

# SECURE SCRIPTING

## THE BASICS

### LAB: LOOKING FOR STRINGS

In this lab you will build a simple tool to look up words containing a sequence of characters (a *string*). You will build this script in three stages. First, assume that both the string and the filename are given. Next, assume that the string is to be read from the command line. Finally, create a script that will read both the string and the filename from the command line. Check for errors throughout the entire process.

For this lab you will need the following files:

- dict.txt
- mycat.sh
- x
- y
- x y

---

#### LAB EXERCISE 1

Find the words in the word list "dict.txt" that contain the string "gry".

- A. What is the format of the file "dict.txt"? How many words per line does it contain?
- B. Look up the program *grep*. What arguments would you give it to look for the string "gry" in dict.txt?
- C. Write a two-line shell script called "lookfor1.sh" that executes this command to look for words containing "gry" in dict.txt. The first line should specify that the program "/bin/sh" is to be used. The second line should contain your command. Try it out. If you did it right, you will see eight words, the first being "agrypnia" and the last being "pougry".



The image shows a Linux desktop environment with a terminal window open. The terminal title is "landshark@localhost:~ — vi lookfor1.sh". The terminal content shows a shell script being executed, which uses 'grep' to search for 'gry' in 'dict.txt'. Below this, the user attempts to run './lookfor1.sh' but receives a 'Permission denied' error. They then try to change permissions using 'chmod ugo+x lookfor1.sh' and 'chmod u+x lookfor1.sh', both of which fail with 'command not found' errors. Finally, they use 'chmod ugo+x lookfor1.sh' successfully, and the script runs, displaying a list of words containing 'gry'.

```
landshark@localhost:~ — vi lookfor1.sh
#1 /bin/sh
grep gry dict.txt
~
~

Activities Terminal Nov 7 10:21

landshark@localhost:~
04.SeS_Unit1_TheBasics_DataFiles.zip file1 Pictures stats.sh
Desktop hardlink1 Public Templates
Documents lookfor1.sh softlink1 Videos

[landshark@localhost ~]$ ./lookfor1.sh
bash: ./lookfor1.sh: Permission denied
[landshark@localhost ~]$ chmod ugo+x lookfor1.sh
bash: chmod: command not found...
Similar command is: 'kmod'
[landshark@localhost ~]$ chmod u+x lookfor1.sh
bash: chmod: command not found...
Similar command is: 'kmod'
[landshark@localhost ~]$ chmod ugo+x lookfor1.sh
[landshark@localhost ~]$ ./lookfor1.sh
grep: dict.txt: No such file or directory
[landshark@localhost ~]$ ./lookfor1.sh
agrypnia
agrypnode
arthrogryposis
grylle
grypanian
gryph
gryposis
puggry
[landshark@localhost ~]$
```

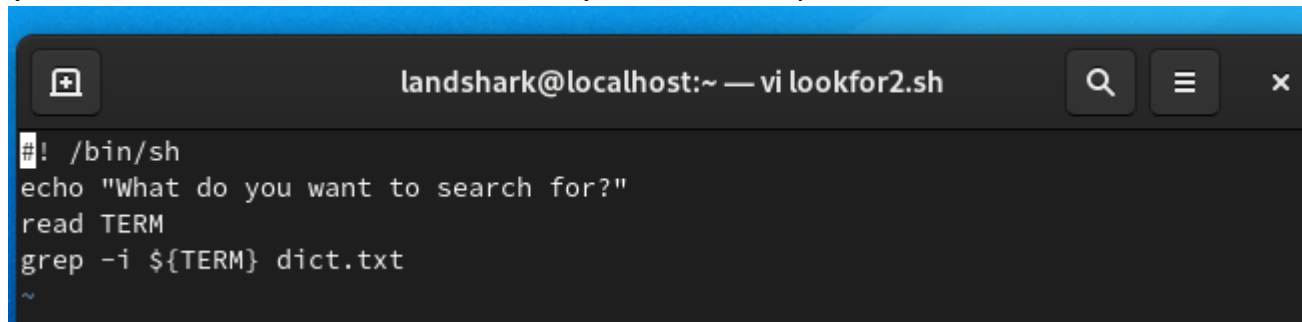


---

## LAB EXERCISE 2

Find the words in the word list "dict.txt" that contain a string supplied by the user. Hint: Begin with your script from Lab Exercise 1 and modify it as indicated.

- A. How do you represent the first argument from the command line in the command you put into your script?
- B. Modify the script you wrote for Lab Exercise 1 to take the string to be searched for from the command line. Test your script by searching for the strings "hello" and "world". Call this script "lookfor2.sh".
- C. What happens if no arguments are given? Two arguments?
- D. How would you embed a blank character in your argument?
- E. What happens if an argument contains a blank?
- F. Modify your script so that an argument with a blank is handled properly (that is, searched for in the file dict.txt). Call this script "lookfor2.sh".



```
landshark@localhost:~ — vi lookfor2.sh
#!/bin/sh
echo "What do you want to search for?"
read TERM
grep -i ${TERM} dict.txt
~
```



```
landshark@localhost:~  
[landshark@localhost ~]$ ./lookfor2.sh  
What do you want to search for?  
hello  
chello  
helloed  
helloes  
phelloplastic  
[landshark@localhost ~]$ ./lookfor2.sh  
What do you want to search for?  
world  
afterworld  
antiworld  
world-abiding  
world-advertised  
world-approved  
world-borne  
world-canvassed  
world-comforting  
world-commended  
world-confounding  
world-corrupting  
world-corrupting  
world-despising  
world-despising
```

```
landshark@localhost:~  
[landshark@localhost ~]$ ./lookfor2.sh  
What do you want to search for?  
hello  
chello  
helloed  
helloes  
phelloplastic  
[landshark@localhost ~]$ ./lookfor2.sh  
What do you want to search for?  
hello  
chello  
helloed  
helloes  
phelloplastic  
[landshark@localhost ~]$ ./lookfor2.sh  
What do you want to search for?  
        heLLo  
chello  
helloed  
helloes  
phelloplastic  
[landshark@localhost ~]$
```

---

### LAB EXERCISE 3

Modify the script you wrote for Lab Exercise 2 by adding an “if” statement that checks whether there is exactly one argument. Call this script “lookfor3.sh”.

If there is *not* exactly one argument, your script should print the error message “Usage: give exactly 1 argument, the string to be looked for” and exit immediately.



```
Activities Terminal Nov 8 13:04
landshark@localhost:~
[landshark@localhost ~]$ ./lookfor3.sh
What do you want to search for?
hello world
grep: world: No such file or directory
dict.txt:chello
dict.txt:helloed
dict.txt:helloes
dict.txt:phelloplastic
Usage: Give exactly one argument, the string to be looked for
[landshark@localhost ~]$
```

#### LAB EXERCISE 4

Find the words in the word list named by the user (like "dict.txt") that contain a string supplied by the user. Your shell script is to be called "lookfor4.sh".

Hint: Begin with your script from Lab Exercise 2 and modify it as indicated.

- A. Modify your script in Exercise 2 so that the word list is the second argument on the command line. Call this script "lookfor4.sh". Remember to change your "if" statement so it balks if there are not exactly two arguments!
- B. Change the error message to "Usage: lookfor4 string file". Call this script "lookfor4a.sh".
- C. Change the error message so it prints the exact name of the script. That is, if you call the script "find4" and not "lookfor4.sh", the error message should print as "Usage: find4 string file". A similar error message should occur if you call the script "catdog" *without changing the script!* Call this script "lookfor4b.sh".



```
landshark@localhost:~ — vi lookfor4.sh

#!/bin/sh
echo "Enter string to search for and file to search seperated by space."
read TERM FILE
grep -i ${TERM} ${FILE}
if [ $# -eq 2 ] ;
then
echo "Usage: You must list exactly two arguments."
fi
```

---

## PUZZLER

In the script you wrote for Lab Exercise 4, if the file does not exist, *grep* prints an error message. This will be confusing to beginners. Add a test at the start of that script that prints the error message

```
script_name: file file_name cannot be read
```

where *script\_name* is the name of the script and *file\_name* is the name of the file that the user gives. Call this script "lookforp1.sh".

---

## BIG PUZZLER

This is for all you Linux experts. It demonstrates that there are several ways to write a script.

- A. Rewrite the script you wrote in Lab Exercise 4 using the program *sed* rather than *grep*. Call this script "lookforp2.sh".
- B. Rewrite the script you wrote in Lab Exercise 4 using the program *awk* rather than *grep*. Call this script "lookforp3.sh".

## 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 "Unit1answers.pdf" or "Unit1answers.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 instructions.

