Script
#!/bin/bash
or
#!/usr/bin/env bash

chmod +x filename
./filename

comments
command ; command ; command

Variables
varname=value
echo $varname

Built in variables:

$1-$N	Arguments passed script by caller
$?	exit value of previous command
$0	name of shell script
$*	array of unquoted arguments from caller ($0 $1 $2 ...)
"$@"	array of arguments from caller ("$1" "$2" ...).

Quotes	ignore whitespace and count as one argument
" … "	interpolate $variable expressions
' … '	don't interpolate
$' … '	respect \n and UTF-8 definitions
` … `	return stout of expression when executed *(backticks)*
$(…)	alternative syntax for backticks

Subshells and grouping

$(command; command)
(command ; command)

Redirection

>
>>
2>&1

Arithmetic

let a=0
b=$((a++))

Variable assignment

a=b
a=b{,,,,,}

Command-line processing

a=b
a=b{,,,,,}

Logical Expression

Numeric Comparisons

int1 -eq int2	True if int1 is equal to int2.
int1 -ge int2	True if int1 is greater than or equal to int2.
int1 -gt int2	True if int1 is greater than int2.
int1 -le int2	True if int1 is less than or equal to int2
int1 -lt int2	True if int1 is less than int2
int1 -ne int2	True if int1 is not equal to int2

String Comparisons

str1 == str2	True if str1 is identical to str2.
str1 != str2	True if str1 is not identical to str2.
str	True if str is not null.
-n str	True if the length of str is greater than zero.
-z str	True if the length of str is equal to zero

File Comparisons

-d filename	True if filename is a directory.
-f filename	True if filename is an ordinary file.
-r filename	True if filename can be read by the process.
-s filename	True if filename has a nonzero length.
-w filename	True if filename can be written by the process.
-x filename	True if filename is executable.

Compound logical expressions

if [[expression l*og-op expression*]]; then ...
if [expression -{ a l o }]; then …
if [! expression]; then ...

Flow Control

```
if [ expression ]; then
    commands
elif [ expression2 ]; then
    commands
else
    commands
fi

case string1 in
    str1)  commands;;
    str2)  commands;;
    *)     commands;;
esac

for var1 in list; do
    commands
done

while [ expression ]; do
    commands
done

until [ expression ]; do
    commands
done
```

Functions

```
fname(){
    commands
}
# Call it by using the following syntax: fname

fname2 (arg1,arg2...argN){
    commands
}
# call with: fname2 arg1 arg2 ... argN
```

On-line resources

ihttp://www.ibm.com/developerworks/aix/library/au-unix-getopt.html
http://wiki.bash-hackers.org/howto/getopts_tutorial

Terrific
http://tldp.org/LDP/abs/html/index.html
http://wiki.bash-hackers.org/doku.php
http://mywiki.wooledge.org/BashGuide

http://tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html#chap_10

Books (both O'Reilly)

Acknowledgement
http://www.linux-sxs.org/programming/bashcheat.html

robin@clanmills.com