bash: JP's Choice Recipes

PLUG North
2008-05-12

PLUG West
2008-06-16

HMS
2008-07-02

JP Vossen
bashcookbook.com
2004-11-30 JP sends email to O'Reilly and proposes "How about a Bash cookbook?"
2005-03-24 Mike gets in touch
2005-04-01 Cameron (author of Learning the bash Shell) is added
2005-05-02 Submit second draft of ToC/outline
2005-09-13 Carl is added
2005-10-01 Start learning/writing in OR wiki
2005-10* Submit first proposal, various revisions, contract discussion
Book Timeline (2 / 4)

• 2005-10-20 Get contract (600 pages, 12% of net revenue, split 3 ways more or less)
• Schedule:
  – 2005-11-30 3 chapters
  – 2006-01-31 1/2 chapters
  – 2006-05-31 First draft
• 2006-02-13 Got first advance check
• 2006-07-13 Move to OpenOffice.org Writer and deployed echidna (for SVN)
• 2006-12-18 Submit real first draft (for technical reviewers)
2007-01-15 Technical Reviews due, give or take
2007-01-12 Got second/third advance check
2007-02-01 Submit copyediting draft
2007-02* Copyediting
2007-02-15 Got last advance check
2007-03-14 Draft from copyeditor to production
2007-03-27 Get PDFs for QC1
2007-03-29 Get FedEx of QC1 hardcopy (printouts of the QC1 PDFs)
Book Timeline (4 / 4)

- 2007-04-10 QC1 due back to O'Reilly
- 2007-04-25 Index (post QC1, pre QC2)
- 2007-05-03 QC2 review (PDF)
- 2007-05-11 Book goes to the printer!!

Recap:
- 2004-11-30 Email
- 2005-10-20 Get contract
- 2006-07-13 Move to OpenOffice.org
- 2007-05-11 To the printer
Why my prompt looks like this

- `PS1='\n[\u@\h:\l:$SHLVL:!\j:\D{%Y-%m-%d_%H:%M:%S_%Z}]\n$PWD\$ '

- Huh?!?
  - \n = newline
  - \u = user, \h = host
  - \l = Pseudo terminal number
  - $SHLVL = Shell level (nesting)
  - ! = Command history number
  - \j = # of background jobs
  - \D = strftime specification (bash ???+)
  - $PWD = Current working directory
  - \$ = # for root, $ for user

Using screen [17.4-6]

- **What?**
  - may need setuid root for tty manipulation
- **Why?**
- **How?**
  - **For training**
    - **Host**
      - `screen -S [session_name]`, e.g., `screen -S training`
      - `CTRL-a:multiuser on`
      - `CTRL-a:addacl [user,user]`, e.g., `addacl alice,bob,eve`
    - **Viewer**
      - `screen -x [user]/[session_name]`, e.g., `screen -x jp/training`
  - `CTRL-aK` to kill the window and end the session
How to show only dot files [1.5]

- What happens with `ls . *`?
  - `ls -d .`
  - `ls -d .b*`
  - `ls -d [.!]`

- Or construct your wildcard in such a way that . and .. don't match.
  - `$ grep -l PATH ~/.[!.]`
Are you running interactively? [1.8]

#!/usr/bin/env bash
# cookbook filename: interactive

case "$-" in
  *i*)  # Code for interactive shell here
    ;;
  *)    # Code for non-interactive shell here
    ;;
esac
Skipping a header [2.12]

- tail -n+2

$ cat data.file
Header line
Line 1
Line 2
Line 3
Line 4

$ tail -n+2 data.file
Line 1
Line 2
Line 3
Line 4
$ ./myscript 3>&1 1>stdout.logfile 2>&3- | tee -a stderr.logfile

3>&1 = FD3 gets FD1 (STDOUT)
2>&3- = FD2 (STDERR) gets 3, 3 closed

Why would you do this?
read -s -p "password: " PASSWD ; printf "%b" "\n"

- Why is that bad?
- Why all on one line?
- Why is the old stty method bad?
Error message on failure [4.8]

- `cmd || printf "%b" "cmd failed. You're on your own\n"

- `cmd || { printf "%b" "cmd failed. You're Toast!\n" exit 10 }

- `set -e`
  - Exit immediately if a command exits with a non-zero status
for SCRIPT in /path/to/scripts/dir/*
do
    if [ -f $SCRIPT -a -x $SCRIPT ]
    then
        $SCRIPT
    fi
done

- -f = is a file
- -a = logical ”and”
- -x = is executable
Embedded documentation [5.2]

- Use embedded POD (Perl's Plain Old Documentation)
- Seriously!

```bash
#!/usr/bin/env bash

echo 'Shell script code goes here'

# Use a : NOOP and here document to embed documentation,
#<"END_OF_DOCS"

Any accurate documentation is better than none at all.

=head1 NAME
[ POD here ]
=cut

END_OF_DOCS
```
Testing using a regex [6.8]

- if 

  \[
  \text{[[ } \text{CDTRACK} \text{ ]}=\sim\text{ "([[:alpha:][:blank:]]*)-([[:digit:]]*) -(.*$)\] ]}
  \]

- Also, sort-of

  case "$-" in
    *i*)  # Code for interactive shell here
    ;;
    * )  # Code for non-interactive shell here
    ;;
  esac
# cookbook filename: func_calc

# Trivial command line calculator
function calc {
    awk "BEGIN {print "The answer is: " $* }";
}

- How cool is that?
Sum a list of numbers [7.13]

- $ ls -l | awk '{sum += $5} END {print sum}'
- How cool is that, too? Especially the $5 part.
Sorting IP Addresses [8.3]

- $ sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n ipaddr.list
- `-t` = field separator
- `-k` = POSIX key definition
- `-n` = number

Watch out for LOCALE for sorting in general
Removing duplicates [8.5]

- cat data.file | sort -u
- sort -u data.file
- sort data.file | uniq -c | sort -nr
- sort data.file | uniq -c | sort -nr | head

- What's the difference?
- Watch out for uniq = uniq [input] [OUTPUT]
less is more [8.15]

- export LESS="--LONG-PROMPT --LINE-NUMBERS --ignore-case --QUIET"
- Debian has a /usr/bin/lesspipe that can be eval'ed and also supports additional filters via a ~/.lessfilter file.
- Use Wolfgang Friebel's lesspipe.sh
Handling odd characters [9.2]

- Shell quoting
  - `$ find . -name '*.mp3' -print0 | xargs -i -0 mv '{}' ~/songs`

- `find -print0` = use NULL instead of white space for field separator
- `xargs -0` = use NULL instead of white space for field separator
Dates and times [ch 11]

- `date '+%Y-%m-%d'`
- `date '+%Y-%m-%d_%H:%M:%S %Z'`
- `date -d '+ 90 minutes' '+%R'`
- `date '+%s'`
- `date -d '2005-11-05 12:00:00 +0000' '+%s'`
- `date -d "1970-01-01 UTC $EPOCH seconds" +"%Y-%m-%d %T %z"`
- `date --utc --date "1970-01-01 $EPOCH seconds" +"%Y-%m-%d %T %z"`
Handling white space [13.14,15]

- `$ while read REPLY; do echo ~~"$REPLY"~~; done < whitespace`
- `$ while read; do echo "~~${REPLY## }~~"; done < whitespace`
- `$ while read; do echo "~~${REPLY%% }~~"; done < whitespace`
- `$ cat data_file | tr -s ' ' '	'`
- `$ awk 'BEGIN { FS = " "; OFS = "\t" } { $1 = $1; gsub(/\t+/, "\t"); print }' data_file1`
Fixed length or no line breaks [13.16,17]

- $ gawk ' BEGIN { FIELDWIDTHS = "18 32 16"; OFS = "\t" } { $1 = $1; gsub(/ +\t/, "\t"); gsub(/ +$/, "" ); print }' fixed-length_file

- $ perl -ne 'print join("\t", unpack("A18 A32 A16", $_) ) . "\n"; ' fixed-length_file
$RANDOM [14.11]

- This is one of my favorite bash features!
  - bash 2.0+
  - not in dash (buggers)

- echo $RANDOM$RANDOM$RANDOM

- How do you create secure temp files?
Script portability [15.1,3,4]

- #!/usr/bin/env bash
- Why not #!/usr/bin/env bash -

- Don't use Linux

- http://www.pcbsd.org/?p=download#vmware
until [ "phase" = "Finished." ]; do

    case $phase in

        phase0 )

            ...
        

    esac

    phase="Finished."

    ;;

    printf "%b" \a # Ring the bell

done
Tweaking your Environment

[16.7,8]

- set and shopt
  - `shopt -q -s cdspell`
  - `shopt -q -s checkwinsize`
  - `set -o notify` # (or set `-b`)
  - `set -o ignoreeof`

- `~/.inputrc` or `/etc/inputrc`
  - `set completion-ignore-case on`
  - `"\C-i": menu-complete`
  - `set show-all-if-ambiguous on`
• mcd () { mkdir "$1" && cd "$1"; }
• OR
  # mkdir newdir then cd into it
  # usage: mcd (<mode>) <dir>
  function mcd {
    local newdir=''_mcd_command_failed_'
    if [ -d "$1" ]; then         # Dir exists, mention that...
      echo "$1 exists..."
      newdir="$1"
    else
      if [ -n "$2" ]; then     # We've specified a mode
        command mkdir -p -m $1 "$2" && newdir="$2"
      else                    # Plain old mkdir
        command mkdir -p "$1" && newdir="$1"
      fi
    fi
    builtin cd "$newdir"      # No matter what, cd into it
  } # end of mcd
bot [16.15]

- Getting to the bottom of things
- alias bot='cd $(dirname $(find . | tail -1))'
Programmable Completion

[16.17]

- Ian Macdonald's set from http://freshmeat.net/projects/bashcompletion/

- Already included in Debian and Ubuntu, but not turned on by default. See "# enable bash completion in interactive shells" in /etc/bash.bashrc. and /etc/bash_completion.
grep ps without grep [17.18]

- pgrep -l ssh
- `ps auwwx | grep '[s]sh'`

But why does this work?

Old, ugly:
- `ps auwwx | grep 'ssh' | grep -v grep`
Adding a per-line prefix or postfix [17.20]

- $ last | while read i; do [[ -n "$i" ]]] && printf "%b" "$HOSTNAME\t"$i
  \n; done
- $ last | grep -v '^$'| while read i; do printf "%b" "$i\t$HOSTNAME\n"; done
- $ last | awk "BEGIN { OFS=\"\t\" } ! /^\$/ \n{ print "$HOSTNAME\", \$0}"$ last | perl -ne "print qq($HOSTNAME\t$_) if ! /^\s*$/;"
- $ last | sed "s/./$HOSTNAME\&/; /^$/d"
Numbering Lines [17.21]

- $ i=0; while IFS= read -r line; do (( i++ ));
  echo "$i $line"; done < lines
- $ cat -n lines
- $ less -N lines
- $ nl lines
- $ nl -ba lines
- $ awk '{ print NR, $0 }' filename
- $ perl -ne 'print qq($.	$_);' filename
Writing sequences [17.22]

- awk should always work
  - `$ awk 'BEGIN { for (i=1; i <= 5; i+=.5) print i}' /dev/null`
- Bash 2.04+ only, integer only
  - `$ for ((i=1; i<=5; i++)); do echo "$i text"; done`
- Bash 3.0+ only, integer or single character only
  - `$ printf "%s text\n" {1..5}`
  - `$ printf "%s text\n" {a..e}`
- man seq
Debugging [19.12]

- bash -n
  - Like perl -c, check basic syntax, but don't run

- set -x
  - debugging; show the final parsed command

- set -v
  - verbose; show the raw unparsed command
Questions?

- http://examples.oreilly.com/bashckbk/
- [N.N] = Recipe numbers
- bashcookbook.com
- PLUG Mailing list
- jp@jpsdomain.org