Coming Out of Your Shell

Kel Cecil
@praisechaos

bash

- Created by Brian Fox in 1989
- Stallman was not pleased with another developer's progress for an open shell

zsh

- Created by Paul Falstad in 1990 while a student at Princeton.
- Named after for Yale professor Zhong Shao whose login id was "zsh"

(Some) zsh Features

- Shared history
- Sweet loadable modules (like FTP and TCP support)
- Extended file globbing
- Theme-able prompts

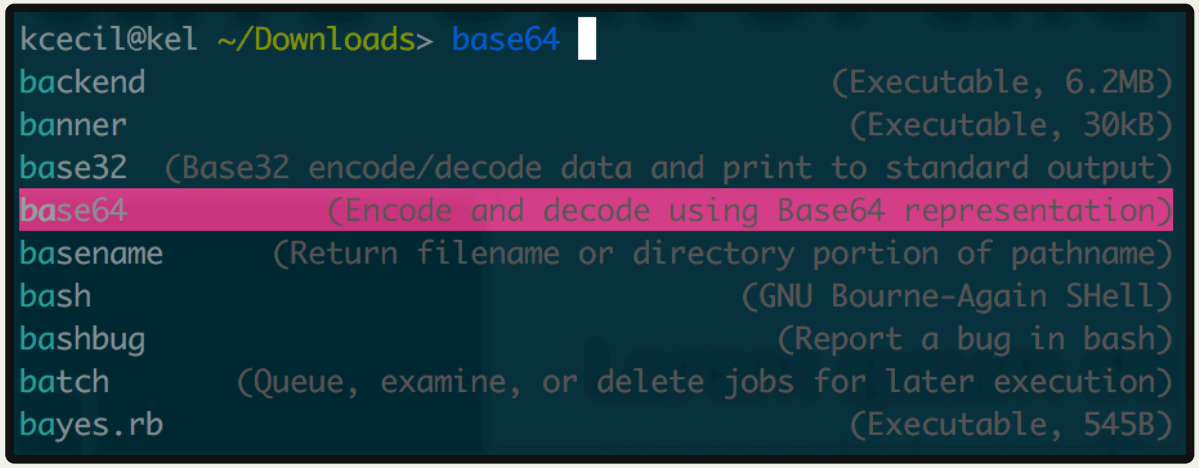
```
kelcecil@Kels-MBP:/Users/kelcecil  $ cd ~/code/thesis-java
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $ du -h -d 1
9.9M    ./.git
 10M    ./.gradle
788K    ./.idea
458M    ./build
1.1M    ./doc
 60K    ./gradle
388K    ./src
481M    .
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $ ls **/*.java(
md-30Lk+10)
src/main/java/org/kelcecil/thesis/data/common/DocumentSSDMatrix.java
src/main/java/org/kelcecil/thesis/exec/ButterflyIndexDocuments.java
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $
```

zsh configured with oh-my-zsh and demonstrating extended globbing

fish

- Orignally written by Axel Liljencrantz and released in 2005
- "friendly interactive shell"
- Considered an "exotic" shell
  - Does not derive syntax from Bourne or c shells
- fish's design is opinionated

(Some) fish Features



- Selectable, searchable tab-completion menu with helpful descriptions
- Aesthetic feedback on completions and more
- Shell completions by parsing man pages
- Web configuration tool (fish_config)
- Event handlers

Digging a Little Deeper

Globbing

Referencing Files with Ease

What is Globbing?

The ability to get a list of filenames matching some descriptive string

Proficiency with globbing can save a lot of time.

Recursive Wildcards

Works out of the box in zsh, fish

```
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $ ls
build           doc             gradlew         settings.gradle thesis-java.iml thesis-java.iws
build.gradle    gradle          gradlew.bat     src             thesis-java.ipr
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $ ls **/*Test.java
src/test/java/org/kelcecil/thesis/clustering/kmeans/KMeansTest.java
src/test/java/org/kelcecil/thesis/data/StopWordsTest.java
src/test/java/org/kelcecil/thesis/data/common/DocumentSSDMatrixTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorFilterTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorsTest.java
src/test/java/org/kelcecil/thesis/data/common/WeightMatrixTest.java
src/test/java/org/kelcecil/thesis/data/datasources/ReuterXMLDataSourceTest.java
src/test/java/org/kelcecil/thesis/data/kdf/KeywordDescriptorTest.java
src/test/java/org/kelcecil/thesis/data/ssd/SemanticSignatureTest.java
src/test/java/org/kelcecil/thesis/distance/impl/CosineDistanceTest.java
src/test/java/org/kelcecil/thesis/distance/impl/EuclideanDistanceTest.java
src/test/java/org/kelcecil/thesis/preprocessor/text/impl/SuffixStemmingPreprocessorTest.java
src/test/java/org/kelcecil/thesis/preprocessor/text/impl/SynonymPreprocessorTest.java
src/test/java/org/kelcecil/thesis/weight/LearnerWordWeightTest.java
src/test/java/org/kelcecil/thesis/weight/WindowWeightTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/AugmentedTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/BooleanTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/LogarithmicTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/TFIDFTest.java
kelcecil@Kels-MBP:/Users/kelcecil/code/thesis-java  git:(master) $
```

Recursive Wildcards

bash 4.x

```
bash-4.3$ cd ~/code/thesis-java/
bash-4.3$ ls **/*Test.java
ls: **/*Test.java: No such file or directory
bash-4.3$ shopt -s globstar
bash-4.3$ ls **/*Test.java
src/test/java/org/kelcecil/thesis/clustering/kmeans/KMeansTest.java
src/test/java/org/kelcecil/thesis/data/StopWordsTest.java
src/test/java/org/kelcecil/thesis/data/common/DocumentSSDMatrixTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorFilterTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorTest.java
src/test/java/org/kelcecil/thesis/data/common/VectorsTest.java
src/test/java/org/kelcecil/thesis/data/common/WeightMatrixTest.java
src/test/java/org/kelcecil/thesis/data/datasources/ReuterXMLDataSourceTest.java
src/test/java/org/kelcecil/thesis/data/kdf/KeywordDescriptorTest.java
src/test/java/org/kelcecil/thesis/data/ssd/SemanticSignatureTest.java
src/test/java/org/kelcecil/thesis/distance/impl/CosineDistanceTest.java
src/test/java/org/kelcecil/thesis/distance/impl/EuclideanDistanceTest.java
src/test/java/org/kelcecil/thesis/preprocessor/text/impl/SuffixStemmingPreprocessorTest.java
src/test/java/org/kelcecil/thesis/preprocessor/text/impl/SynonymPreprocessorTest.java
src/test/java/org/kelcecil/thesis/weight/LearnerWordWeightTest.java
src/test/java/org/kelcecil/thesis/weight/WindowWeightTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/AugmentedTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/BooleanTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/LogarithmicTFIDFTest.java
src/test/java/org/kelcecil/thesis/weight/tfidf/TFIDFTest.java
bash-4.3$
```

Opt-in Features for bash

- There are several opt-in features for bash.
- Opt-in is intended to preserve default behavior where desired.
- A list of opt-in features is available on the shopt builtin info page.
- Be sure to check it out if you're a bash person.

zsh's Glob Qualifiers

- What if I'd like to find an .iso in my Downloads folder?
  - Filter down to images that are larger than 1 GB
  - Modified less than 1 Month Ago

```
*.iso(.Lg+1mM-1)
```

```
kel~/Downloads % ls *.iso
Fedora-Live-Workstation-x86_64-21-5.iso   ubuntu-12.04.2-server-amd64.iso          ubuntu-14.04.3-server-amd64.iso
Fedora-Live-Workstation-x86_64-23-10.iso  ubuntu-12.10-server-amd64.iso           ubuntu-16.04-desktop-amd64.iso
Fedora-Server-DVD-x86_64-23.iso           ubuntu-14.04-beta2-desktop-amd64+mac.iso
kali-linux-1.1.0a-amd64.iso               ubuntu-14.04-server-amd64.iso
kel~/Downloads % ls -l *.iso(.Lg+1mM+1)
-rw-r--r-- 1 kcecil staff 1472200704 Dec  3  2014 Fedora-Live-Workstation-x86_64-21-5.iso
-rw-r--r-- 1 kcecil staff 1469054976 Oct 29  2015 Fedora-Live-Workstation-x86_64-23-10.iso
-rw-r--r-- 1 kcecil staff 2149580800 Oct 29  2015 Fedora-Server-DVD-x86_64-23.iso
-rw-r----- 1 kcecil staff 3063349248 Jul 28  2015 kali-linux-1.1.0a-amd64.iso
kel~/Downloads % ls -l *.iso(.Lg+1mM-1)
-rw-r--r-- 1 kcecil staff 1485881344 Jun  9 17:46 ubuntu-16.04-desktop-amd64.iso
kel~/Downloads %
```

zsh's estrings

- estrings allow you to use a comparison as a condition for expansion

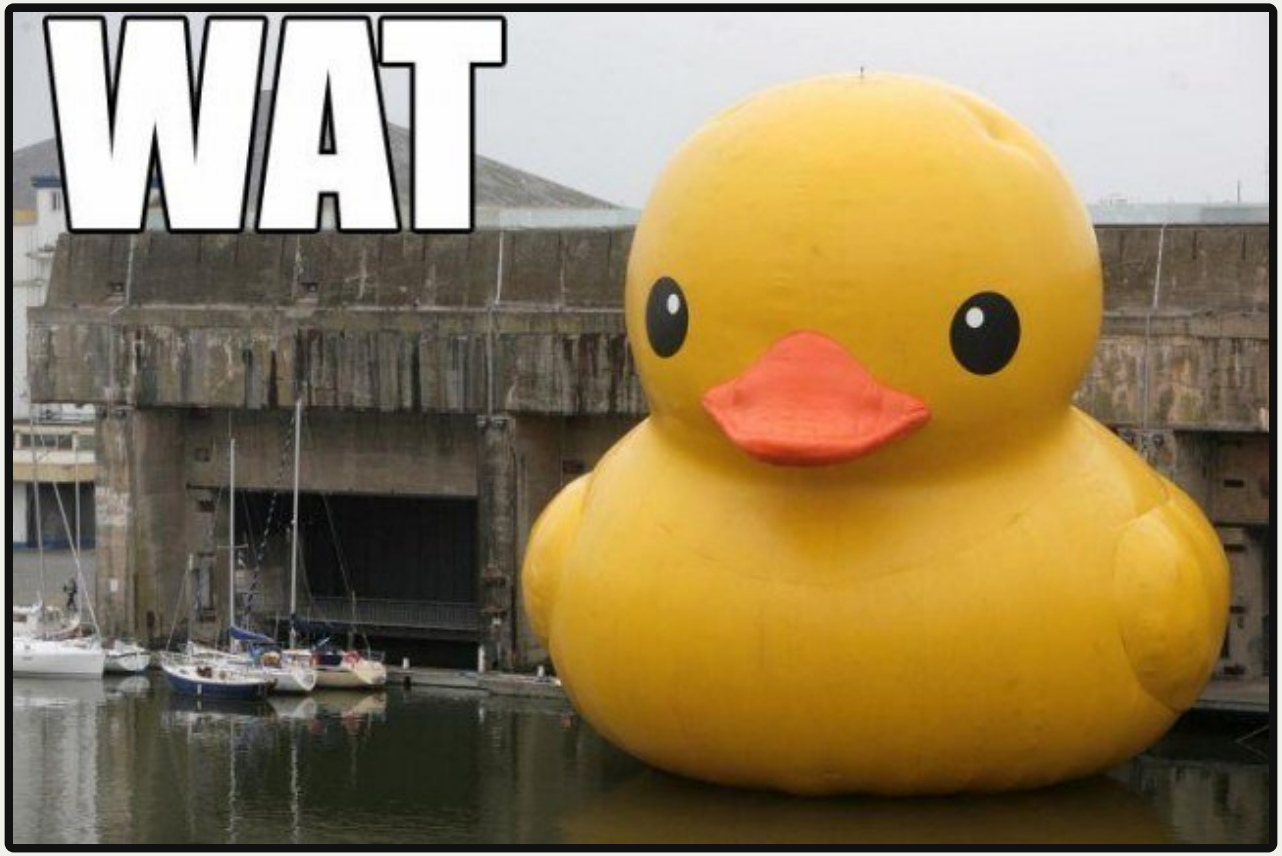
```
~/code/*(/e:'[[ -e ${REPLY}/.git ]]':)
```

```
kelcecil@Kels-MBP:/Users/kelcecil  $ print -l ~/code/*(/) | wc -l
      96
kelcecil@Kels-MBP:/Users/kelcecil  $ print -l ~/code/*(/e:'[[ -e ${REPLY}/.git ]]':) | wc -l
      53
```

zsh's glob qualifier abuse

Glob qualifiers can be nasty if abused.

```
~/code/*(-FUGmM+3e:'[[ -e ${REPLY}/.git ]]':^mM+12)
```


WAT

zsh's glob qualifier abuse

```
~/code/*(-FUGe:'[[ -e ${REPLY}/.git ]]':^mM+12)
```

- Non-empty directories (F) in the code directory in my home directory that are not symlinks (-)
- Directories are owned by the current user (U) AND the current user's primary group (G).
- Include the directory if it includes a .git directory (estring).
- Do not include (^) directories modified more than 12 months ago.

There's plenty more...

Be sure to check out zsh's Expansion manual page

Scripting

Making the shell work for you

A simple example script

- Takes a list of files and two words
- Loops through each of the files
  - Ensure the file is a regular file
  - Replace the first word with the second word
  - Output a status line

Example:

```
./chtext.sh bruceBanner incredibleHulk ./data/*.txt
```

bash

```
#!/bin/bash
# chtext - change text within multiple files

if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  echo >&2 "chtext: change ${FILE}: ${OLD} to ${NEW}"
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    else
      echo >&2 "chtext: could not change file: ${FILE}"
    fi
  fi
done
```

bash

- Check if we've supplied fewer than three parameters.
- Exit with an error code of 1 if so.

```
#!/bin/bash
# chtext - change text within multiple files

if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  echo >&2 "chtext: change ${FILE}: ${OLD} to ${NEW}"
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    else
      echo >&2 "chtext: could not change file: ${FILE}
    fi
  fi
done
```

bash

- Set our word to be replaced as OLD and the replacement word as NEW
- Shift the script arguments two places.
  - Move all array contents left two places.
  - This removes the first two array items.

```
#!/bin/bash
# chtext - change text within multiple files

if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  echo >&2 "chtext: change ${FILE}: ${OLD} to ${NEW}"
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    else
      echo >&2 "chtext: could not change file: ${FILE}
    fi
  fi
done
```

bash

- Loop through the remaining script arguments.
- Set each argument to the FILE variable.
- This isn't exactly intuitive...

```
#!/bin/bash
# chtext - change text within multiple files

if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  echo >&2 "chtext: change ${FILE}: ${OLD} to ${NEW}"
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    else
      echo >&2 "chtext: could not change file: ${FILE}
    fi
  fi
done
```

bash

- Check if the file is a regular file.
- Perform a sed to replace the OLD word with the new word and write to a temp file.
  - $$ represents the process ID for the script.
- If the sed returns a zero status code, then replace the file.

```
#!/bin/bash
# chtext - change text within multiple files

if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  echo >&2 "chtext: change ${FILE}: ${OLD} to ${NEW}"
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    else
      echo >&2 "chtext: could not change file: ${FILE}
    fi
  fi
done
```

fish is a little different...

- bash strives to be POSIX compliant
- zsh strives to be compatible with bash
- fish does it's own thing
  - fish is not POSIX compliant and considers that to be a feature.
  - The scripting language is intended to be more simple.

fish

```
#!/usr/bin/env fish
# chtext - change text within multiple files

set VALUES (count $argv)

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- Check to see if our arguments are less than 3.
- Script arguments are stored in $argv
- We use the math builtin for our comparison
- Redirect the math output to /dev/null to avoid printing our answer on stdout

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- fish uses "set" for variable assignment.

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- List ranges can be specified using ".."
- The last item in the list can be identified by using "-1"

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- Notice that variables do not have curly braces "{}"
  - Curly braces are optional in bash.
  - fish requires you not use them.
  - fish will be nice about it.

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- fish defers to programs when possible instead of reimplementing features.
- This call to test checks to see if a file is a regular file.

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

fish

- %self substitutes the process ID in fish.
  - Using $$ as you would in bash results in a helpful error message.

```
#!/usr/bin/env fish
# chtext - change text within multiple files

if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  echo >&2 "chtext: change $FILE: $OLD to $NEW"
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    else
      echo >&2 "chtext: could not change file: $FILE"
    end
  end
end
```

```
if [ $# -lt 3 ]; then
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
fi

OLD="$1"
NEW="$2"
shift 2

for FILE
do
  if [ -r "${FILE}" ]
  then
    if sed "s|${OLD}|${NEW}|g" < "${FILE}" > /tmp/ct$$
    then
      mv /tmp/ct$$ "${FILE}"
    fi
  fi
done
```

Which do ***you*** prefer?

Bash?

or fish?

```
if math (count $argv) "<3" > /dev/null
  echo >&2 "usage: chtext old new [file  ...]"
  exit 1
end

set OLD $argv[1]
set NEW $argv[2]

for FILE in $argv[3..-1]
  if test -f $FILE
    if sed "s|$OLD|$NEW|g" < "$FILE" > /tmp/ct%self
      mv /tmp/ct%self "$FILE"
    end
  end
end
```

Frameworks

Why use shell frameworks?

- Carefully manicured shell scripts can be a lot of work.
- Community maintained scripts can be higher quality.

Take Your Pick!

oh my zsh!

oh my fish!

bash it

What are my takeaways?

- bash is capable of more than it's given credit for.
- zsh is great for people who like to tweak and invest the time into unlocking it's full potential.
- fish is fantastic for useful features right out of the box.
- All three shells have frameworks to try.
- We've hardly scratched the surface...

Thanks for Listening!

Here's a Game Boy Advance running Unix 5!

Twitter: @praisechaos

Web: http://kelcecil.com