LAB 4: FILE SYSTEM SCANNER

This lab walks you through building a simple file system scanner. The scanner will be able to do three things. First, it will generate a list of files and file attributes, including a checksum of each file's contents. Second, it will compare the current files and file attributes with the previous list. Third, it will delete the list.

PRELAB EXERCISE (5 PTS)

Before we start with the lab lets explore the shasum command that you will use.

A. Compute the shasum of dict.txt from the previous lab. Copy down the result to your answer sheet. (You are going to need to refer to it soon)
B. Update dict.txt by capitalizing the 'p' in the first line. Then compute the shasum of the updated dict.txt. Copy down the updated hash value under the previous hash value.
C. On, how many positions do the hash values match?
D. What useful characteristic of secure hash algorithms does the above demonstrate?
E. Recall that dict.txt had 50,000 lines. Find a file with much fewer lines (e.g. 5-10 lines). Compute the shasum of this file and write the hash value under the previous two.
F. Compare the length of the hash values for a large file versus the smaller file. What useful characteristic of secure hash algorithms is demonstrated?

LAB EXERCISE 0 (1 PT)

For this lab you will need the five files in the subdirectory "sample". Create the directory "sample" with the files:

- abc xyz
- abcde
- demofor
- demofor2
- demofor3

Please do not change anything in that directory — one of the tests you need to run depends on those files being unchanged.

LAB EXERCISE 1 (5 PTS)

Answer A-C. For part D submit a screen shot that uses cat to show the contents of your script, and shows the results of running your script.

A. List the attributes of the files in the directory "sample" using the command *ls* with the *–l* option. Do some research and explain what file attributes are printed?
B. What does the command *shasum* print?
C. How would you cause both outputs to be printed on the same line?
D. Write a shell script that uses a *for* loop to list the attributes of the files in the directory "sample", one per line. Call this script "scan1.sh".Run your script. You should get information for six files, including the one you wrote.

LAB EXERCISE 2

You are now going to create the MASTER file to record information about your file before the attack. Start with the script you wrote for Lab Exercise 1D.

A. At the beginning of the script, define a variable called MASTER, and give it the initial value "MasterList". Call this script "scan2a.sh".
   Make a copy of this script as scan2.sh and continue with the remaining parts. You will use the copy scan2a.sh later in the Lab.

B. In scan2.sh before the loop, test to see if the file reference by MASTER exists. If so, print the following message to **system error** "<*filename*> exists; please delete it", replacing <filename> with the value of MASTER. Then exit with status code 1. If no file with the value of MASTER exists, create it.

C. Update the *for* loop, so that it prints the file attributes of regular files. Skip over the file referred to by variable MASTER and any directories.

D. Store the attributes of regular files in the file named by the value of the variable MASTER.
E. Finally, have the script exit with an exit status code of 0.

   Run your script. There should be no output. Then look for a file called MasterList and compare its contents to what you got for the output of scan1.sh. The outputs for the files abc xyz, abcde, demo.for.sh, demofor2.sh, and demofor3sh should be the same.

   For this problem submit screen shots that show the contents of your scan2a.sh and scan2.sh files.

LAB EXERCISE 3

Rename the script scan2a.sh saved after completing Lab Exercise 2A as scan3.sh. (You can use the mv command to rename a file).

A. At the beginning of the script, define a variable called TMP, and give it the initial value "/tmp/$$". Then add another line that creates the file named by the value of TMP. Test the script. What is the actual name of the file that your script created? Delete the file using command rm.

B. Update the *for* loop, that prints file attributes, so that it skips over the file named by the value of TMP. For all other files, store the attributes in the file named by the value of the variable TMP.

C. After the loop, print the message "Changed files:", and then compare the contents of files named by variables TMP and MASTER. Use *diff* to generate the comparison. What error do you get if the file named by the value of MASTER does not exist?

D. Before the loop, check that the master file (named by MASTER) exists. If it does not, print the following message to **system error** "Master file does not exist; please generate it" and exit with an exit status code of 1.

E. Add a line at the end of the script that deletes the file named by TMP (use rm command). Exit the script with an exit status code of 0.

TESTING: When you test this script, the file named by variable MASTER must already exist. If it does not, execute the script you wrote in Lab Exercise 2.

Running your script, should output at least one line for the MASTER. You will also get lines corresponding to the scripts you wrote since creating the MASTER.

Also, from the command interpreter, check the exit code. Immediately after you execute your script, type

```
echo $?
```

This prints the exit status code of the command that finished most recently. You should see 0.

Submit a screen shot showing the contents of scan3.sh and the results of running scan3.sh. Also show the exit status code.

Answer the questions in part A, and C.

LAB EXERCISE 4

Now combine parts of the scripts you wrote for Lab Exercises 2 and 3 to write the script "scan4.sh".

Start by defining variables TMP and MASTER, then do the parts of Lab Exercises 2 and 3 in the following order: 2A, 3A, 3B, 2C, 3C, 3D, 3E, 2E

Do not include Exercises 2B or 2D in this script. Then, before the line you wrote for Exercise 3D, have the script print "Changed files:" and add the following on the same line as the command you wrote for Exercise 3D:

```
| grep '^\(+\|-\)' | awk '{ print $NF }' | sort | uniq
```

(The vertical bars "|" and quotation marks are critical.) This addition will change the output of your *diff* command so that the files that have been changed have their names listed in alphabetical order. It will make the output clearer for the user.

Run your script twice to test it. First, just run it. The list of changed files should have only the ones you have written and left in the directory. Then change the time of last modification of the file "abcde" by typing

```
touch abcde
```

Now rerun the script. The list should be the same as before but with the addition of "abcde".

For this exercise submit a screen shot showing the contents of your script (using cat) and a screen shot showing the output of your script on the first and 2nd runs above.

LAB EXERCISE 5

Modify the script from Lab Exercise 4 to create scan5.sh. This script will handle two options –g, which creates the master file, and -d, which deletes the master file. If the script is called with no options, it should generate a list of the files whose attributes or contents have changed since the master list was created.

A. Create variables GENMASTER, DELMASTER at the beginning of the script. Set these to "yes" if the corresponding option is given and to "no" otherwise.
B. If any command-line option (or argument) other than –d or –g is given, print to system error the message "Unknown option" followed by the option, and exit with an exit status code of 1.
C. If the –d option is given, check that the master file exists. If it does, delete the master file and exit with an exit status code of 0. If it does not, print to

system error "Master file does not exist; please generate it" and exit with a status code of 1.

D. If the –g option is set generate the MASTER file as you did in exercise 2. Then exit with an exit status code of 0. If the MASTER file already exists you should do as in exercise 2 and show an error message saying the master already exists.
E. If both the –g and –d options are set, print the system error message "Only one of –d, -g allowed" and exit with an exit status code of 1.
F. If neither are set, run the scan as you did in exercise 4. Exit with an exit status code of 0.

Submit a screen shot showing the contents of your scan5.sh file.

Run the following test scan5.sh and take a screen shot of each.

1. With both –g and –d options set: You should see an error message
2. With the –g option set: You should get the error message saying the master file exists, please delete it.
3. With the –d option set
4. With the –d set again: You should get an error message
5. With –g option set
6. With no options set

LAB EXERCISE 6

Make a copy of the previous script and call the copy "scan6.sh". (Use the cp command to copy).

In scan6.sh add an option –m that takes the argument following it to be the name of the master file and set the variable MASTER accordingly.

The script should be called as follows

```
scan6.sh -m filename
```

The given filename should be used for remaining tasks.

Be sure to handle the case in which –m is given and no filename follows with an error message.

Submit a screen shot showing the contents of the file scan6.sh.

Also show a screen shot showing how the script works when –m is set without a file name and how it works if it is set with a file name.