System Software
Linux command line manual: lab 3
2016 - 2017

Bachelor Electronics/ICT

Course coördinator: *Luc Vandeurzen*
Lab coaches: *Floris De Feyter*
Stef Desmet
Tim Stas
Jeroen Van Aken
Luc Vandeurzen

Last update: January 26, 2017

Lab targets: find program locations, search files, find patterns in files, use wildcards and build regular expressions

Program location

There are a number of commands that are helpful in finding the location of commands and programs:

- locate : find files by name by looking in an index database (this database might not contain the most updated information);

- which : find a file but only looks in the user's search path; the user's search path is stored in the environment variable PATH and can be printed with 'echo $PATH'.

  Note: Remember the 'echo' command from the previous lab. You can use 'echo' to print out the values of so called shell and environment variables. You can define and set shell variables very easily as follows: <variable>=<value>. For example:

  ```
  x='hello world!'

  y=13
  ```

  Use '$' to obtain the value of a shell variable, e.g. $x. Printing out then the values with 'echo' is easy, e.g.:

  ```
  echo $x $y
  ```

- whereis : locate the binary, source, and man pages for a command in a list of standard Linux places.

Exercise

Where exactly is the 'pwd' executable located?

- First define a shell variable 'p' containing the search pattern 'pwd';
- Use 'which' on 'p' to find the executable.
- Use 'whereis' on 'p' to find the executable. Do you get the same result as with 'which'?
- Use 'locate' on 'p' to find the executable. You will notice that 'locate' is actually performing a search with wildcards: '*pwd*'. Find out how to prevent 'locate' using these wildcards.

Exercise

'passwd' is a text file containing all user info. Use 'cat' to view its content, but first you need to find its location.

- Try 'cat' on the result of 'which'. Try to understand the result.
- Try 'cat' on the result of 'whereis'.

Try to understand the different results of these commands.

Search with find

'Find' is a powerful tool for finding files. The most common use for finding files is the command:

- find <search_path> -name <file_to_search>

Many other options are possible. Just a few are illustrated in the following examples:

- find <search_path> -size +5000K : find files with file size at least 5000KB;
- find <search_path> -perm -a+r : find files that are readable by everyone (a = all);
- find <search_path> -type d : find file of type directory (d = directory, f = regular file, s =socket, p = pipe, l= symbolic link, etc.)

The file name that needs to be searched for, can be defined with 'wildcards', e.g. 'main*.c'.

Remark: if you don't run the 'find' command as root, 'find' will display an error message for each directory on which you don't have read permission. This may generate a lot of data on your screen such that you might overlook the effective file matches. Of course, you could run the 'find' command with root permissions ('sudo') or, if that's not possible, discard all error messages using redirection of 'stderr' as follows:

find <find-options> 2>/dev/null

Exercise: *find*

Most self-respecting Linux distros contain a dictionary, which is just a readable file containing words on every line. Typically this dictionary is accessed using 'words'. 'words' is not really a file but a link to a file containing the real dictionary word list, like the files 'american-english' or 'british-english'. Where are these files located? And is 'words' linked to American or British English?
Hint: limit the search to '/usr/share' to get a quick response.

Search with grep

'Grep' is a powerful tool for searching a file or files for lines matching the given pattern. We limit our explanation to the basic usage of 'grep -E' or 'egrep' which is:

- egrep “<regex>” <file(s)> : with <regex> a regular expression defining what you are searching for in the input files <file(s)>.

Regular expressions will be explored in more detail in the following topic. For the time being, we only work with strings that must 'literally' appear in the file – no special characters will be used yet. Egrep can also search in input from 'stdin' or a pipe. A few examples clarify the usage of 'egrep':

- egrep “if” main.c : find all lines containing 'if' in 'main.c';

- egrep “if” *.c : find all lines containing 'if' in .c files in the current directory';
- cat *.c | egrep “if” : similar to the above (but without a file indication)

Several options can be added to 'egrep' to further enhance its power. Again, we illustrate this with a number of examples:

- egrep -i “if” *.c : ignore case sensitivity;
- egrep -v “if” *.c : show lines that **don't** match;
- egrep -n “if” *.c : print the line number of where matches where found;

Of course, more details can be found in the man pages.

Exercise: *egrep*

Is the correct spelling 'miscalanous' or 'miscelanous' or 'miscellaneous' or …? Use 'egrep' to search in the 'words' file (see previous exercise) to find out the correct spelling.

Important remark

The discussion on regular expressions is not considered as mandatory. First do your C programming exercises and only when you still have time start studying regular expressions.

Regular expressions (regex)

Wildcards are a start in defining search patterns, but regular expressions go a lot further than wildcards and are much more powerful. Without having the intention to be complete, we explain here the basics of regular expressions.
Regular expressions contain 'literals' and 'special' characters.
Literals exactly specify the characters that must be matched character-for-character.
The following special characters of a regular expression have a similar meaning as in wildcards:

- \ : escape character;
- [ ] : match any character specified in [ ];
  - - : is used to denote a range;

But other special characters are new or have a different meaning as in wildcards:

- [^] : match what's not listed in [ ] (with wildcards: [!]);
- ^ : denotes an anchor stating that a match must be at the beginning of the line;
- $: denotes an anchor stating that a match must be at the end of the line;
- . : matches any single character, except end of line (with wildcards: ?);
- + : matches *one or more* of the preceding building block (at least once);
- * : matches *zero or more* of the preceding building block;
- ? : matches *zero or one* of the preceding building block;
- {x,y} : matches *x* to *y* occurrences of the preceding building block;
- {x} : matches *exactly x* occurrences of the preceding building block;
- {x,} : matches *x or more* occurrences of the preceding building block;

- {,y} : matches *y or less* occurrences of the preceding building block;
- ex1 | ex2 : matches regular expression 'ex1' or 'ex2'.

A few example will make the above much more understandable:

- ^[A-Z] : matches every line in a file starting with a capital;
- [+-]?[0-9]+ : matches integers, with or without a sign;
- main.\.c : matches all files like main1.c, mainb.c, main+.c, etc;
- 016-[0-9]{6} : matches phone numbers starting with '016-' followed by 6 digits.

More details can be found in the man pages using 'man 7 regex'.

Remark: The commands 'find' and 'locate' can also be used with regular expressions. Use the option '--regex' in order to do so.

Summary: list of commands

- egrep
- find
- grep
- locate
- whereis
- which