bash

Monday, March 23, 2015 8:29 AM

Objectives:

- Describe purpose of bash shell
- bash shell startup files
- Using control structures
- Create shell variables
- Understand shell history

Introduction

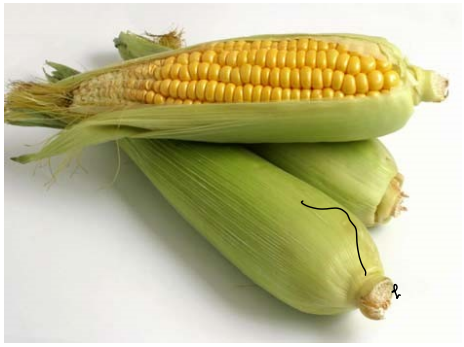
- What is a Kenel?
- What is a Shell?
- Shell history
  - → Ken Thompson creates original system shell (sh)
  - 1977 Unix replaced (sh) with the Bourne Shell
  -

- No.... not that Bourne,
- Steve Bourne of Bell Labs

- Bourne shell is standard on all unix ~~Linux~~ systems
  - sh added
    - Control flow (while)
    - Environment variables
    - Signal handling (what happens when I hit ctrl+c?)

- At same time Bill Joy (1978)
  - vi Bill Joy
  - Created C shell (csh or tcsh)

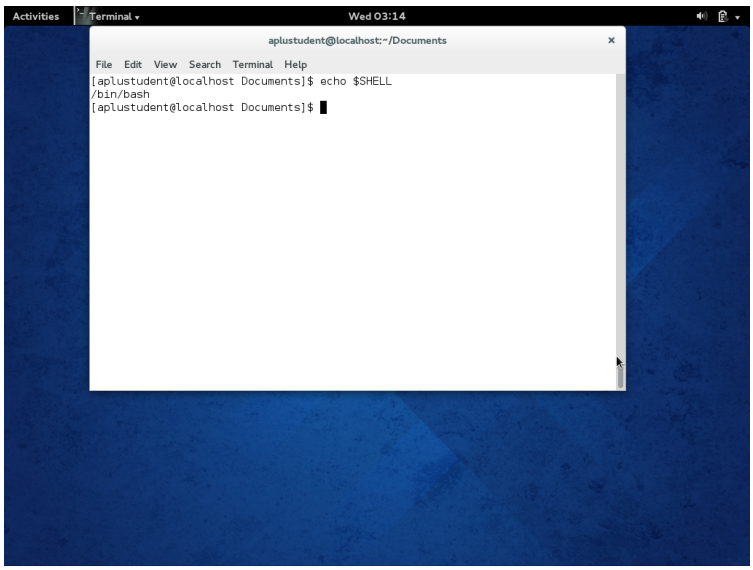
- C shell was improved over sh (bourne)
  - History
  - Aliases
  - tilde notation
  - It used the "C" language syntax
  - Which made sense? Why?
- 1980's → Bell labs David Korn → Korn shell (ksh)



- Basically combined (CSH + SH) into KSH
- Since this was Bell Lab's At&t property
  - remained proprietary until 2000.
  - All shells ksh, sh, csh were owned and not free and open
  - Not a problem when UNIX was only available on mainframes

- but (RMS)... had an issue

- BASH (Bourne Again Shell)
  - Play on words
  - RMS disdain for religion not withstanding
  - 1989 Brian Fox of the GNU project
  - Basically "bashed" together sh + csh + ksh
  - But opened source.... GPL
  - Linux adopted BASH as its default shell
- That is what you are using now



Bash start up files

- When your system first boots it processes various "default" shell variables
- When a new terminal launches, various shell variables are processed.

/etc/profile

- Your shell always executes this file upon boot

- Let's see what is in it
- Less /etc/profile
- What do you see?
- This file handles settings for ALL users (systemwide)

Next...

~/.bash_profile or ~/.bash_login or ~/.profile

- Note the "." what does it mean?
- Note the "~" what does it mean?
- Where are these files located?
- Lets check, ls -la ~, then less each file (if it exists) Compare Ubuntu and FreeBSD

Exit

~/.bash_logout

Non login (when you open a new terminal window or tab)

~/.bashrc

Each users ~/.bashrc is built from /etc/bashrc

PATH

- PATH additions should be added to .bash_profile
- Lets try it
  - First take Virtualbox snapshot ... just in case
    - ↳ Point in time "backup" under "Machine"
  - echo $PATH
  - vi ~/.bash_profile
  - remove the portion of the PATH that says ":$HOME/bin"

- logout user and log back in

https://ask.fedoraproject.org/en/question/30783/how-do-i-enable-the-logout-option-in-gnome-shell/

(GNOME3 disables logout by default)

- Reinitialize without logout or reboot
  - just use a "."

. ~/.bashrc

Writing and Executing a shell Script

- Let's try it
- vi whoson

```
date
echo "Users Currently Logged in"
who
```

- Now try ./whoson
  - What happens?
  - Why?
  - how to fix?
  - Each shell script needs "+x"
  - Be careful who you give "x" to
  - In your "ls" output scripts marked executable will appear green.

#! → Shell specification

- In a shell script it will be executed by the value of $SHELL or CSH or KSH

You can overrride this by placing #!/bin/bash (no spaces)
on the first line of your shell script

- Scripts don't have to always be shell scripts
  - Perl, Python, Ruby, even C++!

- Try it

```
perl_script.pl

#!/usr/bin/perl -w
print "This is perl.\n";
print reverse <>;

./perl_script.pl
```

- Otherwise # is a line comment in BASH, KSH, SH

~~Control~~ Operators (Magic)

; seperator

- allows you to "chain" commands

```
date;ls;date
```

Note Lemmings mode

- command will execute regardless of previous command
  - each

- "&&" "||" (test succeed ~~failure~~)

- This let's you chain commands similar to ";"
  - && checks for for command success
    - then executes following command
  - || executes following command based on failure

"\" is a continuation

Line wrap

Parameters and Values

You can create shell variable -
- They only "live" for the length of the shell
-
- So if you exit - those variables disappear
Cannot share variables accross shells
- Can't start with a number or include punctuation

- Assigning
Use no $ sign

VAR=myvalue

Note by convention variable is all caps
space in assigning
Also no

- Access

echo $VAR
-
- use $ when referencing the variable contents

- Unset
- will remove content of variable

Variable types
- a make variable an array
- i integer (never really want to do this)
- r read only
- x makes it a system wide environment variable

ocesses

- Every process started (ps -ef) descends from (PID) 1
- Hence every process is a child of PID 1
  - Note on ps -ef each process has a PID and PPID
    - Process Parent ID
    - Process ID
    - All but PID 1 (Like Adam and Eve I guess)
- Parent Processes fork call
  - fork is a system call that copies a parent but is seperate
    - Think children to real parents.
    - Whenever you execute a command (ls, cat, ps...)
    - A new shell is "forked" and the command runs in that shell.
      Shell exists and value is returned to parent.
- In Redhat / Ubuntu / Debian / Opensuse Linux
  - PID is systemd
  - An executable that contains a bunch of processes to start
  - Violates the "UNIX" philosophy of "Do 1 thing well"
  - Replaced SysVInit
  - Free and
  - Load up BSD do ps -ef
    This has lead to a linux Civil war
    - Debian has split into two camps
      Devuan

- (Italian) is anti-systemd distribution
They are forking the Debian
- major undertaking
- Run pstree -p to see it "systemd" "claws"
- and what processes built in commands
Shell does not fork
- (cd, pwd, jobs etc etc)

---

story
As you know you execute commands you accumulate history
Just system to command line and hit "up" arrow
- variables to control how much history you keep
$HISTSIZE
~/.bash_history location of history file
history | tail (see last 10 commands executed)

create a alias:
alias command 'h=history | tail' to

fc allows you "re execute" a series of commands from your history.

!! on bash will execute previous command

!n executes event in history

!? letter? can use shell metacharaters to search history

history 10 will execute what?

- Command Completion

[TAB] try it !

---

ummary of Chapter 9

Shell
- Executes is both intepreter and a programming language
  - scripts
  - Scripts must be executable
  - BASH is the default Linux shell
  - SHELLs handle variables, processess, history

Chapter 27 - Programming Bash

Objectives:
- Use Control Structures (If else, while, for)
- in shell scripts
- Handle input ↔ ouput
- Apply the use of shell variables
- Understand the script best practices

Note - don't name a shell script test - there is a bash tool called test

---

~~ontrol~~ Structures

Most scripts or programs for that matter need a "control"
- Decision logic

If statement
- Similar to C and Java, but syntax is different!

if test-command
then

fi commands

note use of "fi" instead of { } ; or end

---

previous condition check

- Done with [con]
-
- watch now Bash contains test
- But out test is a "bashism" may not work in other shells
-

```
$ cat chkargs
if test $# -eq 0
  then
    echo "You must supply at least one argument."
    exit 1
fi
echo "Program running."
$ ./chkargs
You must supply at least one argument.
$ ./chkargs abc
Program running.
```

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

- Remember the $ is not typed its a visual indicator of a prompt of a Command

- For a shell script there are positional argument identifiers $1 $2
- ls -la /etc or (bash) ./list /etc

cat ./list
ls -la $1 ← Same

$variables a BASH variable

Values are accessed via $1, $2, $3, etc, etc

asn

also includes "pre-made" conditional tests

Criterion	Tests file to see if it
–d	Exists and is a directory file
–e	Exists
–f	Exists and is an ordinary file (not a directory)
–r	Exists and is readable
–s	Exists and has a size greater than 0 bytes
–w	Exists and is writable
–x	Exists and is executable

:se

our options allow you to write parameter checks so shell script doesn't do anything illegal or generate errors.

t **chkargs2**

-eq 0]

en

echo "Usage: chkargs2 argument..." 1>&2

exit 1

› "Program running."

0

hkargs2

ge: chkargs2 argument...

hkargs2 abc

ram running.

<http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?entage=0.0&reader=html>

Note: different use of implied test

Note the 1>&2 what is that doing? Redirecting output to stdout and stderr!

If

else

If else elif

ultiple tiered decision structures

use -x on a script for verbose debugging

in

p-index** in **argument-list

ands

p://my.safaribooksonline.com/9780133477443/ch09lev1sec19?
e=0.0&reader=html>

ke for loop
• Incrementor already provided

*for **loop-index***
do
commands
done

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

• has which can implied for arg which will handle all conditionals
is different from C/Java

ile

*while **test-command***
do
commands
done

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

• Built in for use with test and comparisons

script uses test with the **–lt** argument to perform a numerical test. For nerical comparisons, you must use **–ne** (not equal), **–eq** (equal), **–gt** (greater n), **–ge** (greater than or equal to), **–lt** (less than), or **–le** (less than or equal . For string comparisons, use **=** (equal) or **!=** (not equal) when you are working h test.

<http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

il

*il **test-command***

mmands
e

<http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?
entage=0.0&reader=html>

• Note the trap command "catches" any cancel or kill commands so this script process can't be bypassed (you can still kill it with a kill -9)

```
ktty
ash

? 3 18
o
Key: " key_1

Again: " key_2

_1" = "$key_2" ]

clear
```

```
clear

[ "$key_3" = "$key_2" ]


ad key_3

e

) "locktty: keys do not match" 1>&2


p://my.safaribooksonline.com/9780133477443/ch09lev1sec19?
e=0.0&reader=html>
```

Se

```
e test-string in
tern-1)
        commands-1
        ;;
tern-2)
        commands-2
        ;; pattern-3)
        commands-3
        ;;

        esac
```

<http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?
entage=0.0&reader=html>

```
$ cat case1
read -p "Enter A, B, or C: " letter
case "$letter" in
  A)
    echo "You entered A"
    ;;
  B)
    echo "You entered B"
    ;;
  C)
    echo "You entered C"
    ;;
  *)
    echo "You did not enter A, B, or C"
    ;;
esac
```

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?
percentage=0.0&reader=html>

- Note until executes in a loop until you get the correct password
- Use case if you find have an unwieldly if-else tree.
- Note esac (case backwards) ends the case statement
- Different from C/Java can be a|A)
- Left of) is the case to match
- Meta characters can be used
- Even [...] ranges can be placed.

is the catch all

= number of positional parameters

@ = array of all the parameters (useful for forloop)

$$ = prints PID of process

$? = prints exit status (if BASH command succesfully executed)

$ = tells you which set

= optionally you can the (Cr i x ...) command as well

– Last argument use of previous command

```
$ cat last_arg
echo $_
echo here I am
echo $_

$ ./last_arg
./last_arg
here I am
am
```

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

---

ort

variables are always local

Shell

Export command makes them available to the system environment

ENV out

ronment prints what the system has exported

read

accepts user input

```
$ cat read1
echo -n "Go ahead: "
read firstline
echo "You entered: $firstline"

$ ./read1
Go ahead:
    This is a line.
You entered: This is a line.
```

From <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

LY

en you do not specify a variable to receive read's input, bash puts the input into variable named **REPLY**. The following **read1b** script performs the same task as **ıd1**:

```
t read1b
 -p "Go ahead: "
› "You entered: $REPLY"
```

ı <http://my.safaribooksonline.com/9780133477443/ch09lev1sec19?percentage=0.0&reader=html>

Questions?