ΜΥΥ106 - Introduction to Computer Science
3rd lab

Reflection

As we have seen so far, the commands we give in the terminal resemble spoken instructions we would normally give to a person – only in this case, the syntax is stricter and involves some kind of encoding. For example, if we asked a person to copy a file named `file1` from the directory `dir1` to the directory `dir2` under the name `file2`, the sentence would have the form:

```
copy the file "file1" from the directory "dir1" to the directory "dir2"
     with a new name "file2"
```

And if we wanted to be polite, we would start with the word `Please`. However, since we are addressing a computer, where the syntax is strict and the structure of the command is predefined, the corresponding command in the terminal will have the form:

```
cp dir1/file1 dir2/file2
```

In any case, to understand the correct syntax for a command in the terminal, it's helpful to start by thinking about how you would phrase the corresponding sentence in plain English. Then, simply express the arguments and parameters of the command in the way the terminal expects to read them.

Some commands are given without arguments (e.g. `ls`, `pwd`, ...).
Other commands require some arguments (e.g. `mv file1 dir2/`), which must be given in a very specific way.
To understand what arguments each command expects, it is usually enough to use common sense. For example, when we want to move a file to a directory, it is logical to tell the computer:

α′. which file we want to move, and
β′. to which directory we want to put it.

If we omit any of the arguments, an error message will be displayed in the terminal.

Introduction

In this lab, we will use terminal commands to create files and manage their contents. This time, however, we will not open the `Files` application at all, nor the `pluma` text editor for the answer file. We will try to work exclusively with the terminal commands we have learned so far.

Open a terminal (application `terminal` from the menu) and if you have not done so in the previous labs, create the course directory `myy106` in your personal directory (`mkdir` command). Inside it, you will create a new directory named `lab3`. Go to this directory (`cd` command) and create the file `answers-lab3.txt` (`touch` command)

Creating text files

1. Using the `echo` command, display your name and registration number (or the username of your temporary account) in the terminal.
2. Now issue the same command, but this time redirect the output (i.e. `>`) so that the result of `echo` is saved in the file `answers-lab3.txt`.
3. Using the `cat` command, display the contents of the file `answers-lab3.txt` in your terminal.
4. Using the `echo` command and the output redirection `>` write the word **goodmorning** to the response file. Do `cat` again to see the contents of the file.

Oops! The contents of the file were lost!

With the single output redirection > the previous contents of the file are deleted. However, our intention was to add the word **goodmorning** to the end of the file, while also keeping our first and last name. To achieve this we should have used the double output redirection >>

We need to be very careful when saving text to a file. If we don't want to lose its contents, we should use the double output redirection >> and not the single one >

5. By pressing the up arrow key, reissue the previous commands to create the response file from scratch. This time, however, you will use the double output redirection >> to write **goodmorning** at the end of the file.

As there is always the possibility of forgetting and using the wrong redirect, it's a good idea to periodically backup your answer file. That way, if something goes wrong or the file gets lost, you'll have a recent copy available.
To create such a copy, each time we add a new line to the end of the response file, we can also issue the command:

```
cat answers-lab3.txt > answers-lab3.bak
```

This way we will always keep a backup copy which we can easily restore if we accidentally delete the file.

Many times we will need to execute a command that we have already given earlier. However, it is not necessary to type it again from the beginning. By pressing the up arrow we can browse the command history and find those we have used previously. In this way we save time, avoid unnecessary typing and increase our productivity.

In the following activities, after you have found the command that answers each question, press the up arrow to display the command again. Then, add the command `echo` in the beginning and the redirection `>>` to the answer file at the end.

For example, if the answer to a question is `ls -l`, then with the command:

```
echo ls -l >> answers-lab3.txt
```

you will record `ls -l` at the end of the response file.

Whenever you need to record the character `>` in the response file, you should write it as `\>`. For example, the command:

```
cat > joke1
```

should be recorded as:

```
echo cat \> joke1 >> answers-lab3.txt
```

6. Using the `cat` command and output redirection, write (or copy) the following text to a file named `joke1` in the `lab3` directory.

```
Q: How many programmers does it take to change a light bulb?
A: None, that's a hardware problem!
```

(to indicate the end of the input type `Ctrl+d`)

7. Using the appropriate output redirection and, perhaps, several commands, create a file named `joke10` that contains the contents of `joke1` 10 times in a row.

Editing text files

8. Using the `cat` command and redirecting output, create a file named `file.txt`, with the following contents:
```
costas alekou 11 20
alekos konstantinou 20 2
giorgos georgiou 11 9
maria papadopoulou 23 4
panos panagiotou 21 5
katerina alekou 29 11
vaggelis argiris 16 1
costas manolis 12 10
```

9. Using the redirection `>>` add the following additional lines to the above file:
```
giannis ioannou 5 4
alekos panou 28 11
markos markou 12 1
```
and check with the `cat` command that everything was written correctly.

10. Using the commands `cat`, `grep`, the pipe `|` and the output redirection `>`, create a file `new.txt` containing only those lines of the file `file.txt` that have the word `aleko`. Check with the command `cat` that the correct lines were written.

11. Add the lines containing the number `11` to the end of the file `new.txt`.
Hint: use the `grep` command again with output redirection `>>`.
Check with the `cat` command that the correct lines were written.

12. Sort the lines in the file `new.txt` with the command `sort` and write the sorted result to the file `new-sorted.txt`. Check that it is sorted correctly.

13. Use the `uniq` command to display the repeated lines in the file `new-sorted.txt` in a collapsed form, entering the number of repetitions in front of each line.
Hint: Requires a special parameter. See the details of `uniq` with the `man` command or the `--help` parameter.

14. Enter the following commands:
```
cut -f 3 -d " " file.txt > a.txt
cut -f 4 -d " " file.txt > b.txt
```
and view (using `cat`) the contents of the files `a.txt` and `b.txt` that are produced. What is the function of the `cut` command? Comment on the output of the commands and record your answer in the response file (with `echo` and `>>`).

Viewing text files

15. As we have seen before, our personal directory is denoted by $\sim$ while the directory above the current one is denoted by `..`

    Combining these notations, we can go to $\sim$`/../` which is the directory above our home directory, that is, the directory that contains the home directories of all the students in our year.

    α'. Without leaving the directory you are in ($\sim$`/myy106/lab3`) issue the command `ls -lt` $\sim$`/../` (the `-t` parameter sorts by date) to see which users were most recently logged into their account. You will notice that the output of the command does not fit on the screen.

    Issue the same command again, but this time with output redirection, to save the output to the file `users.txt`

    β'. Find the 5 users who have not logged into their account for the longest time (`tail` command).
    Hint: You can use the file `users.txt` or combine the `ls` command and the `tail` command directly, using the | pipe.

    γ'. Find the 5 users who logged into their account most recently (command `head`).

16. In the `/proc` directory (there is a directory named `proc` located in the root directory `/`) there are files that contain information about our system.

    α'. We are interested in finding the total RAM memory of our computer (`MemTotal`). Use the command `cat` to view the contents of the file `/proc/meminfo` (you will notice that once again, the contents do not fit on the screen).

    View the contents of the file again, this time using the `more` command (the contents will appear gradually, page by page). Now it is easier to locate the information we are interested in.

β′. Suppose you want to keep only the first few lines of the file `/proc/meminfo` (with the command `head`). What is the minimum number of lines you need to keep to display the total RAM? Note the number of lines in the response file.

γ′. Using the `less` command, view the contents of the `/proc/cpuinfo` file that has information about the processor. First, navigate up and down line by line or page by page to view all the contents of the file.

Then search for the term `processor` and count how many processors your system has.

Does it make sense to have so many processors?

One of the main characteristics of a processor is the number of `cores` it has. A modern processor may have 4 or more cores, and each core can execute 2 processes simultaneously. In this case, we say that the processor has 4 `cores` and 8 `threads`.

The operating system recognizes each of these processing units as independent, which is why it displays multiple "`processors`". However, they all correspond to the same physical processor in your computer.

Before proceeding to the last activity, first make sure you have backed up the answer file.

17. Use the commands `head` and `tail` (and any others you need) to remove the word **goodmorning** from the response file. Specifically:

α′. Locate which line the word goodmorning is on (with one of `cat`, `more`, `less`, ...)

β′. Isolate the lines that precede (with `head`) and save them (with an output redirection) in a new file named `lines-before.txt`

γ′. Isolate the following lines (with `tail`) and save them (with an output redirection) in a new file named `lines-after.txt`

Hint: You will need to count the lines of the response file (using `wc`), and remove any lines you don't need from the top.

δ′. Merge the files `lines-before.txt` and `lines-after.txt` into a new file named `answers-lab3-new.txt` and then record the commands you gave during this activity in it. This is the file you will submit.

Submit Answers: Click on the link below to open the form to submit your answers. **Copy** all the contents of the file `answers-lab3-new.txt` and **paste** them into the corresponding box (**Ctrl+A** to select all).

https://forms.office.com/e/HNRXQJmv0n

Connecting from home (complete example)

As you may have noticed, the time in the lab is not sufficient to complete the worksheet. So you will have to finish it by yourself, either at some other time in the labs, or from home by connecting to **scylla**.

This requires a little attention. Keep in mind that:

- **scylla** is not a lab computer. It's a server. You cannot run there every command you have learned. It doesn't even have the same operating system as the lab computers.
  (As a homework, use the command **uname -a** to find which operating system is running on this server.)
- Your lab account does not have administrator rights. It is a limited user. This means that you cannot change system settings or install any applications. You can only make changes to your own user.
- If you read the welcome message when you connect to **scylla**, it informs you that you need to **ssh** to one of another machine to work on the lab computers.
- When you connect from home, there will no graphic interface available. You can use the terminal only. Therefore, you cannot open, for example, **pluma** or any other window.
  (Don't worry, once you get familiar enough with the terminal, this won't be a problem.)

Based on the above, to work from home:

- In your terminal or PowerShell (if you have Windows) enter the command
  `ssh cs221234@scylla.cse.uoi.gr` (with your username)
  to connect to **scylla**
- Afterwards, you will **ssh** to another computer (you can see which ones are online using the **rupt** command).
- Once you log in to the lab computers, you can work normally. You can enter any command you want, as long as it doesn't require a graphic interface.
- To edit a file, you can use the text editor **vi**. However, until you become familiar with it, you can use the commands **cat** or **echo** with output redirection, as we saw in today's worksheet (e.g. **echo cmd > answers.txt**).
- When you are finished and want to log out, enter the command **exit** (or **Ctrl+D**).