Shell Scripting Craftsmanship

Ray Smith
Database Administrator
Portland General Electric
Objectives

- Philosophical alignment
  - Context of quality

- Define quality in scripted solutions
  - What are the common elements of good scripts?

- 40 slides in 45 minutes
  - Available online after the conference
Quality cabinets

- Look at the drawers
  - Hardware
    - Attractive, inviting, practical
    - Prevent accidents (limited range)
  - Technique
    - Dovetail, staples, gum

- Expectations
  - Good enough for what it is
  - Something built to last
Quality

- **Rule #1:** SHELL SCRIPTS MUST WORK
  - High-visibility tasks
    - Migrations, deployments, and upgrades
  - Unforgiving tasks
    - Backups and monitoring
  - Repetitive tasks
    - Reporting and analysis

- **Rule #2:** SCRIPTS MUST KEEP WORKING
  - Harmony and joy or . . .
Spiritual Guidance

- *The Art of Unix Programming*
  - Eric S. Raymond
  - Addison-Wesley

- Three decades of unwritten, hard-won software engineering wisdom from 13 Unix pioneers

- Resident anthropologist and roving ambassador of the open source movement
Quality, craftsmanship, harmony

- Dimensions of shell script quality
  - Transparency
  - Clear communication
  - Scalability
Transparency

- **Rule of Clarity**
  - Clarity is better than cleverness

- **Rule of Simplicity**
  - Design for simplicity
  - Add complexity only where you must

- **Rule of Transparency**
  - Design for visibility, inspection, and debugging
Clear Explanations

- Be generous with internal documentation
  - Particularly when being clever or ultra-efficient

```bash
# Find all instances of the string
find . -type f -exec fgrep -i "$mySTRING" \ /tmp/dummy {{}} \; 2>/dev/null
```

- Avoid NVA interpretations

```bash
# List the directory content, including links
ls -laSf
```
Visual Simplicity

- Be kind to yourself and others
- Layouts and formatting

USE WHITESPACE

Break up long lines with \\
back-slash \\

- It is okay to use the TAB key
for thisHOST in `cat ${HOSTLIST}`; do
    if [ ${#thisHOST} -gt 5 ]; then
        echo "BIG: ${thisHOST} is ${#thisHOST} characters"
    else
        if [ ${#thisHOST} -lt 3 ]; then
            echo "LITTLE: ${thisHOST} name is short"
        fi
    fi
done
```
for thisHOST in `cat ${HOSTLIST}`; do
  if [ ${#thisHOST} -gt 5 ]; then
    echo "BIG: ${thisHOST} is ${#thisHOST} characters"
  else
    if [ ${#thisHOST} -lt 3 ]; then
      echo "LITTLE: ${thisHOST} is ${#thisHOST} characters"
    fi
  fi
done

for thisHOST in `cat ${HOSTLIST}`; do
  if [ ${#thisHOST} -gt 5 ]; then
    echo "BIG: ${thisHOST} name is long"
  else
    if [ ${#thisHOST} -lt 3 ]; then
      echo "LITTLE: ${thisHOST} name is short"
    fi
  fi
done
```
Harmonious style and content

This works:

```bash
mySID=`zenity --list --text "Select the database instance" --column "SID" --column "Description" "NICKEL" "Five Cent Database" "URANIUM" "Not-For-Export Database" "CUSTOM" "User defined instance"
```

So does this:

```bash
mySID=`zenity --list \
--text "Select the database instance" \\n--column "SID" --column "Description" \
"NICKEL" "Five Cent Database" \
"URANIUM" "Not-For-Export Database" \
"CUSTOM" "User defined instance"
```

Which would you rather update?
Visual Consistency

- Make \${VARIABLES} stand out in your code

- Variable naming conventions
  - **ALL_CAPS** for variable names
  - **CamelCase** for function names
  - **thisVARIABLE** or **myVARIABLE** for looped variables

- Be internally consistent
  - Make a rule and follow it
for thishost in `cat $hostlist`; do
    if [ $#thishost -gt 5 ]; then
        longmessage
    else
        if [ $#thishost -lt 3 ]; then
            shortmessage
        fi
    fi
done
Outstanding Variables

for thishost in `cat $hostlist`; do
  if [ $#thishost -gt 5 ]; then
    longmessage
  else
    if [ $#thishost -lt 3 ]; then
      shortmessage
    fi
  fi
done

for thisHOST in `cat ${HOSTLIST}`; do
  if [ ${#thisHOST} -gt 5 ]; then
    LongMessage
  else
    if [ ${#thisHOST} -lt 3 ]; then
      ShortMessage
    fi
  fi
done
The Penny Wise Quiz

a. Shorter variable names =
   Less typing =
   Less work

b. Obscure variable names =
   Reduced transparency =
   Poor quality

• Save your cycles for decyphering the logic
  • Not the variable names
Transparency II

- Rule of **Modularity**
  - Write simple parts connected by clean interfaces

- Rule of **Robustness**
  - Robustness is the child of transparency and simplicity
Efficiency

- Use shell script **functions**
  - Simple syntax

```bash
function GiveDirectoryContent {
    echo "This directory contains:"
    ls -lasF
}

> GiveDirectoryContent
```

- Modularize *all* repeated code statements
Predictability

- Layout in sections
  - **Header** with file name, purpose, command-line inputs
  - **Independent variables** – edited w/ every installation
    - “Nothing to change below this line”
  - **Dependent variables** – never require edits
  - **Functions** section
  - **Runtime** block

- Think “On call” when laying out a script
#!/bin/bash

# Independent variables
export BASEDIR=/usr/lbin/orascripts

# Dependent variables
LOGDIR=$BASEDIR/logs
WORKFILE=$LOGDIR/workfile.lst
Transparency Zen

• Rule of Diversity
  • Distrust all claims of “one true way”
  • Including your own

• Collaboration + humility = the cost of quality
  • Design reviews expose opportunities
  • Code reviews expose failure modes

Steal, adapt, improve, repeat
If it ain't broke . . .

- Don't ignore it
  - Particularly for older scripts

- Tune it up
  - Apply new techniques and tools
    - Gzip for compress

- Verify old programs' assumptions
  - Unreliable hardware
Transparency Wrap-up

Well-crafted shell scripts:

- Written with maintenance and on-call in mind
- Provide useful guidance and instructions
- Reflect visual simplicity and clear layout

Comments or questions
User Communication

- **Rule of Silence**
  - Communicate clearly when necessary

- **Rule of Repair**
  - When you must fail, fail noisily and as soon as possible

- **Rule of Least Surprise**
  - In interface design, do the least surprising thing
  - Do the most expected thing
Work with the user

- Verify scripted actions
  ```
  echo "These files will be backed up:"
  cat ${FILELIST}
  ```

- Keep the user informed
  - Share status and decisions
    ```
    echo "You asked to delete everything in /etc"
    read -p "Is this correct? [Y|N]" myVal
    case $myVal in
      [Y]
        echo "Proceeding with deletion"
      [N]
        echo "Stopping deletion"
    esac
    ```
Graciousness

- Handle errors with grace
  - Explain the situation
  - Lead the user to a solution

```bash
if [ $# -eq 0 ]; then
echo "The required value for database name"
echo "was not passed in the command line"
read -p "Enter the database name: " myVal
export thisSID=$myVal
fi
```
Signs of Life

- Same script should work in cron or interactive
  - Test for tty (terminal id)

```bash
if tty -s
then
    echo "Oh good, this is interactive"
    echo "Hello $USER"
else
    date +"Script $0 started %T" >> $LOGFILE
fi
```
Communicating Failure

- Cryptic feedback is neither welcome nor helpful
  - FATAL ERROR: bckpinit_strt error 12
  - File not found: bckpinit_strt.conf
  - Corrective action required:
    - Verify bckpinit_strt.conf exists at ABC
    - and readable by user xyz
  - Email notification has been sent

- Terminal output is free, use it if you need it
Artist and Psychologist

- Break the output into usable blocks
  - Use \n and \t liberally
    - If it makes it easier for user to understand

- Particularly important for interactive scripts
  - Push the read statement into their attention

- Direct their eyes with breaks and groups
  - Apply the same practices to log files and reports
Electronic Communication

- Be complete, be clear
  - Which host
  - Which process / job
  - What happened (in prose)
  - How to troubleshoot / resolve the problem

- Start communicating in the subject line

- Cryptic feedback in email is useless
Relevance

Subject: servtest1: oradb arch_move completed WITH WARNINGS

servtest1: oradb arch_move completed WITH WARNINGS on 07/21/08
Total number of warnings = 1.
Please review warning log,
/oradb_orabase/local/logs/oradb.arch_move.wlog

***** BOF /oradb_orabase/local/logs/oradb.arch_move.wlog *****
13:03:47 STG 3:Warning : NO Archives to move
***** EOF /oradb_orabase/local/logs/oradb.arch_move.wlog *****
Communication Wrap-up

Well-crafted shell scripts:

- Work *with* the user
- Are generous with visual guidance and produce attractive, useful log files and reports
- Provide clear and complete feedback

Comments or questions
Scalability

- Rule of Extensibility
  - Design for the future
  - It will be here before you know it

- Rule of Economy
  - Use machine resources instead of people power

- Mindset
  - Awareness
  - Thoughtfulness
Scalability

- Scalability goal #1: Never customize a script

- Hard and fast rules
  - Overuse variables
  - Never hardcode
    - Passwords
    - Host or database names
    - Paths
  - Use command-line input, ‘read’, or parameter files
Stability and Predictability

- Consistency
  - Use the same code across your entire enterprise

- Security
  - Limit editing permissions
    - 'Them' and you, too

- Revision control
  - Keep a gold copy out there
Make the Machine do the Work

- Create everything you need, every time
  - Fix permissions too

```bash
export thisDIR=/usr/lib/bin/orascripts

if [ ! -d $thisDIR ]; then
  mkdir $thisDIR
fi
chmod 775 $thisDIR
```

- If you would manually check it . . .
Resourcefulness

- Single-point maintenance
  - Host and database name lists
  - Central repository

- Use existing files
  - /etc/passwd, var/opt/oracle/oratab

- Use a function library
  - 'Sourced' whenever needed
  - Edit once for every use
Function library contents

- Suggestions
  - Report headings
  - Common format inserts
  - Email distribution lists or policies
  - Email privacy notices
  - Error handling

- Watch for opportunities
Scalability Wrap-up

Well-crafted shell scripts:

- Never require editing for a new host / installation

- Handle expected problems from the start

- Use existing resources whenever possible

Comments or questions
You are a Unix Programmer

“To do the Unix philosophy right, you have to be loyal to excellence. You have to believe that software design is a craft worth all the intelligence, creativity, and passion you can muster.”

-- Eric S. Raymond
Powells World of Books

- World’s largest independent bookstore
- Dedicated technical bookstore
- Short train / street car ride from Convention Center
  - All Free Fare zone
  - Any west-bound MAX
  - Transfer to Portland Streetcar at 10th Avenue
  - Half-mile north

www.powells.com