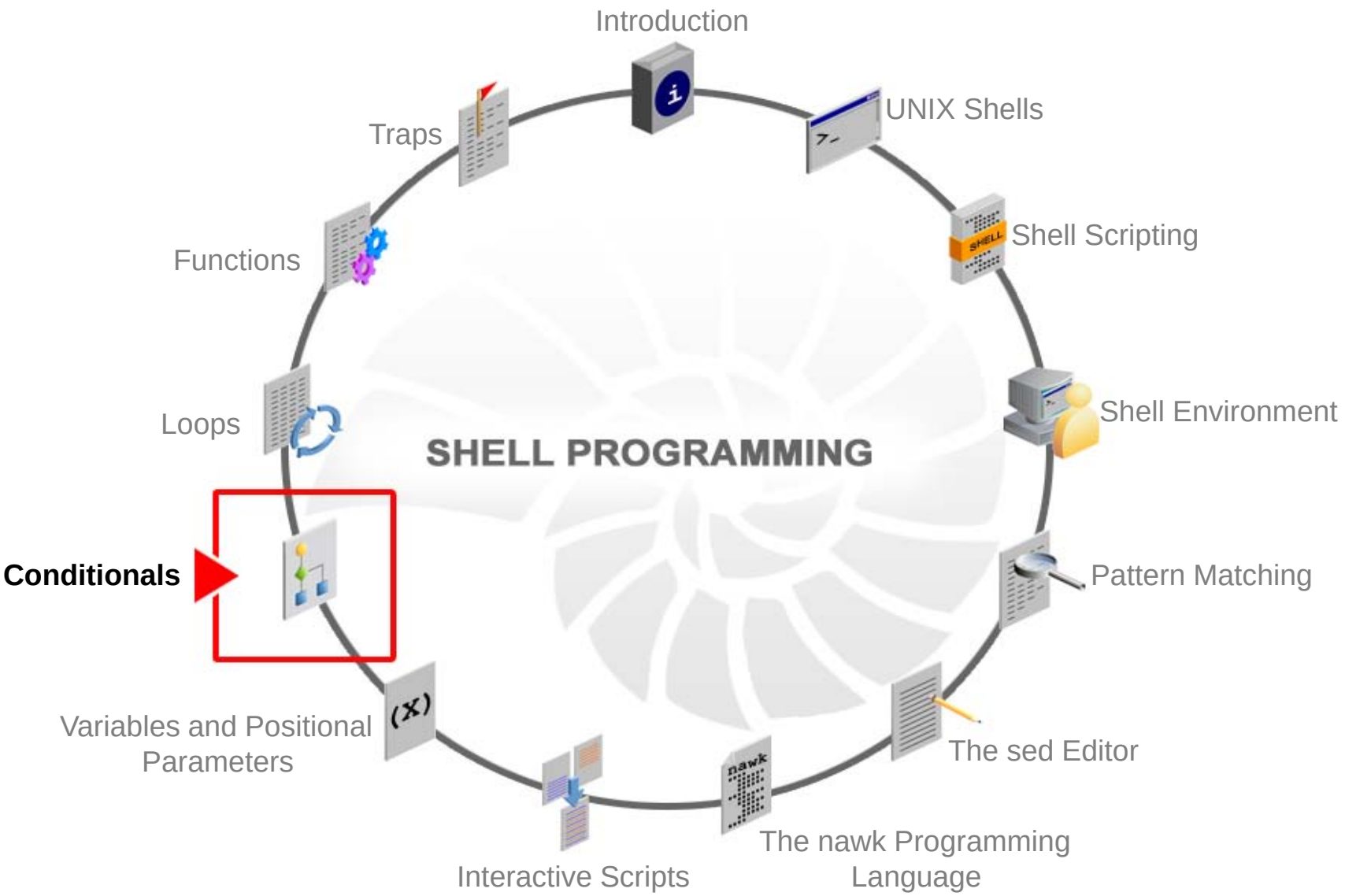
10

Conditionals



Objectives

After completing this lesson, you should be able to:

- Describe the various forms of `if` statements and their role in testing conditions
- Explain the use of `if` statements through examples
- Describe the role of the `case` statement in choosing from alternatives

Agenda

- Describing the various forms of `if` statements and their role in testing conditions
- Explaining the use of `if` statements through examples
- Describing the role of the `case` statement in choosing from alternatives

The `if` Statement

- Allows you to specify courses of action in a shell script, depending on the success or failure of some command
- Is a conditional statement that allows a test before performing another statement
- Has several forms.
  - The simple `if` statement
  - The `if/then/else` statement
  - The `if/then/else/elif` statement
  - The nested `if` statement
- In its simplest form, an `if` statement has the following syntax:

```
if command
then
            block_of_statements
fi
```

Parts of the `if` Statement

The `if` statement contains three parts:

- Command
- Block of statements (if-block)
- End of the `if` statement

The `if` Statement: Example

```
$ cat snoopy.sh
#!/bin/bash

# Script name: snoopy.sh

name=snoopy

if [ "$name" = "snoopy" ]
then
        echo "It was a dark and stormy night."
Fi

$ ./snoopy.sh
It was a dark and stormy night.
```

The `if/then/else` Statement

- If the command in the `if` statement succeeds, the block of statements after the `then` statement are executed.
- If the command in the `if` statement fails, then the block of statements after the `then` statement are skipped, and statements following the `else` are executed.
- In both the above cases, execution continues with the statement following the `fi` statement.
- Syntax:

```
if command
then
block_of_statements
else
block_of_statements
fi
```

The `if/then/else` Statement: Example

```
$ cat snoopynap.sh
#!/bin/bash
# Script name: snoopynap.sh
name=snoopy
if [[ "$name" == "snoopy" ]]
then
        echo "It was a dark and stormy night."
else
        echo "Snoopy is napping."
fi
$ cat findroot.sh
#!/bin/bash
# Script name: findroot.sh
if grep root /etc/passwd > /dev/null
then
        echo "Found root!"
else
        echo "root not in the passwd!"
        echo "Do not logout until the passwd file is repaired!"
fi
```

The `if/then/else/elif` Statement

- The shell first evaluates the commands in sequence.
- The statements associated with the first successful command are executed, followed by statements after `fi`.
- If none of the commands succeeds, then the statements following the `else` statement are executed.
- Execution continues with any statements following `fi`.
- Syntax:

```
if command1
then
    block_of_statements
elif command2
then
    block_of_statements
else
    block_of_statements
fi
```

The `if/then/else/elif` Statement: Example

```
$ cat snoopy2.sh
#!/bin/bash
# Script name: snoopy2.sh
name=snoopy
if [[ "$name" == "snoopy" ]]
then
        echo "It was a dark and stormy night."
elif [[ "$name" == "charlie" ]]
then
        echo "You're a good man Charlie Brown."
elif [[ "$name" == "lucy" ]]
then
        echo "The doctor is in."
elif [[ "$name" == "schroeder" ]]
then
        echo "In concert."
else
        echo "Not a Snoopy character."
fi
```

The `if/then/else/elif` Statement: Example

```
$ cat termcheck.sh
#!/bin/bash
# Script name: termcheck.sh
if [[ "$TERM" == "sun" ]]
then
        echo "You are using the sun console device."
elif [[ "$TERM" == "vt100" ]]
then
        echo "You are using a vt100 emulator."
elif [[ "$TERM" == "dtterm" ]]
then
        echo "You are using a dtterm emulator."
else
        echo "I am not sure what emulator you are using."
fi
```

Nested `if` Statements

- You can use an `if` statement inside another `if` statement.
- You can have as many levels of nested `if` statements as you can track.
- Each `if` statement requires its own `fi` statement.

Nested `if` Statements: Example

```
$ cat leap.sh
#!/bin/bash
# Script name: leap.sh
# Make sure the user enters the year on the
# command line
# when they execute the script.
if [ $# -ne 1 ]
then
echo "You need to enter the year."
exit 1
fi
year=$1

(Continued…)
```

Nested `if` Statements: Example

```
if (( (year % 400) == 0 ))
then
        print "$year is a leap year!"
elif (( (year % 4) == 0 ))
then
        if (( (year % 100) != 0 ))
        then
                print "$year is a leap year!"
        else
                print "$year is not a leap year."
        fi
else
        print "$year is not a leap year."

$ ./leap.sh 2000
2000 is a leap year!
$ ./leap.sh 2014
2014 is not a leap year.
$ ./leap.sh 2050
2050 is not a leap year.
```

The `exit` Status

- When a statement executes, the statement returns a numeric value called an `exit` status.
- The `exit` status is an integer variable and is saved in the `?` shell reserved variable.
- A statement that executes successfully, returns an `exit` status of `0`, meaning zero errors.
- A statement that executes unsuccessfully returns an `exit` status of nonzero, meaning one or more errors occurred.

The exit Status: Examples

- The following example uses `grep` to search the `/etc/passwd` file for the `root` string.

```
$ grep root /etc/passwd
root:x:0:0:Super-User:/root:/usr/bin/bash
$ echo $?
0
```

- You may redirect or pipe `stdout` as part of the output.

```
$ if grep root /etc/passwd > /dev/null
> then
>          echo "Found root!"
> fi
Found root!
```

Agenda

- Describing the various forms of `if` statements and their role in testing conditions
- Explaining the use of `if` statements through examples
- Describing the role of the `case` statement in choosing from alternatives

Using `if` Statements

The `if` statements can be used with various types of scripting parameters and operators for testing conditions such as:

- Numeric and string comparison operators
- Pattern-matching metacharacters
- Positional parameters
- Boolean `AND`, `OR`, and `NOT` operators

Numeric and String Comparison Operators

Shell scripting provides integer and string comparison capabilities in the form of comparison operators.

- For numeric comparison, use [ ] and spaces.
- For string comparison, use […] and spaces.

Numeric Comparison Operators

Bash Shell	Returns true (0) if:
`[ $num1 -eq $num2 ]`	*num1* **equals** *num2*
`[ $num1 -ne $num2 ]`	*num1* **does not equal** *num2*
`[ $num1 -lt $num2 ]`	*num1* **is less than** *num2*
`[ $num1 -gt $num2 ]`	*num1* **is greater than** *num2*
`[ $num1 -le $num2 ]`	*num1* **is less than or equal to** *num2*
`[ $num1 -ge $num2 ]`	*num1* **is greater than or equal to** *num2*

Numeric Comparison Operators: Example

```
$ num=21

$ if [ num > 15 ]; then
>     echo "You are old enough to drive in most places."
> fi
You are old enough to drive in most places.
```

String Comparison Operators

Bash Shell	Returns `true` (`0`) if:
`[ str1 = str2 ]`	*str1* **equals** *str2*
`[ str1 != str2 ]`	*str1* **does not equal** *str2*
`[ str1 < str2 ]`	*str1* precedes *str2* **in lexical order**
`[ str1 > str2 ]`	*str1* follows *str2* in lexical order
`[ -z str1 ]`	`str1` has length zero (holds null value)
`[ -n str1 ]`	`str1` has nonzero length (contains one or more characters)

String Comparison Operators: Example

```
$ name=fred

$ if [ "$name" == "fred" ]
> then
>    echo "fred is here"
> fi
fred is here
```

Pattern Match Metacharacters

Metacharacter	Meaning
?	Matches any one single character
[ ]	Matches one character in the specified set
*	Matches zero or more occurrences of any characters
?(*pat1*|*pat2*|...|*patn*)	Matches zero or one of the specified patterns
@(*pat1*|*pat2*|...|*patn*)	Matches exactly one of the specified patterns
*(*pat1*|*pat2*|...|*patn*)	Matches zero, one, or more of the specified patterns
+(*pat1*|*pat2*|...|*patn*)	Matches one or more of the specified patterns
!(*pat1*|*pat2*|...|*patn*)	Matches any pattern except the specified patterns

Pattern Match Metacharacters: Example

```
$ name=fred
$ if [ "$name" == f* ]; then
>    echo "fred is here"
> fi
fred is here
```

Pattern Match Metacharacters: Example

```
$ cat monthcheck
#!/bin/bash
# Script name: monthcheck
mth=$(date +%m)
if (( mth == 2 ))
then
        echo "February usually has 28 days."
        echo "If it is a leap year, it has 29 days."
elif [[ $mth = @(04|06|09|11) ]]
then
        echo "The current month has 30 days."
else
        echo "The current month has 31 days."
fi

$ date
Fri Mar 13 13:28:24 GMT 2000

$ ./monthcheck.sh
The current month has 31 days.
```

Positional Parameters

Parameter Name	Description
$0	The name of the script
$1	The first argument to the script
$2	The second argument to the script
$9	The ninth argument to the script
$#	The number of arguments to the script
$@	A list of all arguments to the script
$*	A list of all arguments to the script

Positional Parameters: Example

```
$ cat numtest.sh
    #!/bin/bash
    # Script name: numtest.sh
    num1=5
    num2=6
    if (( $num1 > $num2 ))
    then
            print "num1 is larger"
    else
            print "num2 is larger"
    fi

$ ./numtest.sh
    num2 is larger
```

Positional Parameters: Example

```
$ cat argtest.sh
    #!/bin/bash
    # Script name: argtest.sh
    if (( $1 > $2 ))
    then
            print "num1 is larger"
    else
            print "num2 is larger"
    fi

$ ./argtest.sh 21 11
    num1 is larger

$ ./argtest.sh 1 11
    num2 is larger
```

Positional Parameters: Example

- The script should verify that the type of input and the number of values input are correct.
- If they are incorrect, then the script can print an error message in the form of a USAGE message.
- Example:

```
if (( $# != 2 ))
then
print “USAGE: $0 arg1 arg2 “
exit
fi
```

Boolean AND, OR, and NOT Operators

- Boolean operators define the relationships between words or groups of words.
- Following are the Boolean operators:
  - AND operator is &&
  - OR operator is ||
  - NOT operator is !

Boolean AND, OR, and NOT Operators: Example

```
$ cat leap2.sh
#!/bin/bash
# Script name: leap2.sh

# Make sure the user enters the year on the command line
# when they execute the script.
if [ $# -ne 1 ]
then
fi

year=$1
echo "You need to enter the year."
exit 1
if (( ( year % 400 ) == 0 )) ||
        (( ( year % 4 ) == 0 && ( year % 100 ) != 0 ))
then
        print "$year is a leap year!"
else
        print "$year is not a leap year."
fi
```

Agenda

- Describing the various forms of `if` statements and their role in testing conditions
- Explaining the use of `if` statements through examples
- Describing the role of the `case` statement in choosing from alternatives

The case Statement

The case statement helps choose from several alternatives.

```
case value in
pattern1)
        statement1
        ...
        statementn
        ;;
pattern2)
        statement1
        ...
        statementn
        ;;
*)
        statement1
        ...
        statementn
        ;;
esac
```

The case Statement: Example

```
$ cat case.sh
#!/bin/bash
# Script name: case.sh
mth=$(date +%m)
case $mth in
02)
        print "February usually has 28 days."
        print "If it is a leap year, it has 29 days."
        ;;
04|06|09|11)
        print "The current month has 30 days.”
         ;;
*)
        print "The current month has 31 days."
        ;;
esac
```

The case Statement: Example

```
$ date
Thursday, March 6, 2014 01:09:38 PM IST

$ ./case.sh
The current month has 31 days.
$
```

Replacing Complex `if` Statements with `case` Statements

```
$ cat snoopy2.sh
#!/bin/bash
# Script name: snoopy2.sh
name=snoopy
if [[ "$name" == "snoopy" ]]
then
        echo "It was a dark and stormy night."
elif [[ "$name" == "charlie" ]]
then
        echo "You're a good man Charlie Brown."
elif [[ "$name" == "lucy" ]]
then
        echo "The doctor is in."
elif [[ "$name" == "schroeder" ]]
then
        echo "In concert."
else
        echo "Not a Snoopy character."
fi
```

Replacing Complex `if` Statements with `case` Statements

```
$ cat snoopy3.sh
#!/bin/bash
# Script name: snoopy3.sh
name=lucy
case $name in
"snoopy")
        echo "It was a dark and stormy night."
        ;;
"charlie")
        echo "You're a good man Charlie Brown."
        ;;
"lucy")
        echo "The doctor is in."
        ;;
"schroeder")
        echo "In concert."
        ;;
*)
echo "Not a Snoopy character."
;;
esac
```

Summary

In this lesson, you should have learned how to:

- Describe the various forms of `if` statements and their role in testing conditions
- Explain the use of `if` statements through examples
- Describe the role of the `case` statement in choosing from alternatives

Practice 10 Overview: Conditionals

This practice covers the following topics:

- Using Conditionals
  - You answer questions about the usage of conditionals and you create a script that will determine whether the pathname is a directory or a file.
- Modifying the `adduser` Script by Using Conditionals
  - You modify the existing `adduser` script to add more functionality with the use of conditionals to create a new user.