Study Questions: Set No. 3
Introduction to UNIX
Saturday October 5, 2013

Covering:

Topic 08: *Shell Programming*

1. Is it possible to execute a shell Bourne script if you are not given execute access permission? If yes, how?

2. Is it possible to execute a shell csh script if you are not given execute access permission? If yes, how?

3. What is the command that used to make a file an executable file?

4. How do you debug a shell script?

5. In Bourne-style shells, what is the difference between the `while` and `until` loops?

6. In Bourne-style shells, what is the command that used to read input from the standard input? Give an example.

7. In Bourne-style shells, what is the Unix command that adds two integer numbers? Give an example.

8. In Bourne-style shells, what is the Unix command that multiplies two integer numbers? Give an example.

9. In Bourne-style shells, how do you compare two numeric values? Give an example.

10. In Bourne-style shells, how do you compare two strings? Give an example.

11. Consider `script.sh` that contains the following script
```
#!/bin/sh
echo $0
```
Explain the output when you execute `script.sh ABC`

12. Write a Bourne shell script named `LL` that lists your current directory in a long format.
    a. Execute `LL` using the `sh` command (i.e., `sh ./LL`)
    b. How do you execute `LL` again without using `sh` command

13. Write a Bourne script file that performs the following:
    - Clearing the screen
    - Showing the current date and time
    - Showing the current number of users on the system

14. In Bourne shell, if `x=10`, what you will get if you execute `echo $x$x; echo x$x$`

15. What is the output of the following shell script?
```
#!/bin/sh
x=5
echo  expr $x       + 10
echo "expr $x       + 10"
echo 'expr $x       + 10'
echo `expr $x       + 10`
```

16. Explain what happens when you execute the following commands:
```
new_command=ls
echo $new_command
echo "$new_command"
echo '$new_command'
echo `$new_command`
```

17. If you have a shell script called `new_file` as listed below:

```
#!/bin/sh
#
echo $0
echo $1
echo $#
echo $*

shift

echo $0
echo $1
echo $#
echo $*
```

Explain what happens when you execute the following commands:
`new_file a b c`

18. Write a Bourne shell script called `file_checker` that reads a filename from the standard input and produces the properties of that file (e.g., exists, readable, executable).

19. Write a Bourne shell script called `executable` that lists the names of all executable files in the current directory.

20. Write a Bourne shell script called `s` that displays the name of your login shell.

21. Write a Bourne script file that sums the numbers passed to it as arguments on the command line and displays the results. Use a **`for`** loop in your program. If this program is called `SUM`, and you execute
`SUM 10 20 30`
the program should display the following:
`10 + 20 + 30 = 60`

22. Write a Bourne script file that sums the numbers passed to it as arguments on the command line and displays the results. Use a **`while`** loop in your program. If this program is called `SUM`, and you execute
`SUM 10 20 30`
the program should display the following:
`10 + 20 + 30 = 60`

23. Write a Bourne script file that sums the numbers passed to it as arguments on the command line and displays the results. Use a **`until`** loop in your program. If this program is called `SUM`, and you execute
`SUM 10 20 30`
the program should display the following:
`10 + 20 + 30 = 60`

24. Explain what happens when you execute the following shell script

```
#!/bin/sh
#
echo "Please enter a name: "
read name

if who | grep -s $name > /dev/null
then
  echo $name is logged
else
  echo no such user $name
fi
```

25. Trace and explain the following shell script.

```
#!/bin/sh
#
echo
echo "Are you OK? "
echo -n "Input Y for yes and N for no: "
read answer

if test "$answer" = Y
then
  echo "Glad to hear ! "
elif test "$answer" = N
  then
    echo "Go home! "
  else
    echo "Your answer should be Y or N"
fi
echo
```

Is this script case insensitive to your input?
If not, then how do you modify it to make it case insensitive?

26. Explain what happens when you execute the following commands:

```
pwd
mkdir new_dir
cd new_dir

cat <<+ > new_file
#!/bin/sh

echo "I am inside new_file"
echo "Current directory is `pwd`"
+
chmod u+x new_file
/bin/ls –l
`/bin/ls`
cd ..
rm -r new_dir
pwd
```

27. Explain what happens when you execute the following shell script

```
#!/bin/sh
#
echo
hour=`date +%H`
if [ "$hour" -lt 12 ]
then
  echo "GOOD MORNING"
elif [ "$hour" -lt 18 ]
  then
    echo "GOOD AFTERNOON"
  else
    echo "GOOD EVENING"
fi
echo
```

If you decided not to use "`elif`", what you should change in the program to keep it works the same way.

28. Consider that you executed the following command:
```
(echo a b c; echo 1 2 3) > data_file
```
Also consider that you have a shell script called `script.sh` as listed below:
```
#!/bin/sh
while read a b c
do
  echo $a $a $b $b $c $c
  echo $a $a $b $b $c $c
done | tr a-z A-Z
```
Trace and explain the output of the following command:
```
script.sh < data_file
```

29. Consider you executed the following command:
```
(echo a b c; echo 1 2 3) > data_file
```
Also consider that you have a shell script called `script.sh` as listed below:
```
#!/bin/sh
while read a b
do
  echo $a $a $b $b $c $c
  echo $a $a $b $b $c $c
done | tr a-z A-Z
```
Trace and explain the output of the following command:
```
script.sh < data_file
```

30. Consider you executed the following command:
```
(echo a b c; echo 1 2 3) > data_file
```
Also consider that you have a shell script called `script.sh` as listed below:
```
#!/bin/sh
while read a
do
  echo $a $a $b $b $c $c
  echo $a $a $b $b $c $c
done | tr a-z A-Z
```
Trace and explain the output of the following command:
```
script.sh < data_file
```

31. Trace and explain the following shell script
```
#!/bin/sh
mkdir new_dir
cd new_dir
pwd > new_file
ln new_file new_file_2
rm new_file
cat new_file_2
cd ../
rm -r new_dir
```

32. Consider following Unix commands
```
cat <<+ > script_2.sh
#!/bin/sh
echo $0
+

cat script_2.sh
```
Did you find the content of `script_2.sh` as you typed? Explain why.
How do you fix the above write up so `script_2.sh` to contain what you typed.

33. Write a Bourne shell script that displays all command line arguments, even if they are more than 9 arguments. Hint: use `shift` command.

34. Write a Bourne shell script that accepts from the command line three integer numbers and sort them from the largest to the smallest.

35. Write a Bourne shell script that interactively reads from the user three integer numbers and sort them from the largest to the smallest.

36. Write a Bourne shell script that accepts two directory names, `dir1` and `dir2`, and deletes the files in `dir2` that are identical to their namesakes in `dir1`.

37. Write a Bourne shell script called `median` that takes one argument (`input-filename`) and gives the median number of the numbers in the provided file. Create a file called `input-filename`. Write the following numbers in file, one number in each line, (**`3, 6, 9, 11, 3, 4, -8, -10, 0, 16, 5`**). Test your scrip using the data file you have created. Hint: you may want to utilize `sort` and `wc` commands in your code.

38. Write a Bourne shell script that processes every file with name ended by `.c` in the current directory by searching inside it for the *keywords* `printf` or `fprintf`. If found, the script should add the statement

    ```
    #include <stdio.h>
    ```

    at the beginning of the file, *if, and only if*, the file does not already have it, *regardless of the number of spaces between* `#include` *and* `<stdio.h>`

39. Write a Bourne shell script that causes the following output (below) to be displayed. The number of column should be taken as input. For example, if the input to the program is 6, the program should produce the following output:

    ```
    0 1 2 3 4 5
    1 2 3 4 5
    2 3 4 5
    3 4 5
    4 5
    5
    ```

40. Write a Bourne shell script that causes the following output (below) to be displayed. The number of column should be taken as input. For example, if the input to the program is 6, the program should produce the following output:

    ```
    5
    4 5
    3 4 5
    2 3 4 5
    1 2 3 4 5
    0 1 2 3 4 5
    ```

41. Write a Bourne shell script that causes the following output (below) to be displayed. The number of column should be taken as input. For example, if the input to the program is 6, the program should produce the following output:

    ```
          *
         ***
        *****
       *******
      *********
     ***********
    ```

42. Write a Bourne shell script that computes the factorial of a positive integer. The number should be taken as input from the standard input.

43. Write a Bourne shell script that computes the factorial of a positive integer. The number should be taken as an inline parameter.

44. Write a Bourne shell script that takes a positive integer n as an argument and tell if n is prime, or not.

45. Write a Bourne shell script that takes a positive integer as input and returns the leading (the most significant) digit. For example, the leading digit of 234567 is 2.

46. Write a Bourne shell script that takes two parameters, *n* and *k* and returns the k^{th} digit (from the right) in *n* (a positive integer). For example, if the parameters are 829 and 1, it returns 9. If the parameters are 829 and 2, it returns 2. If the parameters are 829 and 3, it returns 8. If *k* is greater than the number of digits in *n*, have the script return 0.

47. Write a Bourne shell script that takes three parameters, *month*, *day*, and *year*, and returns the day of the year (an integer between 1 and 366).

48. Write a Bourne shell script that prints the prime numbers below *n*, where *n* is an input parameter.

49. Write a Bourne shell script that returns the number of digits in a positive integer, which is taken as a parameter. Hint: to determine the number of digits in a number, divide it by 10 repeatedly. When 0 is reached, the number of divisions indicates how many digits originally had.

50. Write a Bourne shell script that takes a *dollar amount* as an integer parameter and returns the smallest number of $20, $10, $5, $2, and $1 bills/coins necessary to pay this amount.

51. Write a Bourne shell script that dispenses change. The user enters the amount paid and the amount due. The script determines how many dollars, quarters, dimes, nickels, and pennies should be given as change.

52. Write a Bourne shell script that takes a *seconds* as a positive integer parameter (less than 86400) representing the number of seconds since midnight and returns the equivalent time in hours (0-23), minutes (0-59), and seconds (0-59), respectively.