Elec. 377 - Operating Systems
Lab 4 - Shell Scripting

Lab Date: Nov 5, 2012, Due: Nov. 15, 2012

Objectives

- Learn shell scripting and its application in collecting system information

Introduction

In this lab you will write two scripts:

1. One that provides system information and provides some of the information that the *ps* command does
2. The other does a simple analysis of the source code written in the previous labs.

In the lab you will be exposed to several features of shell scripts which will enable you to write more complex scripts in the future.

Script 1: System Information

As you may recall from lab1, the *ps* command does not use a system call to find process information, but instead searches through the */proc* directory to find the information. Recall that each process has a directory, the name of which is the pid of the process. Information about each process is available there. For example, information about the *init* process, with a process id of 1, is available in the directory */proc/1*. You will write a script that traverses the proc directory and produce the same output at the command **'ps -eo pid,user,rss,args'**. For example, on one of my research machines, I get the following output:

```
 PID USER       RSS COMMAND
    1 root      1868 /sbin/init
    2 root         0 [kthreadd]
...
14787 stephan       968 ps -eo pid,user,rss,args
...
```

In this output, the RSS field gives the list of resident pages (i.e. Resident Set Size). Some process entries actually refer to a kernel process that runs inside the kernel space, so they do not have a separate page tables and thus have a RSS of 0. This information is found in the *status* file in the process directory (on the line starting with VmRSS:).

The user name can be found in one of two ways, by using the numeric user id in the status file and then looking up the user name in the file */etc/passwd*, or by checking the owner of the process directory. The command line of the program is in the cmdline file. However arguments are separated by the null('\0') character. You will have to change the null characters to spaces.

To solve this problem you will have to use the *for* statement along with patterns for directory names. Some of the commands you should use in this assignment are *stat, tr, awk, echo,* and *sed*. Use the *man* command to find out more about these commands. In particular, the *tr* command reads only from stdin, so you must use I/O redirection if you want to read from a file. Use the *awk* command to get the final output formatted correctly.

Testing will be similar to the first lab: Compare the output of your script with the *ps* command and explain any differences.

Script 2: Source Code Information

Write a script that takes a path as an argument. It should go recursively through all the subfolders of that path and produce the following information on source files:

1. For each source file that contains a main function, display the full path of the file; and append the number of "printfs" and the number of "fprintfs" occurrences
2. For each module file (i.e. a file that contains init_module), list the full path of the file and append the lines where "printks" occur.
3. For any other source file (i.e. containing neither main nor init_module), just list the full path of the file.

If no main file is found, the script must display "No main file"; if no module file, it displays "No module file", and if no other source file, it displays "No other file"

Here is an example of execution of such a script:
[judi@F16-ROAMING-BEAST lab4]$ ./sc2 ../../
Main Files:
/home/judi/TA/myNetId/lab1/lab1.c:0,0
/home/judi/TA/myNetId/lab3/consumer.c:2,0
/home/judi/TA/myNetId/lab3/meminit.c:3,0
/home/judi/TA/myNetId/lab3/producer.c:2,0

/home/judi/TA/myNetId/my_code/lab3/consumer.c:4,0
/home/judi/TA/myNetId/my_code/lab3/meminit.c:3,0
/home/judi/TA/myNetId/my_code/lab3/producer.c:3,0
Modules Files:
/home/judi/TA/myNetId/my_code/lab2/lab2.c:77,80,100
Other Files:
/home/judi/TA/myNetId/lab2/lab2.c
/home/judi/TA/myNetId/lab3/common.c
/home/judi/TA/myNetId/my_code/lab3/common.c

In this example, ../../ is the path passed to the script as argument.
The script found the main file /home/judi/TA/myNetId/lab1/lab1.c with 0 printf and 0 fprint;
/home/judi/TA/myNetId/lab3/meminit.c with 3 printfs and 0 fprintf, etc.
The script also found the module source /home/judi/TA/myNetId/my_code/lab2/lab2.c with printk on lines 77, 80 and 100
etc.

Here is another execution where none of the categories of files exist:
[judi@F16-ROAMING-BEAST lab4]$ ./sc2 ./
No main file
No module file
No other File

For this second script, commands like *grep* and *find* should come in handy. Once again the *for* statement can be useful.There is no oracle for this script. Produce the output of your script with various paths as arguments; comment on the folder contents and the output of your script.

What to check in.

Create a lab4 directory in your repository. Use svn add to add both the directory and any files in it that you create as part of the lab. Check in the shell scripts, documentation, testing documentation and the all of the output files referred to in your testing documentation.