The Command-Line Interface Mindset

Session M222

Macworld San Francisco 2005

Scott M. Neal

Senseption

Apple Certified Trainer, ACSA 10.3

Apple Certifications for 10.3.x

Course Names	Certification Level	Length (days)
Mac OS X Help Desk Essentials	ACHDS	3
Mac OS X Server Essentials	ACTC (with Help Desk Essentials)	4
System Administration of Mac OS X Clients	Apple Certified System Administrator	5 days each course
System Administration using Mac OS X Server		

train.apple.com • 800-848-6398

Why use the CLI?

- Remote Administration
- Security & Monitoring
- Automation with Scripts
- Running commands as a different user
- Troubleshooting

Learning the CLI Mindset

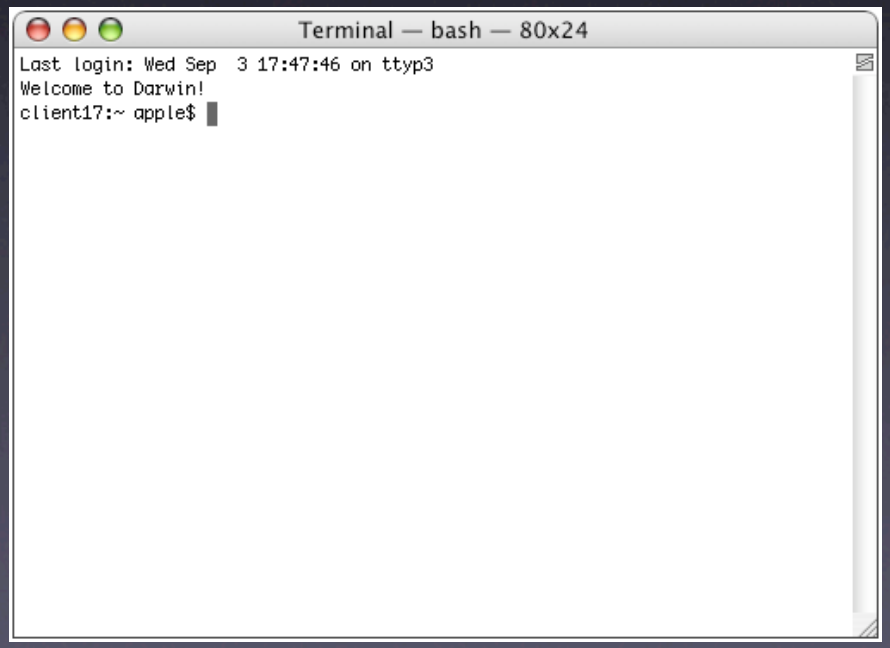
- There is a pattern to the CLI, it's not COMPLETELY random!
- CLI Filesystem representation
- Common CLI Shell syntax
- Many small commands designed to do specific jobs well, but still work together with other commands
- Documentation (so you don't have to memorize everything)

The Environment in which the CLI Arose

- The 70s, a Schizophrenic decade
  - We got Disco, Van Halen, B52s, and the Sex Pistols, ALL AT THE SAME TIME

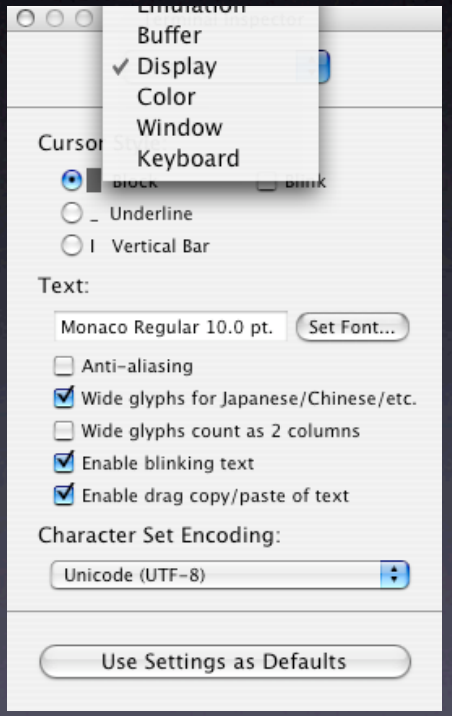
How to get a CLI

- Terminal application
- `ssh`
- Single-User Mode
- >console
- X11



Terminal

- The GUI way to get a CLI
- Allows customization



ssh

- The `ssh` command allows you to access a remote computer using the command line

  `ssh` *`username@hostname`*

  `ssh -l` *`username hostname`*

  - Secure replacement for telnet
  - Remote Login must be enabled on remote machine
  - Authentication takes place on remote machine
  - Commands are executed remotely
  - Can setup public/private key to remove need to type in password
- Example:

  `ssh scott@mac1.pretendco.com`

Single User Mode

- Boot into single user mode by holding Command-s after startup chime
- Starts the computer up in a very lean, mean version of the operating system
  - no user accounts (single-user is root)
  - no networking
  - no peripherals
  - full access to file system of all internal devices
- VERY useful for troubleshooting a system that does not boot

>console Login

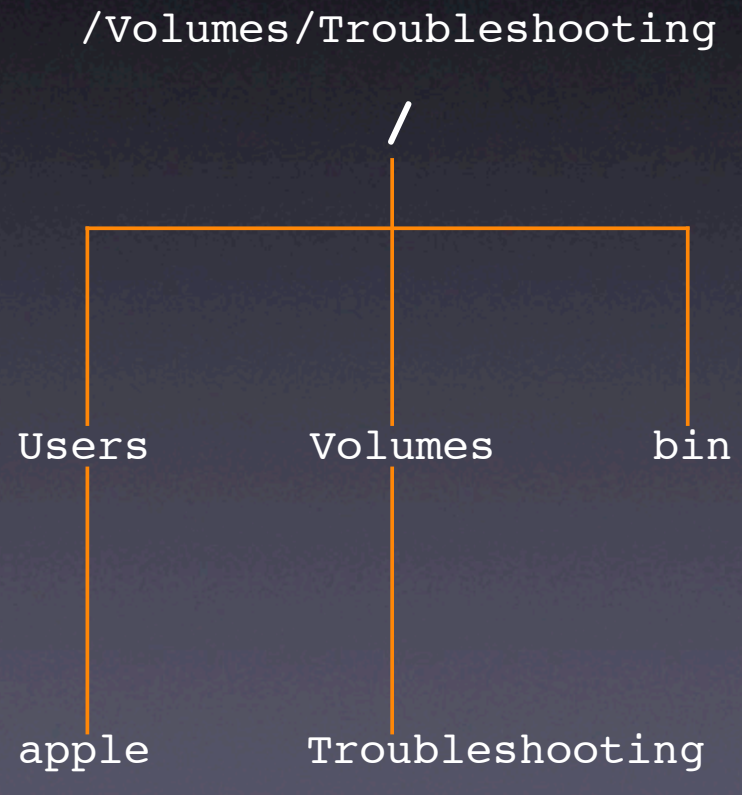
- Instead of using a regular account short name to log in, use >console
- The GUI will disappear, and you will get an old-fashioned VT-100 style login window
- NOT the same as Single User Mode: full OS running (minus GUI)
- VERY useful for debugging user account-specific issues
  - Startup item problems
  - User record configuration issues

X11

- The GUI system for most version of Unix
- Somewhat equivalent to Quartz on OS X
- Support of X11 in OS X allows GUI applications written for other versions of UNIX to function in OS X with little to no code rewrite

File System: GUI & CLI

The command-line's view of the file system consists of a single hierarchy of folders & files, while the Finder shows one or more disks containing folders and files.



Absolute vs. Relative Path

- In the CLI, a file can be referenced with two path types:
  - Absolute: Can be in any folder

    ```
    ls /Users/smn/Documents
    ```

  - Relative: Must be in folder with file/folder you are accessing

    ```
    ls Documents
    ```

Other Path Nomenclature

- Refer to current folder with `.` (no leading /)

  `ls ./Documents`
- Refer to parent folder with `..` (no leading /)

  `ls ../smn/Documents`
- Refer to current logged in user's Home folder with `~` (no leading /)

  `ls ~/Documents`
- Refer to ANY Home folder with `~`*`username`* (no / leading or between `~` and *`username`*)

  `ls ~judy/Public`

Location of Files

- Most OS X GUI applications reside in the folder `/Applications`.
- Operating system files that generally should not be touched are in `/System`.
- Many UNIX configuration files are in `/etc` or `/var`
- Most command-line tools and daemons are in one of the following directories:
  - `/bin`
  - `/sbin`
  - `/usr/bin`
  - `/usr/sbin`
  - `/usr/libexec`

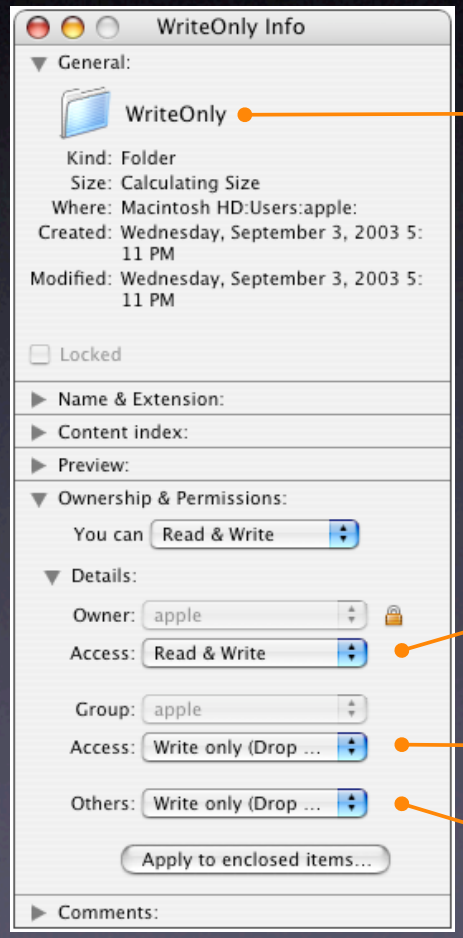
Location of Files

- Command-line tools that ship with the Developer Tools are installed by the DevTool installer in /Developer/Tools.
  - `FileMerge`
  - `CpMac`
- It's a good idea for administrators to install the Developer Tools on both OS X and OS X Server machines they use for admin

Directories and Files Not Seen in the Finder

- /
- Files with names starting with a period (.)
- Files specified in /.hidden
  - Adding/Deleting folders or files to/from the .hidden file will make them invisible/visible in the Finder
    - Files and folders must be at the root level of the file system
    - To hide folders and files that are not at the root level of the file system, use `SetFile`

File Permissions: GUI vs. CLI



Type of file:
d for directory
- for file
l for symbolic link

`drwx-wx-wx`

Owner permissions:
rwx=read, write, and execute

Group permissions:
-wx=write and execute

Others' permissions:
-wx=write and execute

The UNIX Shell

- A UNIX Shell is a portal into the Unix CLI
  - Shells allow commands to be typed, interpreted, and executed
- Shells are applications, and there are MANY
  - SH, BASH, TCSH, Korn Shell, Bourne Shell, ...
  - BASH shell is default in OS X 10.3
- Shells interpret "Shell Scripts"
  - There are also non-Shell script interpreters: Perl, Tcl, Python

Common commands

ls *file ...*
cd *directory*
pwd
cp *file1 file2*
mv *oldFilename newFilename*
rm *file ...*
mkdir and rmdir *directory ...*
echo
cat
scp *file1 file2*
df, du, and lsof
top and ps
grep
chflags
ping
tcpdump
host & hostname
id

Command Patterns

```
ls
ls -lA ~/Documents
ls -lA ~/Documents > /tmp/doc_ls_list.txt
ls -lA ~/Documents >> /tmp/doc_ls_list.txt
ls -lA ~/Documents | grep -i myfile
```

Wildcard Characters

- '*' is a wildcard that matches anything
- '?' matches any single character
- For example, in the directory

`/System/Library/StartupItems/`

  - The command `ls -d Sy*` outputs

    ```
    SystemLog    SystemTuning
    ```

  - The command `ls -d *Servi*` outputs

    ```
    AppServices  DirectoryServices  IPServices
    ```

  - The command `ls -d *System???` outputs

    ```
    SystemLog
    ```

Regular Expressions

- [] : match a character within the brackets
  - [AXj5] will match any of the listed characters
  - [0-9,a-z,A-Z] will match the listed characters in a range
- For example, in the directory

  `/System/Library/StartupItems/`

  - The command `ls -d [B-C]*` outputs

```
BIND                ConfigServer  CoreGraphics
 CrashReporter      Cron
```

Regular Expressions

- {} : match strings of characters within the braces
  - {dog,cat,fish} will match any of the strings in the list
- For example, in the directory

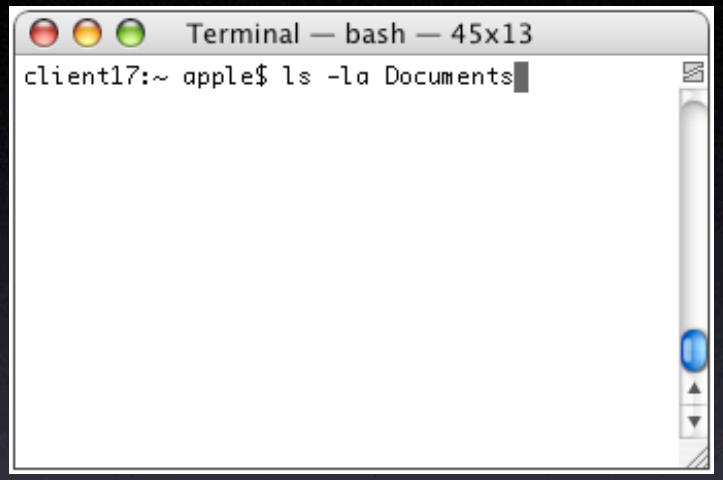
  `/System/Library/StartupItems/`

  - The command `ls -d {Net,Sys}*` outputs

```
NetInfo  NetworkExtensions     SystemLog
Network  NetworkTime           SystemTuning
```

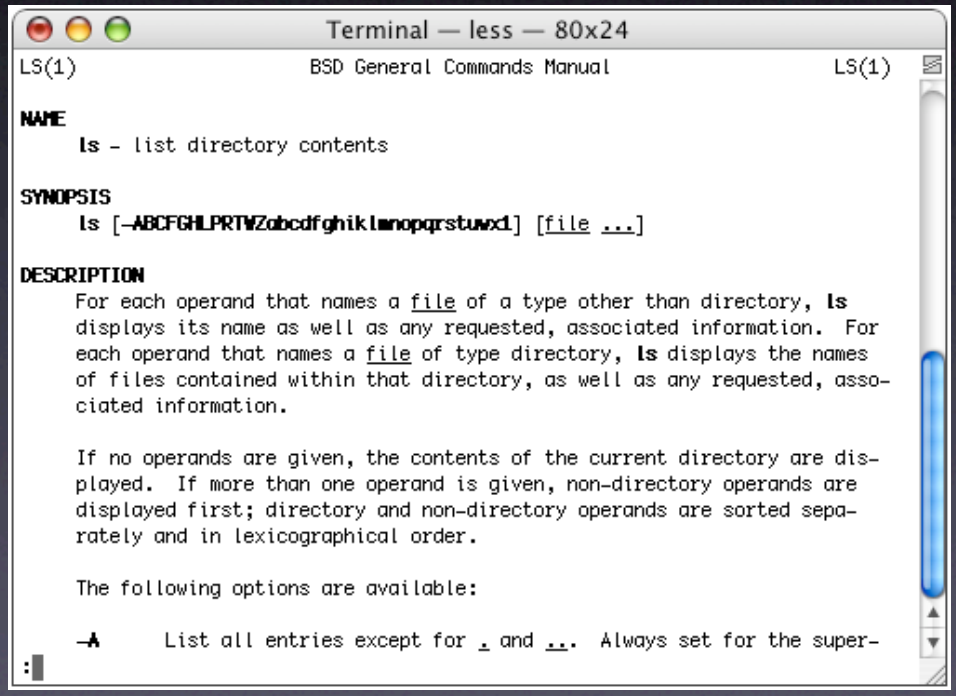
CLI Typing Shortcuts

-
- Unix Shell
  - Tab completion
  - Arrow keys
  - `Control-a`, `Control-e`
  - `Control-f`, `Control-b`
  - `Esc f`, `Esc b`
  - `Control-l`, clear screen
  - `Control-u`, clear line
- Terminal application-specific
  - `command-k`



CLI Documentation

-
- `man`
- `apropos`
- `makewhatis`
- `locate`



man page

- Different sections (Chapters) in man page
  - Can specify another section with argument to man (section 1 by default)

```
$ man ls
...
SEE ALSO
     chflags(1), chmod(1), sort(1),
xterm(1), termcap(5), symlink(7),
sticky(8)
$ man 8 sticky
```

man page (cont.)

- If no man page, try executing command with no args, or with -h or -help argument

```
$ man screencapture
No manual entry for screencapture
$ screencapture
screencapture: illegal usage, file
required if not going to clipboard
usage: screencapture [-icmwsWx] [file]
...
```

OS X CLI Tools: Who to Thank/Blame

- CLI tools originate from different working groups for OS X
  - BSD
  - Darwin
  - Mac OS
  - Shell built-in (specific to shell)
  - Others (GNU, 3rd party, ?)

OS X CLI Tools: Who to Thank/Blame

- Questions about generic BSD or shell built-in commands can be directed to BSD or shell information sources
  - OS X is a first-class BSD and shell citizen
- Darwin or Mac OS specific commands should be directed to OS X-based information sources
  - KBase
  - OS X mailing lists & newsgroups

Special Characters

- Include <space>, $, #, [,], {, }, !, =, <, >, ., &, ;, |, “, ‘, `, (,), and the wildcard characters
- Cannot represent themselves unless quoted or escaped with the backslash character:

```
echo “Use the # character.”
echo Use the “#” character.
echo Use the \# character.
```

Spaces

- A space is special because it is a separator.
- Spaces in file names must be quoted or escaped with the backslash character (\), for example:

```
ls "My List”
ls My” “List
ls My\ List
```

- UNIX file systems support spaces in filenames, it’s the shell that needs the “escaping”

Command Output Substitution

- Substitute output of a command by enclosing it in backquotes (also called grave accent marks)
  - Assign the output of `id -u`, which is your User ID, to the environment variable `UID`:

    ```
    UID=`id -u`
    ```

  - Assign the output of cat myfile, which is the contents of myfile, to the variable WORDLIST

    ```
    WORDLIST=`cat myfile`
    ```

Stopping and Suspending processes

- Control-c
  - Stops the execution of a process
  - CLI equivalent to old Mac OS Option-.
- Control-z
  - Suspends the execution of a process
  - Once suspended, process can be backgrounded, foregrounded, or killed
  - Sometimes a process will not respond to Control-c, but will respond to Control-z

Running commands in the background

- Use the ampersand character (&) to make a process run in the background
- The PID is written to the screen
  - Example:

```
$ ditto -rsrcFork /Users/smn /Volumes/\
mybackup/smn/. &
[1] 3017
$ type happily here while ditto executes
```

Running commands in the background

- Preexisting CLI-launched processes can be backgrounded with Control-Z and the `bg` command
- Example:

```
$ longcommand -sk /
```

(I'm sick of waiting, so I press Control-Z)

```
^z
[1]+  Stopped                 longcommand -sk /
$ bg
[1]+ longcommand -sk / &
```

Running commands in the background

- Commands can be “foregrounded” with `jobs` and `fg`
- Example:

```
$ jobs
[1]+  Running      longcommand -sk / &
$ fg %1
```

Environment variables

- Preset values from OS, but you can modify (specific to shell)
- Type `env` or `printenv` to view them
  - Examples:

```
HOME=/Users/scott
USER=scott
LANG=en_US
PATH=~scott/bin:/usr/local/bin:/usr/
bin:/bin:/usr/local/sbin:/usr/sbin:/
sbin
GROUP=staff
HOST=scotts_mac
EDITOR=vi
```

Assigning and using Environment variables

- Shell specific
- For bash, assign a value to a variable with the equals sign (=)
  - Do not put spaces before or after =
  - Example: `MYNAME=Scott`
- To use the value of a variable, precede it with the dollar sign ($)
  - Example: `echo $MYNAME`
- The value of a variable that has not been assigned is an empty string

`path` environment variable and `which`

- User-specified order in which commands are searched in CLI
- If absolute path for command not specified, first one matching name in `path` will be used
- Use `which` to find out which command is used by default
  - Example:

```
$ which ls
/bin/ls
```

`path` environment variable and `which`

- If you want to use a specific version of a command and ignore the search through `path`, specify absolute path to command
- Often need to specify current directory to run command or script, since it may not be in your path
  - Example:

```
$ /usr/local/bin/ls
$ ./ls
```

OS X Resource Searching

- OS X is designed to search for resources in a certain order
- Most resources are searched in the following order:
  - ~/Library
  - /Library
  - /Network/Library
  - /System/Library
- Unix `path` and `manpath` environment variable not used here

Pipes

- Commands can be “plumbed together” by a vertical bar (|) called a pipe.
- Output of the first command becomes the input to the second command (and so on and so on)
- Supports UNIX philosophy of having each command do its job well and not try to do too much
  - ”Kitchen Sink” commands unnecessary

Pipes

- Examples:

```
ps -ax | grep httpd
```

  - Runs the ps -ax command and pipes its output into grep
  - The grep command prints the lines that contain "httpd"

```
ls | wc -w
```

  - Runs the ls command, which prints the files in the current directory, and pipes its output into the wc command
  - The wc -w command prints the number of words in its input, which, in this case, is the number of files in the current directory

```
NUMFILES=`ls| wc -w`
```

  - Assigns the number of files in the current directory to `NUMFILES`

Command-Line Issues

- Resource forks
- No trash or undo
- O and 0 (uppercase letter O and the numeral zero)
- l and 1 (lowercase letter l and the numeral one)
- Spaces
- Man page sections
- Man pages out-of-date
- Man page does not exist

Text Editors

- The OS consists of many, many configuration files
- When editing these files, need a program to do the editing
- Typical text editors
  - `pico`
  - `vi`
  - `emacs`
  - TextEdit

Useful File Commands

- Display the contents of a text file a page at a time:

```
more file
less file
tail {-n num_of_lines} file
tail -f file
```

- Create an empty file or change the modified date on that file:

```
touch file
```

Useful File Commands

- Find a file with certain attributes:

  ```
  find /Users -name \*.mp3
  ```

- Append a text file to the end of another text file:

  ```
  cat file1 >> file2
  ```

- Determine the type of file:

  ```
  file file
  ```

Darwin & Mac OS X Specific Commands

```
open
ditto
screencapture
lsbom
atlookup
certtool
ipconfig
osascript
CpMac
SetFile & GetFileInfo
```

Darwin & Mac OS X Specific Commands

```
system_profiler
dscl
nidump & niload
diskutil
hdiutil
lookupd
dsconfigad
installer
softwareupdate
sw_vers
```

Application Bundles

- Applications can be double-clicked to run from the Finder, but are actually specially-treated Folders
- From the Finder, the contents of an application bundle can be opened by "right button" (control) clicking
- From the CLI, you can "drill down" into the application bundle and launch applications, even GUI applications
  - Example:

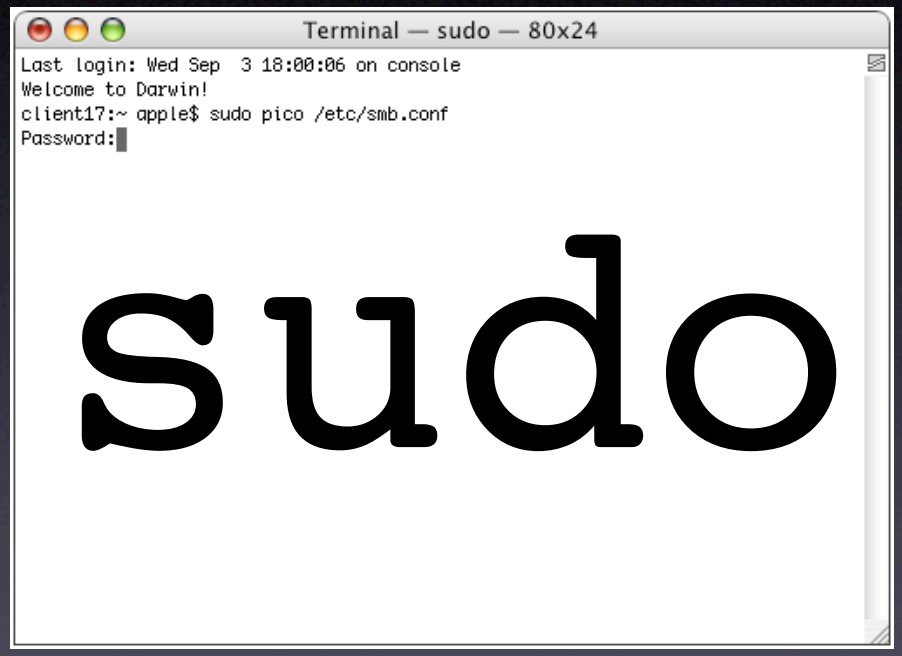
```
$ /Applications/TextEdit.app/Contents/
MacOS/TextEdit
```

The open command

- From the CLI, `open` invokes LaunchServices in the same manner as double-clicking a file from the Finder
- LaunchServices ascertains which application should launch the desired file
  - CLI tools by default depend on Environment variables like EDITOR, do not use LaunchServices
  - Examples

```
$ open /etc/named.conf
$ open ~/Documents
$ open ~/Library/Keychains/login.keychain
$ open ~/Library/Preferences/com.apple.finder.plist
```

Executing Command as System Administrator



Warning: `sudo` does not work with shell built-in commands (see `man builtin`)

`telnet` to Service Port

- You can telnet directly to a port and communicate with a service, just as the daemons themselves do
  - Good for quick-check: firewalls, service running
  - Can get and set useful information
  - For more than just quick-check, need to understand low-level protocol details
- /etc/services has list of ports and their assigned usage
- Typical ports to test include
  - SMTP — smtp — 25
  - Password Server (10.3) — apple-sasl — 3659
  - Apache WWW — http — 80

Startup Items

- Unique to OS X method of launching services when OS is started
- Custom startup items placed in

  `/Library/StartupItems`

  - May need to make directory if doesn't already exist
  - Do NOT put your custom Startup Items in

    `/System/Library`
- Includes the following files:

`/Library/StartupItems/`*`startup_item_name`*`/`

`startup_item_name`	Executable Script
`StartupParameters.plist`	run Order spec
`Resources`	(optional)

cron

- Use OS to invoke periodic events
  - System crontab
    - file located in `/etc/crontab`
    - Edited directly
  - User or Administrator crontab
    - file(s) located in `/var/cron/tabs`
    - Edited using crontab command
- See man page for different periodic interval parameters

Quotas

- Commands set and manage soft and hard quotas from CLI
  - `quotacheck`
  - `quotaon and quotaoff`
  - `edquota` uses `EDITOR` env variable
  - `repquota`
  - `quota`
- Create the files needed
  - `.quota.user & .quota.opts.user`
  - `.quota.groups & .quota.opts.group`
- OS X does not automatically notify user of quota violations
  - Hard to check from Finder

Synopsis

- There is a mindset involved in learning the CLI
- Do NOT attempt to master the CLI simply by memorizing commands
  - The most common commands will be memorized automatically by repeated usage
- Wielding the power of the CLI allows you to do things that are more difficult or impossible to do in the GUI

Thank You!
The Command-Line Interface Mindset

Session M222

Macworld San Francisco 2005

Scott M. Neal

Senseption

Apple Certified Trainer, ACSA 10.3