# Getting Comfortable With the command-line

Daniel Freedman

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# What Is The Command-Line?

- The command-line is a text based interface to interacting with your computer.

# What Is The Command-Line?

- The command-line is a text based interface to interacting with your computer.
- To use the command-line, one enters commands with arguments and hits "Enter" to make the command execute.

# What Is The Command-Line?

- The command-line is a text based interface to interacting with your computer.
- To use the command-line, one enters commands with arguments and hits "Enter" to make the command execute.
- More Info on Wikipedia

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# Xterm

- Xterm is best known and most compliant terminal emulator for *nix operating systems.

# Xterm

- Xterm is best known and most compliant terminal emulator for *nix operating systems.
- Terminals created by Gnome, KDE, XFCE, etc. usually wrap xterm to provide support for tabs and copy/paste with keyboard combos.

# Terminal.app

- Terminal.app is the default terminal provided by Apple in Mac OS X.

# Terminal.app

- Terminal.app is the default terminal provided by Apple in Mac OS X.
- It is very fast, but only supports 8 colors unlike xterm's 256 (useful for syntax highlighting).

# Terminal.app

- Terminal.app is the default terminal provided by Apple in Mac OS X.
- It is very fast, but only supports 8 colors unlike xterm's 256 (useful for syntax highlighting).
- Macs can also use xterm by installing X11 and using the provided X11 xterm.

# Cygwin

- Cygwin emulates a unix-like terminal on Windows, but allows for Windows style file paths and syntax.

# Cygwin

- Cygwin emulates a unix-like terminal on Windows, but allows for Windows style file paths and syntax.
- Cygwin requires tools and applications be compiled inside cygwin for compatibility.

# SUA

- SUA stands for Subsystem for Unix Applications.

# SUA

- SUA stands for Subsystem for Unix Applications.
- SUA is provided by Microsoft as a pure UNIX application layer for Windows, without any compatibility for filepaths or syntax

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# The Easy And Common Tools I

- echo: Repeats what was typed back out

```
sls@2405 cmdline $ echo "Hello World"
Hello World
```

- mv: Moves files from arguments 0 through N-1 to argument N, also used for renaming files

```
sls@2405 cmdline $ mv samples/file1 hi.txt
```

- cp: Copies a file from argument 1 to argument 2

```
sls@2405 cmdline $ cp hi.txt hello.txt
```

- rm: Removes files specified

```
sls@2405 cmdline $ rm hello.txt
```

# The Easy And Common Tools II

- mkdir: Makes a new directory

```
sls@2405 cmdline $ mkdir newdir
```

- ls: Lists files in the current directory

```
sls@2405 cmdline $ ls
hi.txt
newdir
samples
```

- cat: Prints out the contents of a file

```
sls@2405 cmdline $ cat hi.txt
Hello There
```

- grep: Lists the lines in the specified files that match the given expression

```
sls@2405 cmdline $ grep "Hello" hi.txt
Hello There
```

# The Easy And Common Tools III

- ssh: Remotely log into another machine

```
sls@2405 cmdline $ ssh dfreedm2@yt.acm.uiuc.edu
dfreedm2@yt /home/dfreedm2 $
```

- less: Saves output from a command into a scrollable window

```
sls@2405 cmdline $ less samples/bigfile
This is pdfTeX, Version 3.1415926-1.40.10 (TeX Live
2009/Debian) (format=pdflatex 2010.8.27) 11 SEP 2010
17:49
samples/bigfile lines 1-3/1162 0%
```

- ln: With the '-s' flag, ln can make "symlinks" that provide a pointer to the real location of a file.

```
sls@2405 cmdline $ ln -s hi.txt pointer
```

# The Easy And Common Tools IV

- sudo: Run a command as the 'root', the super user.

```
sls@2405 cmdline $ whoami
sls
```

```
sls@2405 cmdline $ sudo whoami
root
```

- dmesg: Print the system log

```
sls@2405 cmdline $ dmesg
[71012.760576] Initializing CPU#1
[71012.760576] CPU: L1 I cache:  32K, L1 D cache:  32K
[71012.760576] CPU: L2 cache:  3072K
[71012.760576] CPU 1/0x1 -> Node 0
```

# The Easy And Common Tools V

- head: Read only the first $N$ lines from a file. Default $N = 20$. Adjustable with the `-n` flag.

```
sls@2405 cmdline $ head -n 1
This is pdfTeX, Version 3.1415926-1.40.10 (TeX Live
2009/Debian) (format=pdflatex 2010.8.27) 11 SEP 2010
17:53
```

- tail: Read only the last $N$ lines from a file. Default $N = 20$. Adjustable with the `-n` flag.

```
sls@2405 cmdline $ tail -n 1
161 words of extra memory for PDF output out of 10000
(max.  10000000)
```

# Output Redirection

- Output Redirection is the ability to move text between applications, directly into files, or directly from files.
- |: Pipe moves text from the application on the left to the application on the right.

```
sls@2405 cmdline $ dmesg | less
```

- >: Redirect output from a program into a file.

```
sls@2405 cmdline $ dmesg > log
```

- <: Read a file directly into the input of a program.

```
sls@2405 cmdline $ tr a-zA-Z < log
```

# Tar

- Tar is the default compression application in most Unix based OSes.

# Tar

- Tar is the default compression application in most Unix based OSes.
- Tar is quite complicated, so for the purposes of this talk, we will only discuss extracting and creating archives.

# Tar

- Tar is the default compression application in most Unix based OSes.
- Tar is quite complicated, so for the purposes of this talk, we will only discuss extracting and creating archives.
- Tar just concatenates files into one blob, so other helper programs such as gzip and bzip2 are used to compress the blob.

# Tar

- Tar is the default compression application in most Unix based OSes.
- Tar is quite complicated, so for the purposes of this talk, we will only discuss extracting and creating archives.
- Tar just concatenates files into one blob, so other helper programs such as gzip and bzip2 are used to compress the blob.
- Creating a tar archive can be done with "tar czvf", which invokes gzip as the compression program

```
sls@2405 cmdline $ tar czvf tarball.tar.gz hi.txt
hi.txt
```

- Extracting an archive can be done with "tar xvf", which automatically picks
the compression program to use for decompression *on most machines*.

```
sls@2405 cmdline $ tar xvf tarball.tar.gz
hi.txt
```

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# dpkg

- dpkg is the package management system for Debian based systems.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.
- `aptitude search` returns packages that match what you search for.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.
- `aptitude search` returns packages that match what you search for.
- `aptitude install` will install the package, and install all dependencies.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.
- `aptitude search` returns packages that match what you search for.
- `aptitude install` will install the package, and install all dependencies.
- `aptitude update` will pull update the package listings from the server, and alert you to new updates.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.
- `aptitude search` returns packages that match what you search for.
- `aptitude install` will install the package, and install all dependencies.
- `aptitude update` will pull update the package listings from the server, and alert you to new updates.
- `aptitude safe-upgrade` will try to safely upgrade all the packages.

# dpkg

- dpkg is the package management system for Debian based systems.
- dpkg is rather barebones, so a good frontend for this system is found in aptitude.
- aptitude handles all the package fetching and dependency resolution for you, in a safe and easy way.
- `aptitude search` returns packages that match what you search for.
- `aptitude install` will install the package, and install all dependencies.
- `aptitude update` will pull update the package listings from the server, and alert you to new updates.
- `aptitude safe-upgrade` will try to safely upgrade all the packages.
- Other uses exist, but read the man page.

# rpm

- rpm is the package management system for Red Hat based systems.

# rpm

- rpm is the package management system for Red Hat based systems.
- yum is the most used frontend for these systems.

# rpm

- rpm is the package management system for Red Hat based systems.
- yum is the most used frontend for these systems.
- `yum search -C` returns packages that match what you search for.

# rpm

- rpm is the package management system for Red Hat based systems.
- yum is the most used frontend for these systems.
- `yum search -C` returns packages that match what you search for.
- `yum install` will install the package, and install all dependencies.

# rpm

- rpm is the package management system for Red Hat based systems.
- yum is the most used frontend for these systems.
- `yum search -C` returns packages that match what you search for.
- `yum install` will install the package, and install all dependencies.
- `yum update` will pull update the package listings from the server, and alert you to new updates.

# rpm

- rpm is the package management system for Red Hat based systems.
- yum is the most used frontend for these systems.
- `yum search -C` returns packages that match what you search for.
- `yum install` will install the package, and install all dependencies.
- `yum update` will pull update the package listings from the server, and alert you to new updates.
- `yum upgrade` will try to safely upgrade all the packages.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers
- Just follow the Guide to Installing Git.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers
- Just follow the Guide to Installing Git.
- `brew search` returns packages that match your search terms.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers
- Just follow the Guide to Installing Git.
- `brew search` returns packages that match your search terms.
- `brew install` fetchs, compiles, and installs a package.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers
- Just follow the Guide to Installing Git.
- `brew search` returns packages that match your search terms.
- `brew install` fetchs, compiles, and installs a package.
- `brew outdated` returns which packages are outdated.

# brew

- Macs do not have a built in package management system as powerful as dpkg, but there are good third party ones.
- The most popular/usable is brew.
- brew is based on git, which means that you must install git first using the Mac installers
- Just follow the Guide to Installing Git.
- `brew search` returns packages that match your search terms.
- `brew install` fetchs, compiles, and installs a package.
- `brew outdated` returns which packages are outdated.
- `brew install $(brew outdated)` upgrades outdated packages.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.
- Fortunately, most open source software comes with a standard set of tools to help build the project and install it.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.
- Fortunately, most open source software comes with a standard set of tools to help build the project and install it.
- This technique does not handle dependencies, so you'll have to manage those on your own.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.
- Fortunately, most open source software comes with a standard set of tools to help build the project and install it.
- This technique does not handle dependencies, so you'll have to manage those on your own.
- `configure` is a script that automatically checks if you have the required libraries installed and sets up the Makefile.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.
- Fortunately, most open source software comes with a standard set of tools to help build the project and install it.
- This technique does not handle dependencies, so you'll have to manage those on your own.
- `configure` is a script that automatically checks if you have the required libraries installed and sets up the Makefile.
- Assuming all went well, `make` will read the Makefile and compile the code.

# Other

- If a package you need is not included in brew or your linux repo's repository, you're going to have to build it yourself.
- Fortunately, most open source software comes with a standard set of tools to help build the project and install it.
- This technique does not handle dependencies, so you'll have to manage those on your own.
- `configure` is a script that automatically checks if you have the required libraries installed and sets up the Makefile.
- Assuming all went well, `make` will read the Makefile and compile the code.
- `make install` will then install the code into the right spot.

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.
- Another smaller editor is `nano`, which is less powerful, but more accessible.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.
- Another smaller editor is `nano`, which is less powerful, but more accessible.
- These two editors are so vast that they quickly fall out of the scope of this talk, but you can learn more about them from the links below.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.
- Another smaller editor is `nano`, which is less powerful, but more accessible.
- These two editors are so vast that they quickly fall out of the scope of this talk, but you can learn more about them from the links below.
- Vim: `http://www.vim.org`.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.
- Another smaller editor is `nano`, which is less powerful, but more accessible.
- These two editors are so vast that they quickly fall out of the scope of this talk, but you can learn more about them from the links below.
- Vim: `http://www.vim.org`.
- Emacs: `http://www.gnu.org/software/emacs/`.

# Editors

- Commandline editors are loved by some, confusing to most, and the topic of many flamewars.
- The two big command line editors are `vim` and `emacs`, which are very different and have whole communities behind them.
- Another smaller editor is `nano`, which is less powerful, but more accessible.
- These two editors are so vast that they quickly fall out of the scope of this talk, but you can learn more about them from the links below.
- Vim: `http://www.vim.org`.
- Emacs: `http://www.gnu.org/software/emacs/`.
- Nano: `http://www.nano-editor.org/`.

# screen

- screen is a terminal multiplexer, and allows you to have multiple running terminal sessions at once and in the background.

# screen

- screen is a terminal multiplexer, and allows you to have multiple running terminal sessions at once and in the background.
- In addition, screen can be used to run commands remotely that can survive terminal disconnects.

# screen

- screen is a terminal multiplexer, and allows you to have multiple running terminal sessions at once and in the background.
- In addition, screen can be used to run commands remotely that can survive terminal disconnects.
- screen has many features outside the scope of this talk.

# screen

- screen is a terminal multiplexer, and allows you to have multiple running terminal sessions at once and in the background.
- In addition, screen can be used to run commands remotely that can survive terminal disconnects.
- screen has many features outside the scope of this talk.
- More info can be found on the homepage: `http://www.gnu.org/software/screen/`.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.
- Now create a file with your name in it called name.txt.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.
- Now create a file with your name in it called name.txt.
- `git add name.txt` will tell git to add the file's changes to the repo.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.
- Now create a file with your name in it called name.txt.
- `git add name.txt` will tell git to add the file's changes to the repo.
- `git commit` will then commit the changes to the repo.

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.
- Now create a file with your name in it called name.txt.
- `git add name.txt` will tell git to add the file's changes to the repo.
- `git commit` will then commit the changes to the repo.
- `git log` will show you that you commited the file to the repo!

# Version Control

- Version Control systems are vital to correctly maintaining code which multiple people access.
- The two major players are `git` and `svn`.
- The commands for both are near identical, and you'll have `svn` experience in classes, so we will only cover `git` here.
- `git clone git://github.com/azakus/Command-Line-Repository.git` will clone the repository that contains this talk.
- Now create a file with your name in it called name.txt.
- `git add name.txt` will tell git to add the file's changes to the repo.
- `git commit` will then commit the changes to the repo.
- `git log` will show you that you commited the file to the repo!
- SourceMage has a rather useful guide to beginning, intermediate, and advanced `git` useage.

# man

- Man pages are the manuals installed with most applications on the command-line.

# man

- Man pages are the manuals installed with most applications on the command-line.
- To read a man page, just type `man` followed by the program name to launch it's manual.

# man

- Man pages are the manuals installed with most applications on the command-line.
- To read a man page, just type `man` followed by the program name to launch it's manual.
- Man pages exist for everything from trivial programs to language functions and libraries.

# Shell

- Shell is a programming language purpose built to patch together the output from applications.

# Shell

- Shell is a programming language purpose built to patch together the output from applications.
- It is a very lightweight language with only a few keywords.

# Shell

- Shell is a programming language purpose built to patch together the output from applications.
- It is a very lightweight language with only a few keywords.
- `man sh` should give you all the syntax.

# Shell

- Shell is a programming language purpose built to patch together the output from applications.
- It is a very lightweight language with only a few keywords.
- `man sh` should give you all the syntax.
- Commands wrapped with `$(COMMAND)` return the output of that command.

# Shell

- Shell is a programming language purpose built to patch together the output from applications.
- It is a very lightweight language with only a few keywords.
- `man sh` should give you all the syntax.
- Commands wrapped with `$(COMMAND)` return the output of that command.
- How would you get the total size of a directory? **du -b gives size in bytes.**

# Example Program

```
1  #!/bin/sh
2  set -e
3  TOTAL=0
4  for i in $(find .)
5  do
6   if [ ! -d $i ]
7   then
8    SIZE=$(du -b $i | awk '{print $1}')
9    TOTAL=$(echo $TOTAL + $SIZE | bc)
10  fi
11 done
12 echo "$TOTAL bytes"
```

This is also known as `du -s`.

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# Tips and Tricks

- xclip can put text into and out of the copy/paste buffer.
- Ctrl-a goes to front of line, Ctrl-e goes to end of line.
- Ctrl-u clears a whole line, Ctrl-k clears a line from the cursor onwards.
- Ctrl-w removes a word from the line.
- Invoking a command with a & afterwards will run it in the background.

# Outline


1. What Is The Command-Line?
2. The Command-line On Your Machine
   - Linux
   - Mac
   - Windows
3. Common Command-line Tools
   - Easy Tools
   - Archivers
4. Package Managers
   - Linux
   - Mac
   - All Machines
5. Programming On The Command-line
   - Editors
   - screen
   - Version Control
   - Man Pages
   - Shell
6. Tips and Tricks
7. QA

# QA time

Questions Anyone?