Tobias Neckel Max-Planck, March 2019

Bash course - Tutorial 2

Working with `screen`

In this task, we are going to prepare process managament and use `screen` with different processes.

a) Prepare the usage of the `tail` command by downloading and unzipping `http://www5.in.tum.de/lehre/vorlesungen/progcourse/ss19/Bash/bash_day2_examples.zip` in a suitable location.

b) Run the script `tail_test.sh` and put it to the background using `Ctrl-z` and afterwards `bg`. Use `ps` to identify the process ID of the (still running) process associated with the script `tail_test.sh`. Kill this process.

c) Run the steps of task **b)** without killing the process. Then, use the command `tail -f temp.txt` as on the lecture slides to show the content (last 10 lines) of the file `temp.txt`. You may now see a changing output until Iteration 30.

d) Use `screen` to create a first virtual terminal. Run the steps of task **c)**. Detach the screen.

e) Use `screen` to create a second virtual terminal. Start `top` in this terminal and detach the screen. Now, you have two virtual screens (in addition to your actual terminal).

f) Use `screen -r` in a suitable way to recover your desired virtual terminal (the first or the second). Play around by going back and forth; if desired, try to give the virtual terminals intuitive names. Stop the corresponding processes and kill the virtual terminals. Now, `screen -r` should not be possible any more.

In case you have access to a remote machine, you might want to try something similar on that machine involving login/logout steps.

Working with `find`, `grep`, and regular expressions

Hint: You may want to pipe the result of your find/grep results into `wc` to count the number of lines.

a) Download and unzip `http://www5.in.tum.de/lehre/vorlesungen/progcourse/ss19/Bash/story_flatFolders.zip` in a suitable location. The extracted folder has two subfolders which separately contain text files with either one or two digits in their name (e.g., `file2.txt` and `file16.txt`).

b) Use the `find` command to find all files with two characters or digits directly after the character sequence "file" in their name. In particular, the file `dummy_fileXX.txt` shall appear in the list of found files.

c) Use the `find` command together with the options `-regextype sed -regex` and a corresponding regular expression in single quotes to identify files with exactly two digits (and no characters) in their name. In particular, the file `dummy_fileXX.txt` shall NOT appear in the list of found files.
Modify the regular expression to again find files with exactly two digits or exactly two characters in their name.

d) Use the `grep` command together with the option `-rn` (recursive and showing the line number) and a regular expression in normal quotes to find all occurrences (i.e., lines) of a digit in the text files. The file `test.txt` contains such lines.
Modify your regular expression to select occurrences (lines) with two digits separated by something and afterwards separated by a single dot.

e) Use the `grep` command together with the option `-rn` and a regular expression to find valid email addresses. The file `email_addresses.txt` contains such lines.

File renamer

Sometimes, large numbers of files have to be renamed, e.g. by adding a prefix, appending a suffix or replacing a certain pattern. In this task, you should write a program doing that automatically. The program should have three different modes:

- `./file_renamer prefix <prefix> <file>+`
- `./file_renamer suffix <prefix> <file>+`
- `./file_renamer replace <pattern> <replacement> <file>+`

a) If the prefix mode is executed, the given prefix should be added to all filenames.

b) If a file has no '.', the suffix should just be appended to the filename. But if there is a '.' in the filename, the suffix should be appended before the dot. So 'name.txt' should become 'namesuffix.txt' and 'name' should become 'namesuffix'.

c) The replace-mode just applies the given pattern.

Hint: Before modifying real files, test your program by just outputting text on the command line!

Hint 2: You may use the following command

```
sed s/$pattern/$rep/g
```

which calls the stream editor `sed` to find text matching `pattern` and replace the found pattern with `rep` while keeping all the rest as is.