Chapter 11

Advanced Programming

Overview

- Shell Arithmetic
- The `select` Statement
- Terminal Independence in Scripts
- The `eval` Command

Lesson: Shell Arithmetic

- Working with arithmetic
- Arithmetic Operators
- Working with Numeric Data

Working with Arithmetic

- **Shell has built-in rudimentary arithmetic capabilities**
  - Arithmetic expansion - `$((expression))`
  - `Expr` command
- **Example:**

```
$ count=19
$ x=$((4+count))
$ y=$(expr $count + 8)
$ echo $x $y
23 27
```

- **Korn shell family can recognize `let` command and arithmetic test `((expression))`**

```
$ let "x = 4 + count"
$ if (( x > 3)) ...
```

Arithmetic Operators

Operator	Operation	Operator	Operation
+	Add	+=	Add assign
-	Subtract	-=	Subtract assign
*	Multiply	*=	Multiply assign
/	Divide	/=	Divide assign
%	Modulo	( )	Group

```
$ b=5
$ echo $(( b / 2 ))
2
$ echo $((var=(b+2)*3))
21
$ echo $var
21
```

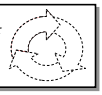
Working with Numeric Data

`bc`	Desktop calculator
`eval`	Evaluate command
`awk`	UNIX command/program language
`perl`	UNIX command/program language
`ls -l file | cut`	Pipelines producing numbers
`$(wc -l < file)`	Command substitution producing numbers
`${#string}`	Length of string variable contents
`$$`	Current shell Process ID (PID)
`$PPID`	Integer number of parent process
`$RANDOM`	Random integer
`$SECONDS`	Elapsed seconds

Lesson: The `select` Statement

- **Syntax**

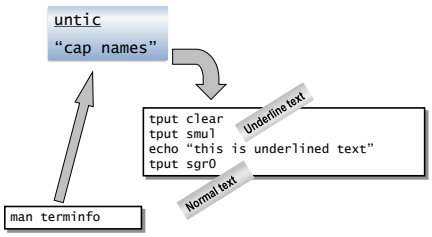
```
PS3="prompt string"
select var in argument_list
do
  commands
done
```



- **Example**

```
PS3="Command?"
select ans in ls pwd date quit
do
  if [ "quit" = "$ans" ] ; then
    break
  else
    $ans
  fi
done
```

Lesson: `tput` – change terminal characteristics



Lesson: The `eval` Command

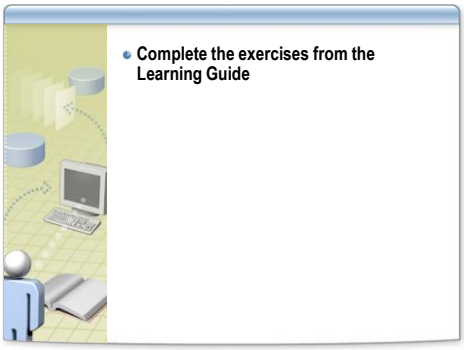
- **Syntax:**

```
eval [argument ...]
```

  - Construct command by concatenating arguments
- **Helps build commands on the fly**
- **Example:**

```
$ set one two three four
$ echo $#
4
$ echo '$'$#
$4
$ eval lastarg='$'$#
$ echo $lastarg
four
```

Review Exercises



Topics for Review