

Chapter 8

The Bourne Again Shell

Graham Glass and King Ables,
UNIX for Programmers and Users,
Third Edition, Pearson Prentice Hall, 2003.

Original Notes by Michael Weeks

Bourne Again SHell (bash)

- **Typical shell on Linux systems**
- **Written by Brian Fox**
 - **of the Free Software Foundation**
- **Backwards Compatible with Bourne Shell**
 - **Bourne Shell info applies here, too**
- **Runs /etc/profile first, then:**
 - **At login, runs ~/.bash_profile**
 - **At startup, runs ~/.bashrc**

Variables

- Setting and accessing variables
 - same as before
 - For example
 - numbers="one two"
 - echo \$numbers

Arrays

- Defining arrays
 - declare -a teams
 - teams[0]="Hawks"
 - teams[1]="Falcons"
- Must use { } to access
 - \$ echo The \${teams[1]} play football.
 - The Falcons play football.

Accessing Arrays

- Access all entries with [*] / [@]
- Find the number of entries with {#array[*]} / {#array[@]}

```
$ echo ${teams[*]}  
Hawks Falcons
```

```
$ echo I know of ${#teams[*]} teams.  
I know of 2 teams.
```

Building Lists

- Options

- Assign entries with number
- Give list in parenthesis
 - Use \ to continue a line, as needed

```
$ declare -a list1
$ list1[0]="Thrashers"
$ list1[1]="Braves"
$ list1[2]="Hawks"
$ list1[3]=Falcons
$ echo ${list1[*]}
Thrashers Braves Hawks
Falcons
```

```
$ declare -a list2
$ list2=("Thrashers" "Braves" \
> "Hawks" Falcons)
$ echo ${list2[*]}
Thrashers Braves Hawks Falcons
```

Deleting Lists

- Command *unset*
- Delete an entry or entire list

```
$ echo ${list2[*]}  
Thrashers Braves Hawks Falcons
```

```
$ unset list2[1]  
$ echo ${list2[*]}  
Thrashers Hawks Falcons
```

```
$ unset list2  
$ echo ${list2[*]}
```

```
$
```

Aliases

- Command *alias* defines commands
- That is, put these in *.bashrc*

```
alias qubit="ssh -l myacct qubit.cs.gsu.edu"  
alias squbit="sftp myacct@qubit.cs.gsu.edu"
```

```
$ alias qubit="ssh -l cscmcw qubit.cs.gsu.edu"
```

```
$ alias
```

```
alias ls='ls --color=auto'
```

```
alias qubit='ssh -l cscmcw qubit.cs.gsu.edu'
```

```
$ qubit
```

```
cscmcw@qubit.cs.gsu.edu's password:
```

```
Last login: Tue Oct 16 17:43:26 2007 from carmaux
```

```
-bash-3.00$
```

History

- Bash keeps track of the commands you type
 - So do other shells, e.g. Korn, Csh
- `$HISTSIZE` defines # commands to remember
- Command *history* shows list

History Shortcuts

- !! repeats last command
- !34 repeats command #34
- !gre repeats the last command starting with gre
 - Such as grep ...
- ^str1^str2
 - repeats last command, substituting str1 for str2

```
$ mroe myfile
```

```
-bash: mroe: command not found
```

```
$ ^ro^or
```

```
more myfile
```

```
!!!
```

Bash Arithmetic

- Instead of *expr*, use `((operation))`
 - `+`, `-`, `++`, `--`, `*`, `/`, `%`
 - `<=`, `>=`, `<`, `>`, `==`, `!=`, `!`, `&&`, `||`
- Declare an integer
 - `declare -i intName`
 - Integers are faster to evaluate

Arithmetic Example

```
#!/bin/bash
```

```
#
```

```
# Demonstrate some Bash arithmetic
```

```
#
```

```
declare -i x=10
```

```
while (( x > 0 ))
```

```
do
```

```
    echo x is $x
```

```
    (( x-- ))
```

```
done
```

Another Example

```
#!/bin/bash
# find max area x*y, where x+y=20
declare -i x=20
declare -i y
declare -i maxarea=-1
declare -i area

while (( $x > 0 )); do
    y=20
    while (( $y > 0 )); do
        (( area = $x * $y ))
        if (( area > maxarea && x + y == 20 ))
        then
            maxarea=$area
        fi
        (( y-- ))
    done

    (( x-- ))
done
echo "Max area is $maxarea"
```

If statement

- Allows “elif” (else if) as well as “else”

```
#!/bin/bash  
#
```

```
echo "enter your favorite class: "  
read favorite
```

```
# Use " " around $favorite so it keeps multiple  
# words together.
```

```
if [ "$favorite" == "Systems Programming" ]; then  
    echo "good choice."  
elif [ "$favorite" == "3320" ]; then  
    echo "good choice."  
else  
    echo "what are you thinking?!"  
fi
```

Review

- Bash overview
- Compatible with Bourne shell
- Allows arrays
- Has a history mechanism (as do others)
- Handles arithmetic well
- Control structures are very similar to Bourne