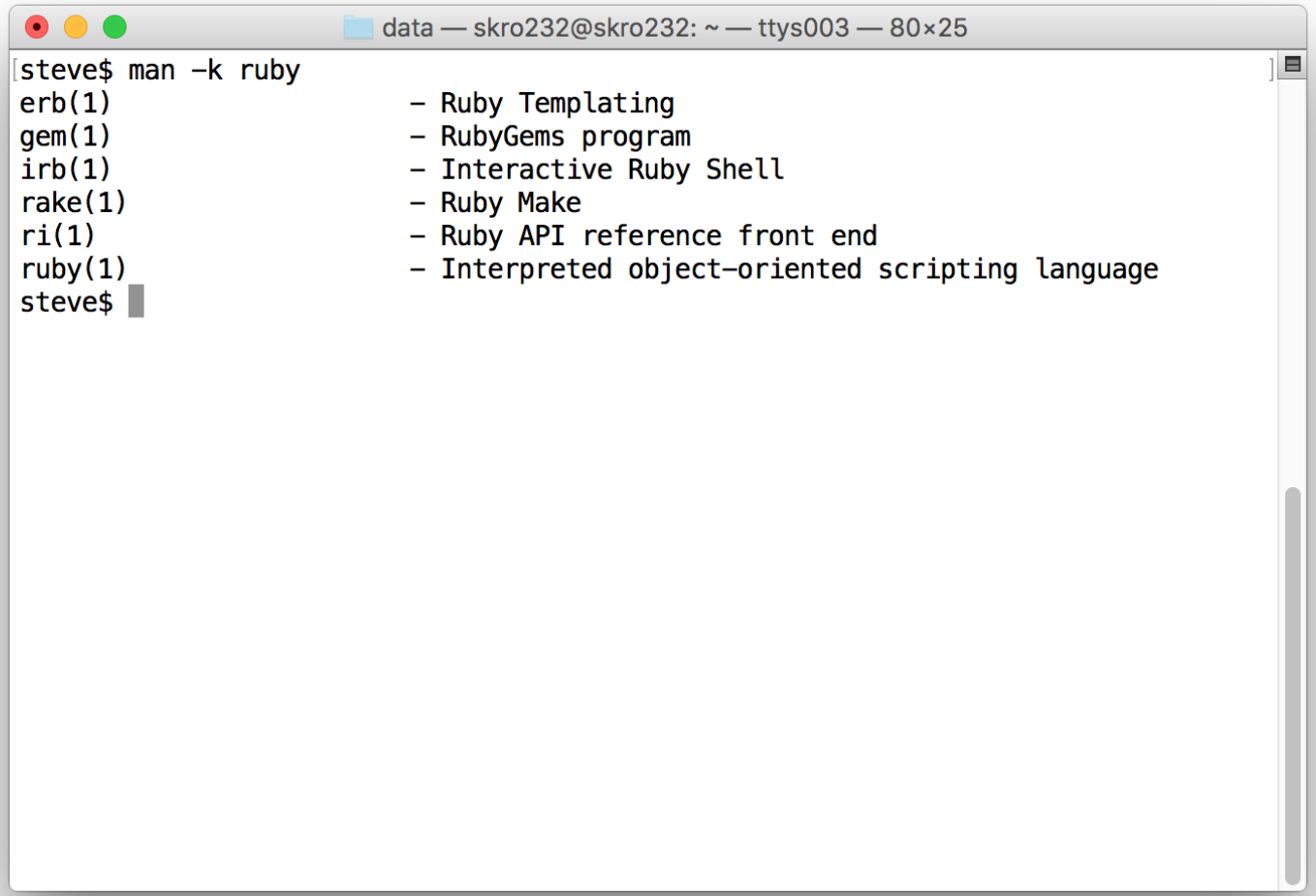
Man pages

- `man` pages contain multiple sections
- Typical sections include:
  - NAME - name of the command and brief discussion
  - SYNOPSIS - how to invoke the command and list of arguments
  - DESCRIPTION - detailed description of the program
  - OPTIONS - options and their description
  - EXAMPLES - example invocations and what they do
  - EXIT STATUS - `exit(2)` codes (0 indicates success)
  - ENVIROMENT - how shell variables affect program
  - SEE ALSO - other programs or documents to consult
  - BUGS - known issues

Man pages - cont.

- UNIX documentation system
- Organized into multiple sections
  - 1 - programs or shell commands
  - 2 - System (kernel) calls
  - 3 - Library calls
  - 4 - Special files (usually found in /dev)
  - 5 - File formats
  - 6 - Games
  - 7 - Miscellaneous
  - 8 - System administration commands (root)
  - 9 - Non-standard kernel commands

Man pages - How to find commands with 'man -k'



Regular expressions - searching strings and finding things

- Regular expressions allow you to search using string patterns ‘
- Many programs in *NIX make use of regular expressions:
- Most characters match themselves, thus ’abc’ matches the string “abc”
- Meta characters:
  - ˆ matches the beginning of a line
  - $ matches the end of a line
  - . matches a single character
  - ? matches zero or one of the preceding pattern
  - * matches zero or more of the preceding pattern
  - + matches zero or more of the preceding pattern
  - [and] indicate the start and end of a character class
  - (and) indicate the start and end of an atom

Regular expressions - examples

- /abc/ matches “abc”
- /[a-z]/ matches a lower case character
- /[a-z][A-Za-z0-9_]+/ matches identifiers in many languages
- /^[0-9]+$/ matches a line containing only a single integer
- /^a.*z$/ matches a line starting with ’a’ and ending with ’z’

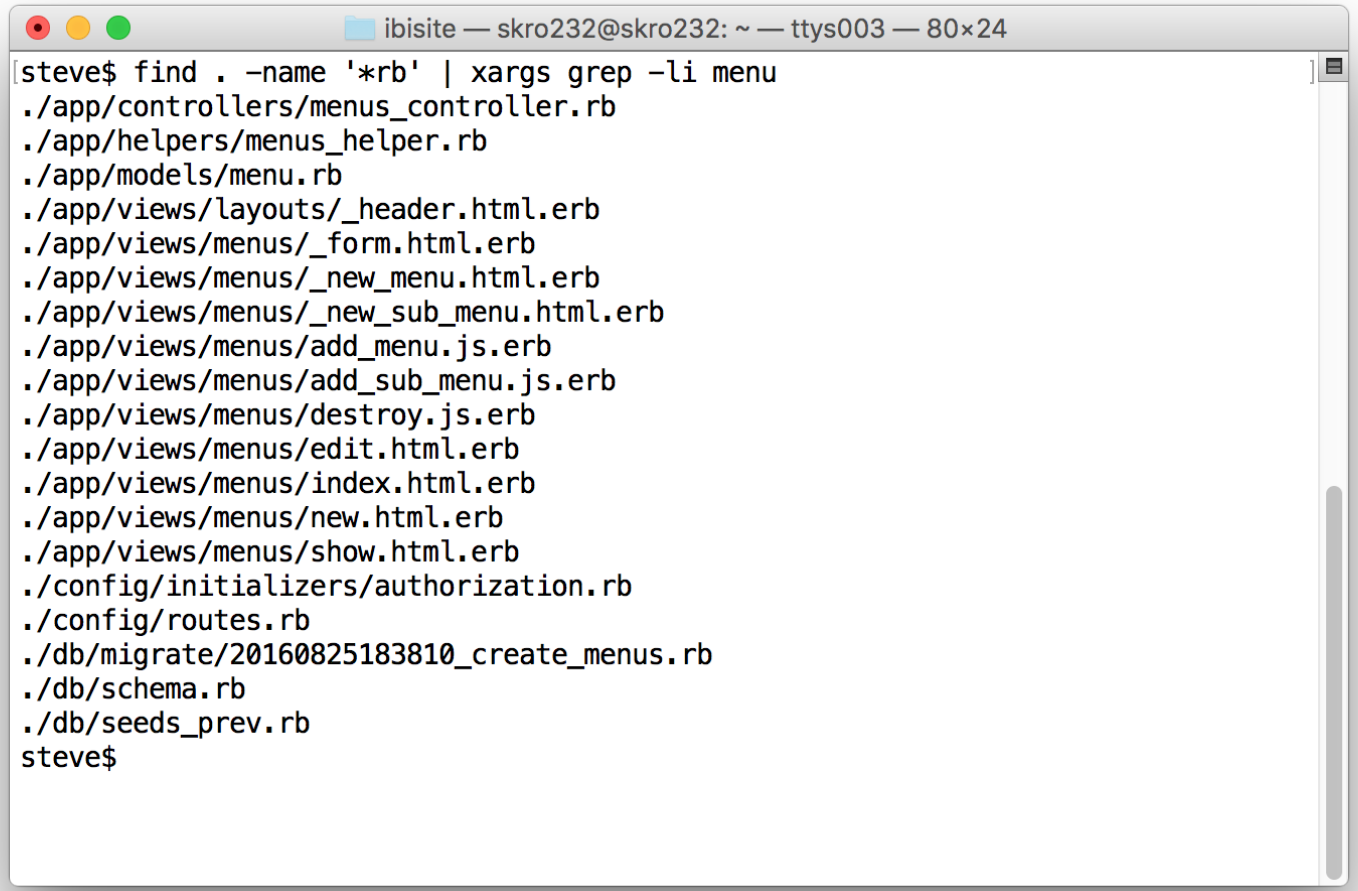
Regular expressions - grep

- grep (Global Regular Expression Print) is the standard search program
- grep -i PATTERN FILES - ignores case
- grep -l PATTERN FILES - prints just the FILES containing PATTERN
- grep -f RE_FILE FILES - reads regular expressions, one per line, from RE_FILE

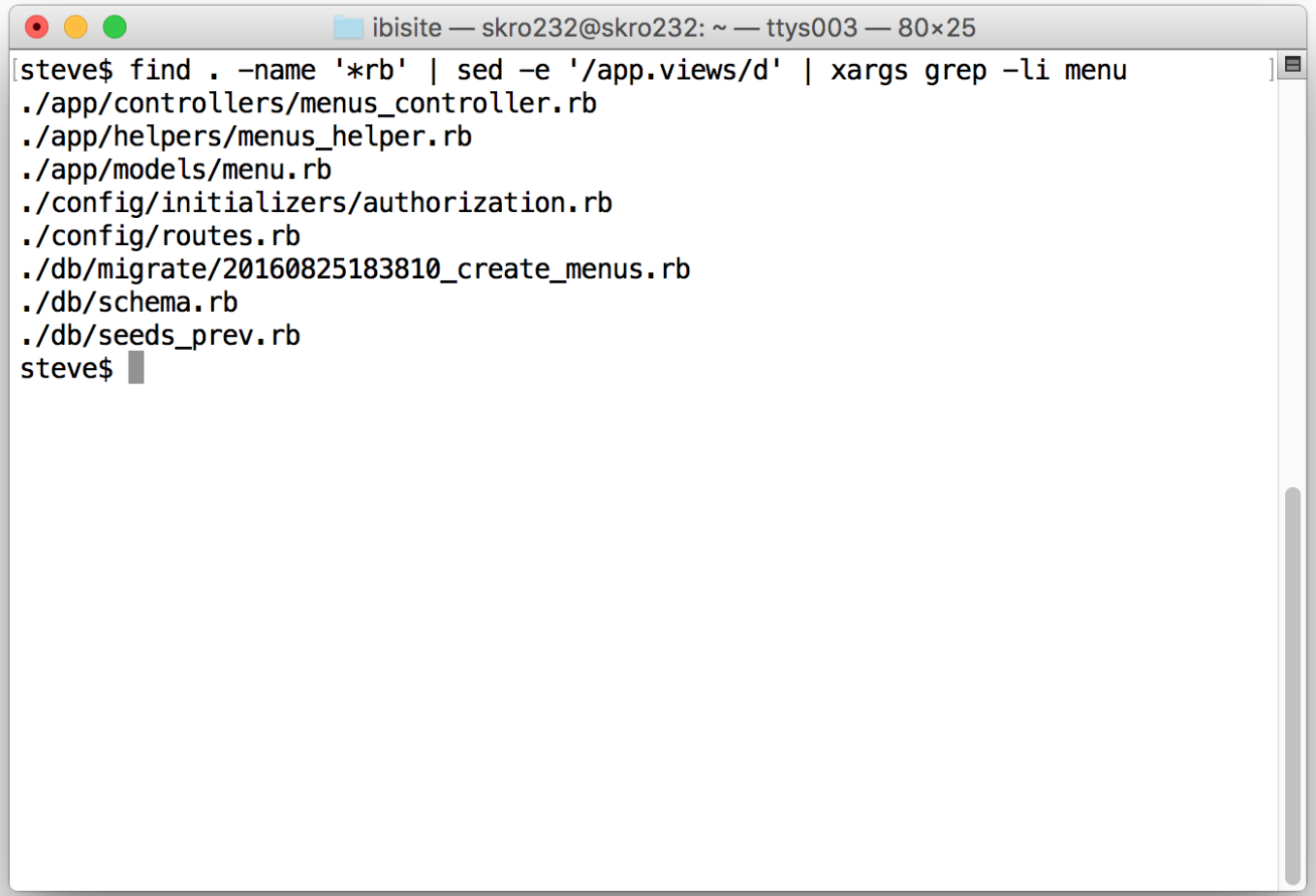
find - Query your file system

- find traverses a file system performing an action on each file it matches
- It provides many filters to fine tune a query:
  - file name pattern
  - file types (regular file, directories, links)
  - file properties (create time, modification time, etc.)
  - owner, group
  - number of links
  - and more
- Default action is print the name of the file

find - Example



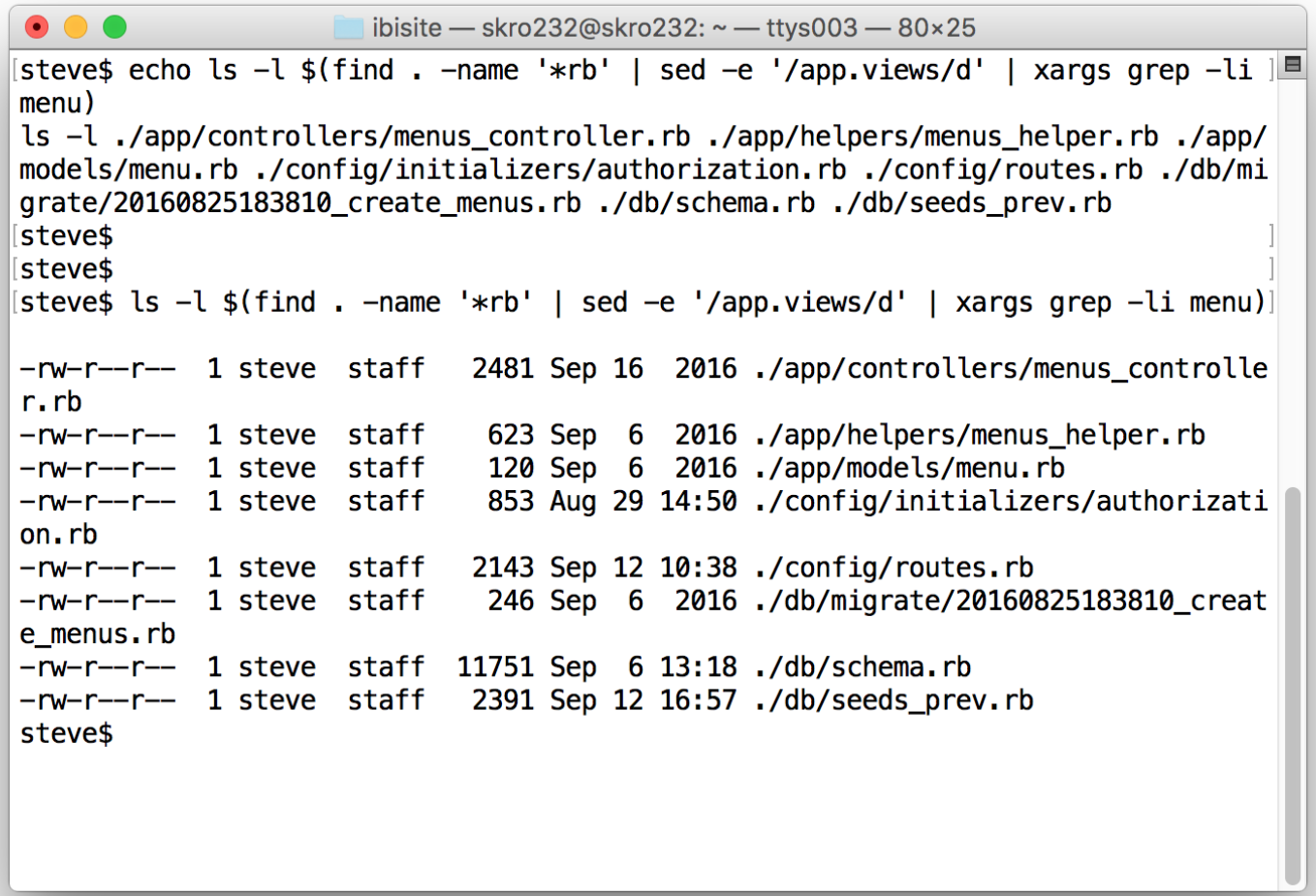
find - Example using sed to filter



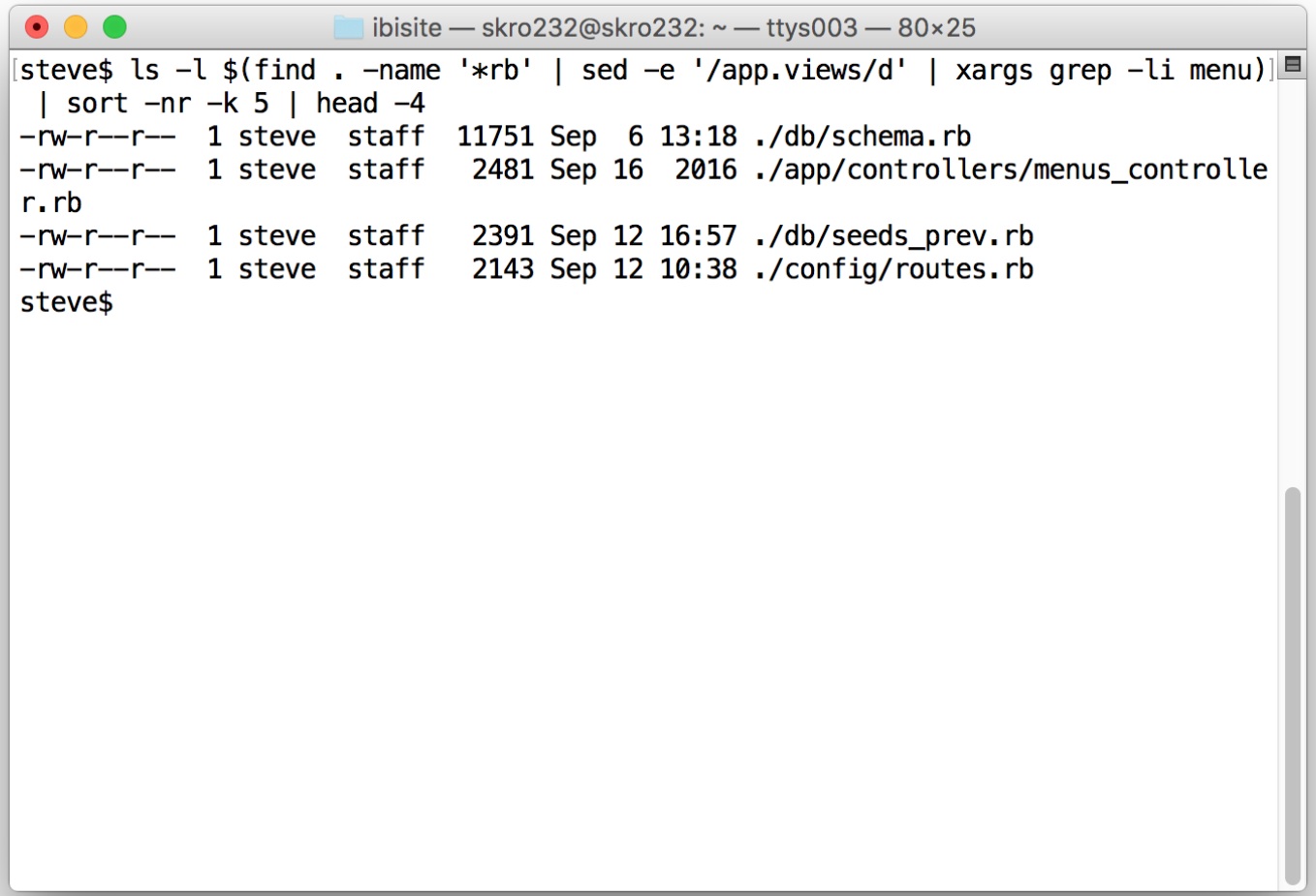
Command substitution - use the shell to write your commands

- `$( pipe )` executes *pipe* and substitutes the text in place

Command substitution - cont



Command substitution - cont



Common programs

- awk - The AWK text processing language
- grep - find strings in files
- less - display a file a screen at a time
- ls - list files
- sed - Stream editor
- sort - Sort lines of a file
- uniq - Filters out repeated lines in sorted files
- wc - Counts characters, words and lines of a file
- xargs - used with `find` to execute a command on multiple files

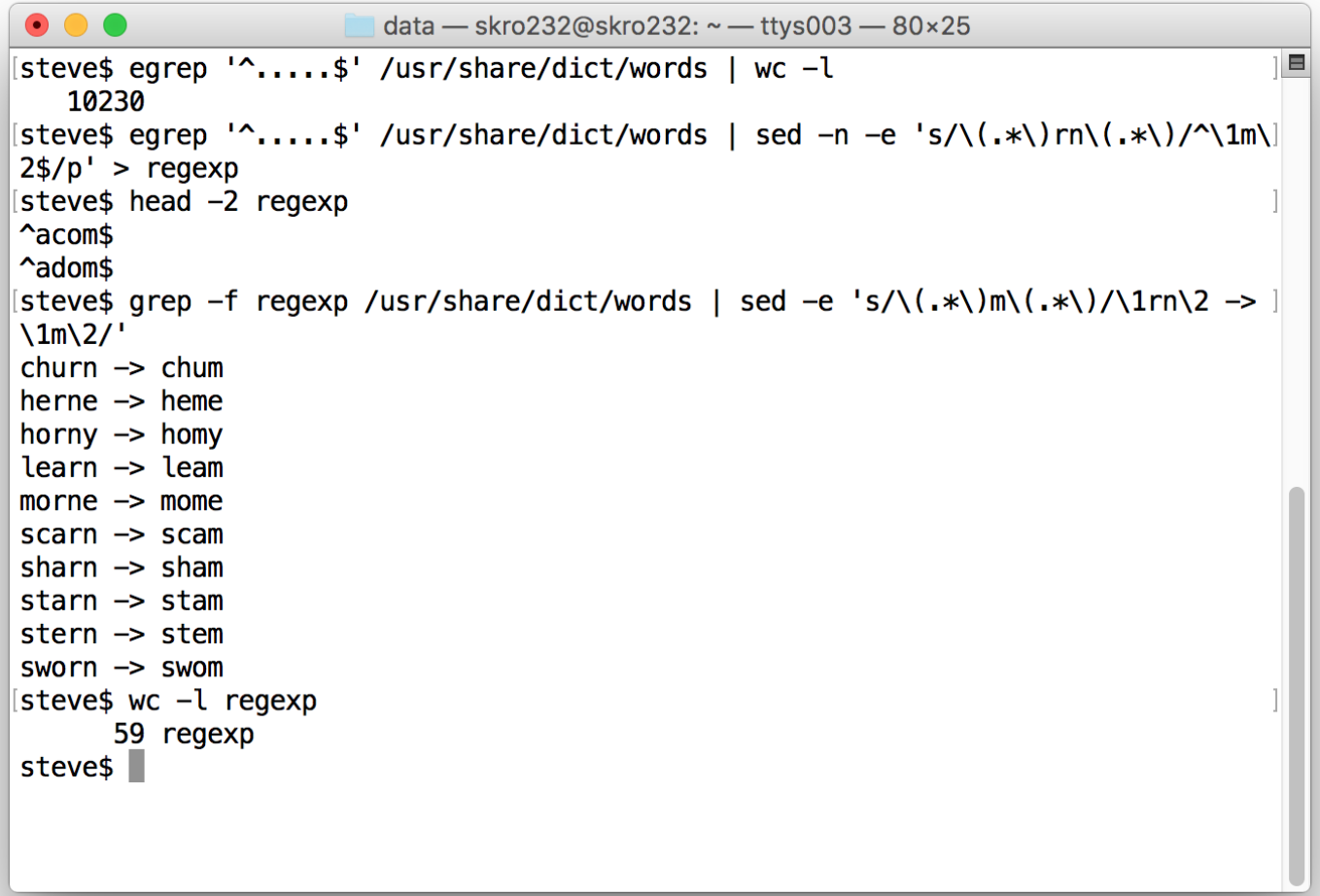
Example problem - Solve NPR Sunday puzzle

From NPR Weekend Edition Sunday, Sept 4, 2016.

Next week's challenge, from listener Norm Baird of Toledo, Wash.: If you squish the small letters "r" and "n" too closely together, they look like an "m." Think of a common five-letter word with the consecutive letters "r" and "n" that becomes its own opposite if you change them to an "m."

About 840 people solved this puzzle.

Let's use our Ancient and Decrepit Tools to solve it in just two lines of code.



Answer: STERN and STEM

Wrapup

- Covered why you might want to use the command line
- Described six concepts essential to efficient use of command line:
  - Command shell
  - Pipes to connect commands
  - Man pages for documentation
  - Regular expressions for text searchig
  - Find to query a file system
  - Command substitution
- Demonstrated efficient command line use to solve an NPR Sunday puzzle in two lines of code

Questions?