Chapter 7

Shell Scripting: Condition Statements

In this chapter, you will learn the following to World Class standards:

1. **Adding Comments**
2. **Defining the Program**
3. **Prompting the User**
4. **Capturing System Data**
5. **Creating the Conditional Statement**
6. **Making the Script an Executable File**

An Interactive Shell Script

A common function in programming is the conditional statement. These functions work by triggering a specific action when certain predetermined conditions are met. For our example program, we will be using a short, simple script that checks the system date against the user-defined date, and then notifies the user if there is a discrepancy between the two, signifying that there may be an error with the machine, or the user.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.1 – Day of the Week Check Program

Adding Comments

Just like the example program we used in the previous chapter, this program uses many comments inside of the script to help describe the purpose of the program as well as the functions of individual lines of code. If we wanted we could also add copyright information into the script by using comments.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.2 – Adding Comments to Define the Program and Lines of Code

Defining the Program

The second line of code displays the purpose of the program for the user. This is considered proper programming etiquette, and is especially useful if there are more than one executable shell files in a single directory. Use the **echo** command and enter the text in quotation marks as shown below.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.3 – Outputting the Program Definition

Prompting the User

Next we will prompt the user to supply the original data to be checked by the program. Type **echo –**and then in quotation marks we ask the user for the day of the week, specifying the full name of the day. The next line of code contains the Read command. Type **read day** to record the data received from the user as variable **day**. Notice again that after each line of code there is a comment describing its purpose.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.4 – Prompting the User

Capturing System Data

Before we can set up our conditional statement, we need to capture the date from the system to a variable. First let's define the variable as **sdate**. Next we need to enter a command that will capture the system date at the exact time the program is run: for this we will use the Date command. Type **$(date + "%A")** so that the program will capture the current date into the **sdate** variable. The "**%A**" extracts the full system day to the variable.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.5 – Capturing the System Date

Creating the Conditional Statement

Now we are ready to program our conditional statement. Start by typing **if**, and then add **"$day" = "$sdate"** inside brackets. By adding this we have defined the condition that must be met before the action attached to our Then command will be executed. To continue, type **then** and on the next line program a text string that displays a message saying that the date is correct. If our previous condition is met, and the system date matches the date input by the user, the program will display that message.

```
# dotw is a program that check the day of the week against the system day
echo "This program checks the day of the week"              # program definition
echo -n "What is the day of the week? Full name please."   # read day
read day                                                   # user prompt
sdate=$(date +"%A")                                        # retrieve the system day
if [ "$day" = "$sdate" ]
then
echo "System date is correct"
else
echo "Check system date for error"
fi                                                         # end of program
```

Figure 6.6 – Conditional Statement Code

The last part of the conditional statement is an action that will be triggered if the condition is not met. For this we will use the **else** command, in exactly the same way we used the **then** command: on the next line of code create a text string that will display a message that alerts the user to the difference in dates. To end the conditional statement, type **fi**.

Making the Shell Script an Executable File

Now that we are done coding in the text editor, we can save the file as **dotw.exe** or just **dotw**. If we save the file as **dotw**, we can convert it to an executable file by typing in the Bash command **chmod ugo+x dotw**. To run the program, type **sh dotw.** When prompted for the date, type the correct or incorrect date to see if the program behaves as expected. If not you should recheck your script.

*** World Class CAD Challenge 44-7 * - Write a Script that displays two message boxes, the first will contain the script name, copyright date and author. The second message will display information from the computer.**

Continue this drill four times using some other messages, each time completing the VBScript in less than 30 minutes to maintain your World Class ranking.