ΜΥΥ106 - Introduction to Computer Science
6th Lab

Users - Permissions - Processes

In this lab we will deal with process management, user environment configuration, and file permissions.

Create the directory `lab6` inside the directory `myy106` located in your home directory. Using the `vi` text editor, create the answer file `answers-lab6.txt`.

Using `wget`, download the files `sum.py` and `sum.c` to the `lab6` directory. The files are located at `https://www.cse.uoi.gr/~myy106/files/` (you should fill in the file name at the end of the url).

Measuring process time

In Worksheet #2 we had seen in detail the steps of translation (`compile`), symbol translation (`assemble`) and linking (`linking`) required to execute a program written in the `C` programming language. In each step you used the `gcc` command with different parameters and a file was created as output, namely the assembly language file during compilation and the object file during assembly. In today's worksheet you will use the `gcc` command only once and directly create the final executable file.

1. Run the command `gcc sum.c -o mysum` to compile the code of the file `sum.c` and create the executable file named `mysum`, as specified by the parameter `-o`. Then run the file `mysum` by issuing the command `./mysum`.
2. Run the command `python3 sum.py` to run the same program written in `Python`. What do you observe? Record your response in the answers file.
3. For the `C` program, use the command `time` to first measure the compilation time of the file `sum.c` and then the execution time of the program `mysum`.
4. For the `Python` program, use the command `time` to measure the execution of the program `sum.py`.
5. Why did you measure the time twice for the `C` program, but only once for the `Python` program? How does this reason relate to the fact that the `Python` program is much slower than the `C` program? Record your response in the answers file.

Environment Variables

6. To run the file `mysum` you must specify its relative or absolute path. Rerun the file `mysum` located in the current directory, using its relative path. Record the response in the answers file.
7. Move to the parent directory of `lab6` and run the file `mysum` using its new relative path. Record the response in the answers file.
8. Update the environment variable `PATH` appropriately so that you can run the file `mysum` from any directory on the file system by simply giving the name `mysum` (without the absolute or relative path). Record the response in the answers file.
9. Move to your home directory, using the `cd ~` command, and execute the file `mysum` from there, using the `mysum` command, to make sure that the `PATH` has been updated correctly and you can now execute the file from any directory.
10. Open a new terminal and run the command `mysum`. What do you notice? Record your response in the answers file.
11. Consider what you need to do to make the update of the environment variable `PATH` permanent. Before implementing the solution, ask for help to make sure it is correct. Before implementing it, open a new terminal, apply the solution, and test whether the command `mysum` can now be executed.
12. Open a new terminal and run the command `mysum`. What do you observe? Record the response in the answers file.

Permissions

13. Use `ls -l` to see the permissions for the files `sum.c` and `mysum`. Do you notice any difference between the two files? Record the response in the answers file.

14. Use the appropriate command with symbolic syntax to remove the execute permission from the file `mysum`. View the updated permissions with `ls -l`. Try running `mysum`. What do you notice? Record the response in the answers file.

15. Use the same command with numeric syntax this time, to restore execute permission for the owner only. View the updated permissions with `ls -l`. Try running `mysum` again.

16. Remove read permission from everyone for the file `sum.py` (using symbolic or numeric syntax). View the updated permissions with `ls -l`. Display the contents of the file `sum.py` on the screen. What do you notice?

17. The write permission for the owner of the file `sum.py` remains as you can verify with `ls -l`. Try opening the file `sum.py` with the text editor `vi` to modify the file. What do you observe? Record the response in the answers file.

18. Add a line to the end of the file `sum.py` with the command `echo "print(\"Done\")" >> sum.py` (be careful to use `>>` so as not to delete the contents already in the file). Then give read permission to the owner of the file `sum.py` and try again to display the contents of the file `sum.py` on the screen. What do you observe? Record the response in the answers file.

19. Move to the parent directory of `myy106` and view the permissions of the directory `lab6`. Remove all permissions. Use the ls command to view the contents of the `lab6` directory. Use the `cd` command to move into the `lab6` directory. What do you observe? Record the response in the answers file.

20. Restore the original permissions to the directory `lab6` using numeric syntax.

21. Display the contents of the directory `lab6` in detail to see the permissions of all the files it contains. What do you notice? Record the response in the answers file.

22. Remove all permissions from the file `mysum` without moving to the directory `lab6`.

23. While in the directory `myy106`, copy (using the appropriate parameter) the directory `lab6` and all its contents to a new directory named `newlab6`. What do you notice? Record the response in the answers file.

24. Give read permission to the owner of the file `mysum`, and repeat the copy of the directory `lab6` to `newlab6`. Display the contents of the directory `lab6` in detail, and then display the contents of the directory `newlab6` in detail. What do you observe? Record the response in the answers file.

25. Repeat the copy of the directory `lab6` to a new directory named `correctlab6`, this time using the `-p` parameter when running `cp`. Display the contents of the directory `correctlab6` in detail.

Process Management

26. Run the Python program `sum.py` and after 1-2 seconds, temporarily suspend its execution with the shortcut `Ctrl-z`. The word "`Suspended`" should appear on the screen.

27. Run the command `ps -o pid,user,stat,etime,comm` to display a snapshot of your processes currently running on the system using the format described by the `-o` parameter. Locate the process for the Python program you interrupted. Note the `PID` field, which refers to the process identifier, and the `STAT` field, which refers to the status of each process. Also, identify the process for the shell that executes the commands you issue. What is the status symbol of the two processes and why is it different? Record the response in the answers file.

28. Use the command that displays all the processes whose execution we have interrupted, and resume the execution of the process for the Python program, sending the execution to the foreground, and then interrupt the execution again before it has time to complete.

29. Run the same Python program again, but this time in the background using the appropriate symbol. Before the execution ends, issue the command `jobs` several times and continue to execute it shortly after the Python program has finished. What do you notice in the state of the process? Record the response in the answers file.

30. Run the command `ps -o pid,user,stat,etime,comm` again. What do you notice about the `ELAPSED` field for the Python program process that we have stopped. Record the response in the answers file.

31. Use the appropriate command again to see the process ID of the process we have interrupted, and continue its execution in the background until it completes.

32. Rerun the Python program and immediately suspend its execution. Verify that its status is `Suspended` and find the process ID with the appropriate command. End the suspended process gracefully by using the appropriate command to send the termination signal `SIGTERM`.

System information

33. Run the appropriate command to display information about disk usage and the limit available for your account. Use the appropriate parameter to display the values in a human-readable format.

34. Notice the values of the space, quota, and limit fields. Space refers to the total space occupied by the files in your account, quota is the soft limit that you will receive a warning if you reach that your account has almost used up all of its storage, and limit is the hard limit that you cannot exceed. and if you reach it you will not be able to write new files to the disk. You should frequently check the space your files are taking up on the disk to delete any that are no longer needed.

35. Use the appropriate command to display on the screen all directories and subdirectories along with their size that are in your home directory. Use the appropriate parameter to display the values in human-readable format.

36. Use the same command in combination with `grep` to create a chain of commands (using pipes) and display on the screen only the directories whose size is a few MegaBytes so that you can easily identify which directories consume the most disk space. Use the appropriate regular expression for grep to show you results that start with one or more digits and end with the letter M.

Reset the PATH environment variable

37. Before completing the worksheet, close all windows and log out of your account. Then log back into your account and open the terminal. While in your home directory, run the command `mysum` without giving the absolute or relative path of the executable. The execution should be successful this time as the update of the `PATH` variable made in query (8) is now permanent due to the logout from the account.

38. You can now restore the `.profile` file to its original form by removing the absolute path to the `mysum` executable from `PATH`.

As we have seen, for an executable to be located from the command line, it must reside in a directory included in the `PATH`. The system executables are located in the directories `/bin, /sbin, /usr/bin, /usr/local/bin`, as we can see with the command `echo $PATH`.

Our own executables are not located there, so we run them using their absolute or relative path. However, we can also add our own directories to the `PATH` so that the shell can locate them.

It is common practice to create a `bin` directory in our home directory, place our executables there, and add `$HOME/bin` to the `PATH` (inside the `.profile`), so that they are executed just like the system commands.

Submit your answers: https://forms.office.com/e/HNRXQJmv0n