UNIX and Shell Programming

Assignment 10: Control Flow I

10 September 2019

1 Exercise 1

1. Write a shell script that prints 10 command line arguments. What happens if we pass fewer than 10 arguments?

2. Change the value of a positional parameter. Did you succeed?

2 Exercise 2

1. Define a shell function `fun()`. Let it take 2 parameters and print them. Call `fun()` from the main script with two arguments.

2. Modify the script so that the main script reads two command line arguments, and prints them. Let it call `func` with two arguments. What does each print?

3. Define a variable in the main script. Print it in the main script and inside the function. Does the function print the value of the variable?

4. Change the value of the variable inside the function. Print its value before and after the function call. Did the value change?

5. Define a variable inside the function. Print it before and after the function call. What do you observe?

6. Define a local variable inside the function. Print it before and after the function call. What do you infer?

3 Exercise 3

1. Write a shell function `minimum` that takes two parameters and prints the smallest of the two. Demonstrate it by calling it from main shell script.

2. Write a shell function `maximum` that takes two parameters and returns the largest of the two. Demonstrate it by calling it, and then printing the exits status.

3. Anna University converts the marks in an exam to letter grades according to the following table. Write a shell script to translate the marks of a student in a semester into letter grades.

Mark range	Grade points	Leter grade
91-100	10	S
81-90	9	A
71-80	8	B
61-70	7	C
57-60	6	D
51-56	5	E
< 50	0	U

4 Exercise 4

1. Write a shell function `string-compare` that takes two strings `s1` and `s2` as arguments and returns 1 if `s1` comes before `s2` by ASCII order, 0 if `s1` is the same as `s2`, and 2 if `s1` comes after `s2`.

5 Exercise 5

1. Run the commands `true` and `false` and check their exit status.

2. Write a script that prints essentially the same information as `ls -l` a but in a more user-friendly way.

   (a) file exists or not

   (b) regular file?

   (c) directory?

   (d) readable?

   (e) writable?

   (f) executable?

   (g) owner

   Print suitable messages.

```
# test-file: Evaluate the status of a file
FILE=$1
if [ -e "$FILE" ]; then
   if [ -f "$FILE" ]; then
      echo "$FILE is a regular file."
   fi
   if [ -d "$FILE" ]; then
      echo "$FILE is a directory."
   fi
   if [ -r "$FILE" ]; then
      echo "$FILE is readable."
   fi
   if [ -w "$FILE" ]; then
      echo "$FILE is writable."
   fi
   if [ -x "$FILE" ]; then
         echo "$FILE is executable/searchable."
   fi
else
    echo "$FILE does not exist"
    exit 1
fi
exit
```

3. Rewrite the script as a shell function `finfo` and call the function with a filename.

6 Exercise 6

1. The syntax of `for` command is:

```
for variable in list
do
  command1
  command2
  ...
done
```

or

```
for variable in list; do
  command1
  command2
  ...
done
```

where `list` is a list of strings (the keywords `do` and `done` should be preceded by newline or semicolon). Write a script to print a sequence of numbers from 1 to 10.

```
for v in 1 2 3 4 5
do
   echo $v
done
```

2. `((expression))` evaluates `expression`. `$((expression))` accesses the value of the expression. We can use the value in another command such as `echo`. Using `for` statement, write a script to print the multiplication table. Take the number from the command line.

```
u=5
v=2
v=$((u+v))
echo $v
echo $((v+1))
```

3. `seq` prints a sequence of numbers. The syntax is
   ```
   seq last
   seq first last
   seq first step last
   ```
   Print a list of numbers from 10 to 50 in steps of 5.

4. Use `seq` with `for` statement to print the multiplication table.

5. Filenames are strings. Write a script to list the informative contents of a directory using `finfo`.

6. Change the last script to print informative list of only the regular files.