SECURE SCRIPTING

**ADVANCED CONTROL**

# LAB: 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 can compare the files and file attributes of the files in the directory with that list. Third, it can delete that list.

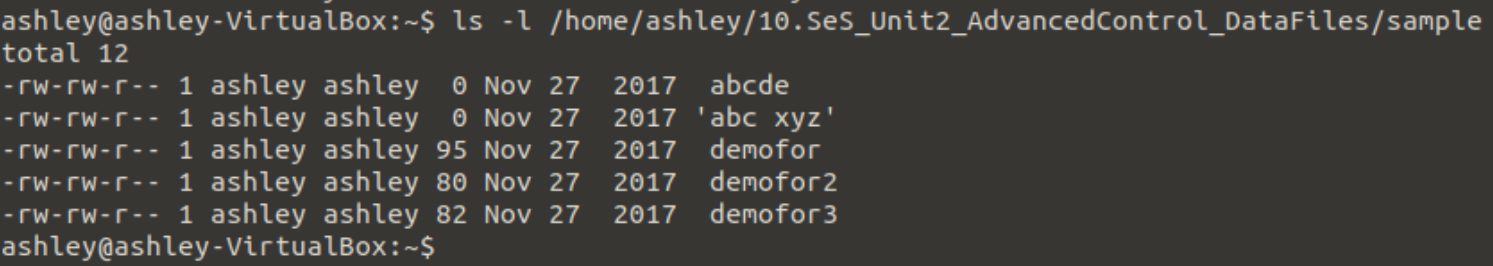
For this lab you will need the five files in the subdirectory “sample”. Please do not change anything in that directory — one of the tests you need to run depends on those files being unchanged.

* abc xyz
* abcde
* demofor
* demofor2
* demofor3

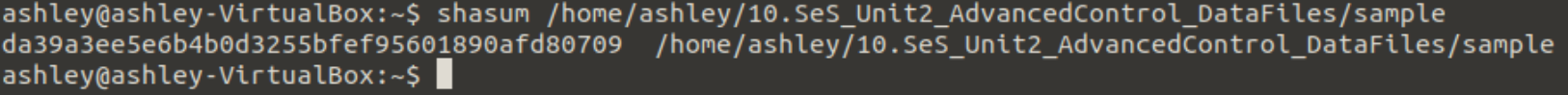
### Lab exercise 1

This exercise starts you off. You do not need to write any scripts until Part D. First, list the attributes of the files in the directory “sample”, one per line.

1. When you execute the command *ls* with the *–l* option, which of the desired attributes are printed? The read and write permissions.

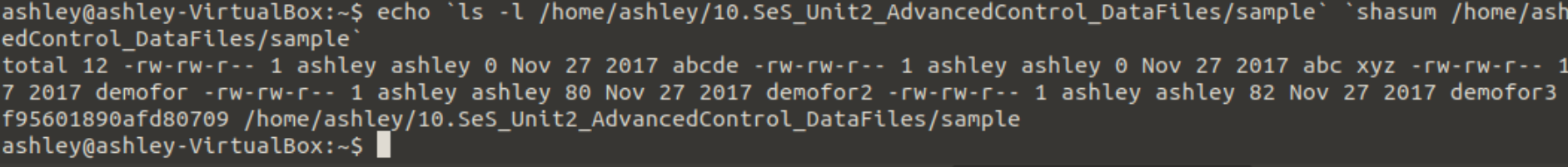


1. What does the command *shasum* print? It will compute a hash of the contents of the file.

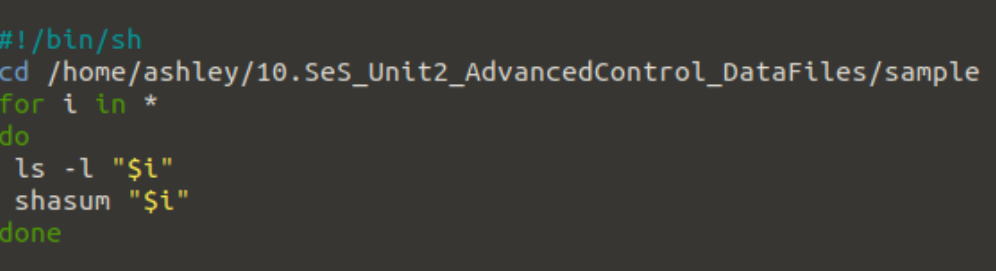


1. How would you cause both outputs to be printed on the same line? Use back ticks to enclose the output and list both outputs on the same line.

Echo `ls -l /home/Ashley/10.SeS\_Unit2\_AdvancedControl\_DataFiles/sample` `shasum /home/Ashley/10.SeS\_Unit2\_AdvancedControl\_DataFiles/sample`



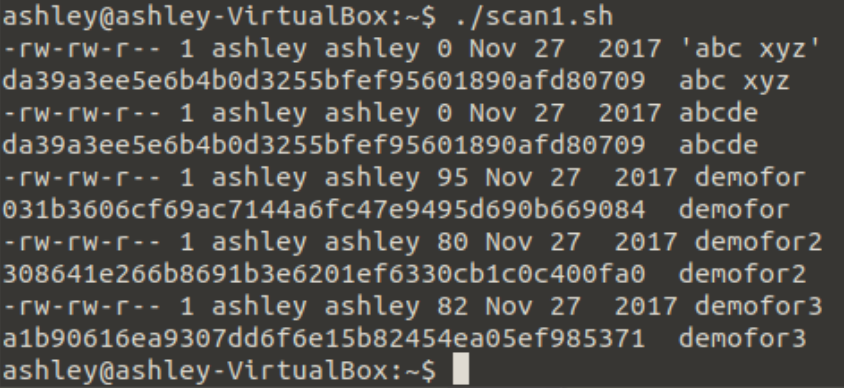
1. Write a shell script that uses a *for* loop to list the attributes of the files in the directory “sample”, one per line. Call this script “scan1.sh”.



Run your script. You should get information for six files, including the one you wrote. The checksums of the files “abcde” and “demofor.sh” are:

da39a3ee5e6b4b0d3255bfef95601890afd80709 abcde

031b3606cf69ac7144a6fc47e9495d690b669084 demofor.sh



### Lab Exercise 2

You are now going to modify the script you write for Lab Exercise 1D to see if a master file exists and, if it doesn’t, create one. Then, in the loop, you will ignore any non-regular files and the master file.

1. At the beginning of the script, add a line that defines a variable called MASTER, and give it the initial value “MasterList”. Call this script “scan2a.sh”. When you are done, save a copy because you will use this in Lab Exercise 3.
2. After the variable definition but before your loop, test to see if the file named by the value of the variable MASTER exists. If so, print the message that the file “exists; please delete it” and exit, giving exit status code 1. If not, create it. Call this script “scan2b.sh”.
3. In the *for* loop, before you print the file attributes, check to see if the file being examined is either a regular file or the MasterList file. If it is neither, immediately go to the next file (that is, do not get the file attributes). Call this script “scan2c.sh”.
4. Modify your script so that the attributes are stored in the file named by the value of the variable MASTER. Call this script “scan2d.sh”.
5. Finally, if everything works, have the script exit with an exit status code of 0. Call this script “scan2e.sh”.

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 script1.sh. The outputs for the files abc xyz, abcde, demo.for.sh, demofor2.sh, and demofor3sh should be the same. The other files that are present should all be ones you created or put there.

### Lab Exercise 3

This exercise uses the copy of the script you saved after completing Lab Exercise 2A. We will be modifying it in a way similar to the rest of Lab Exercise 2.

1. At the beginning of the script, add a line that defines 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. Call this script “scan3a.sh”. What is the actual name of the file created?
2. In the *for* loop, before you print the file attributes, check to see if the file being examined is the file named by the value of TMP, and if so, *immediately* go to the next file (that is, do not get the file attributes). Call this script “scan3b.sh”.
3. Modify your script so that the attributes are stored in the file named by the value of the variable TMP. Call this script “scan3c.sh”.
4. After the loop, print the message “Changed files:”, and then compare the contents of the files named by TMP and MASTER. Use *diff* to generate the comparison. Call this script “scan3d.sh”. What happens if the file named by the value of MASTER does not exist?
5. Before the loop, check that the master file (the file named by the value of MASTER) exists. If it does not, print an error message saying “Master file does not exist; please generate it” and exit with an exit status code of 1. Call this script “scan3e.sh”.
6. Conclude by using the command *rm* to delete the file named by the value of TMP after the comparison in Part D. Again, have the script exit with an exit status code of 0. Call this script “scan3f.sh”.

When you test this script, the MasterList file must already exist. If it does not, execute the script you wrote in Lab Exercise 2.

You should get at least one line for the file MasterList. You will also get other lines corresponding to the scripts you wrote. As long as you get the line for MasterList, you’re doing it right.

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. That command is your script. So you should see 0.

### Lab Exercise 4

Now combine the scripts you wrote for Lab Exercises 2 and 3. Your script, which you are to call “scan4.sh”, should define the 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”.

### Lab Exercise 5

Now you will modify the script from Lab Exercise 4 to handle two options. First we will handle the option –g, which creates the master file, then -d, which deletes the master file. If the script is called with no arguments, it will generate a list of files the attributes or contents of which have changed since the master list was created.

For the –g option:

1. Create a variable called GENMASTER and, at the beginning of the script, set it to “no”. Call this script “scan5a.sh”.
2. If the –g option is given, set GENMASTER to “yes”. Call this script “scan5b.sh”. What happens if you give some other argument or option, like “-m”?
3. After the argument processing loop, check to see whether GENMASTER is “yes”. If it is, do what you do in the script that was the answer to Lab Exercise 2; you can just copy it into this script if you like, but if you do, don’t copy the line setting the variable MASTER. Then exit with an exit status code of 0. If it is “no”, do what you do in the script that was the answer to Lab Exercise 3; again, you can just copy it into this script if you like, but don’t copy the lines setting the variables MASTER and TMP. Exit with an exit status code of 0. Call this script “scan5c.sh”.

Test your script by running it in the sample directory. First, do not give the –g option; you should get a list of files that have changed, most likely including abcde (from Lab Exercise 4). Then run it again giving the –g option. You should get the error message saying the master file exists, please delete it.

For the –d option:

1. Create a variable called DELMASTER and, at the beginning of the script, set it to “no”. Call this script “scan5d.sh”.
2. If the –d option is given, set DELMASTER to “yes”. If any command-line option (or argument) other than –d or –g is given, print the error message “Unknown option” followed by the option, and exit with an exit status code of 1. Call this script “scan5e.sh”.
3. Before you check the value of the variable GENMASTER, if the value of DELMASTER is “yes”, 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 “Master file does not exist; please generate it” and exit with a status code of 1. Otherwise, if the –d option is not given, continue. Call this script “scan5f.sh”.

Test your script by running it in the sample directory. First, do not give the –d option; you should get a list of files that have changed, most likely including abcde (from Lab Exercise 4). Then run it again giving the –d option. You should get the error message saying the master file exists, please delete it.

Finally, do some sanity checking. It makes no sense to give both the –d and –g options, so we need to give an error message if both are set.

1. Right after you process the arguments (options), check the values of DELMASTER and GENMASTER. If both DELMASTER and GENMASTER are “yes”, print the error message “Only one of –d, -g allowed” and exit with an exit status code of 1. Call this script “scan5g.sh”.

### Puzzler

Add an option –m that takes the argument following it to be the name of the master file and set the variable MASTER accordingly. Be sure to handle the case in which –m is given and no filename follows. Call this script “scanP.sh”.

### Big Puzzler

Modify the script you wrote for the Puzzler so that there does not need to be a space between the –m and the new master file name. That is, if the new name is to be ML, either of these will work:

scanBP –m ML

scanBP –mML

Call this script “scanBP.sh”.

Hint: Look at the pattern matching operation in the command *expr*.

## What to Submit

For the parts of the exercises that do not require you to write a script, put your answers in a PDF or text file numbered appropriately, and call that file “Unit2answers.pdf” or “Unit2answers.txt” respectively. For the parts of the exercises that do require scripts, create plain text files to hold your script, and name the file containing your script as indicated in the problem.