Taking the headaches out of creating regular expressions

Shell Wizard

For geeks, regular expressions might be an interesting mental exercise, but the jungle of arcane abbreviations and symbols can be terrifying for newbies. Enter the txt2regex editor.

BY ANDREAS KNEIB

Rudolf Blum, visipix.com

Trying to find the right regular expression to return a specific string is a familiarly frustrating experience for most readers who are not regular expression gurus. As is the resignation that slowly sets in when you discover that your **regex** would work in the Perl programming language, but not in the Lisp code you just happen to be working on.

Enter the regex wizard, txt2regex, which we will be investigating in this article. The wizard processes text patterns for a variety of programs and languages, from awk, through Emacs, to Perl, procmail, sed, and vim. The txt2regex program comprises mainly of a shell script that requires a Bash version 2.04 or later.

Although nearly any state of the art Linux distro should satisfy this requirement, you might just like to test this to be on the safe side. Type *echo $BASH_VERSION*, or *bash --version* to do so. Assuming that your version number is high enough, you can carry on with the install.

If your distribution does not include the wizard, check out the project download page at [1] for a tar archive. Debian users can simply enter *apt-get install txt2regex* to install the tool.

After completing the download, first unpack the archive and change to the directory created by doing so:

```
~ > tar xvzf ➥
txt2regex-0.7.tgz
~ > cd txt2regex-0.7
```

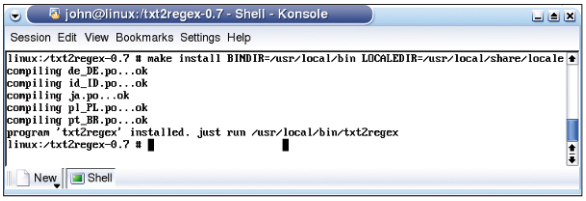

Figure 1: Passing targets while making.

As you do not need to build the program, you can simply enter: *make install* as *root*:

```
~ > su
Password: (type root➥
password)
root# make install
```

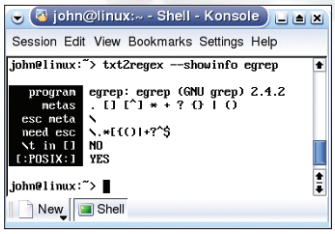

Figure 2: Displaying meta-characters.

However, there is a disadvantage to this approach as it stores the program components in the */usr/bin* and */usr/share/locale* directories, where you really should not install anything foreign to your distribution. To modify the target directories, pass the *BINDIR* and *LOCALEDIR* variables to *make install*. See Figure 1.

This puts the program components in */usr/local/bin* and */usr/local/share/locale*. There is another possible approach which involves editing the variables in the Makefile *txt2regex-0.7/Makefile*. The *MANDIR* variable is not used in the version 0.7 Makefile. You can copy *txt2regex-0.7/txt2regex.man* to */usr/local/man/man1/*, for example, to be able to call the manpages in future (*man txt2regex*):

```
root# cp txt2regex.man /usr/➥
local/man/man1/txt2regex.1
```

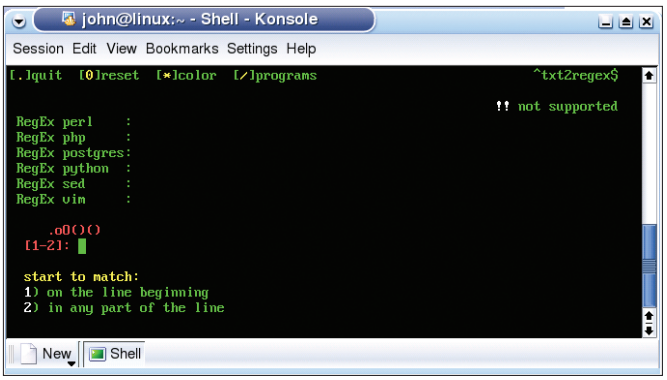

Figure 3: Txt2regex – initial view.

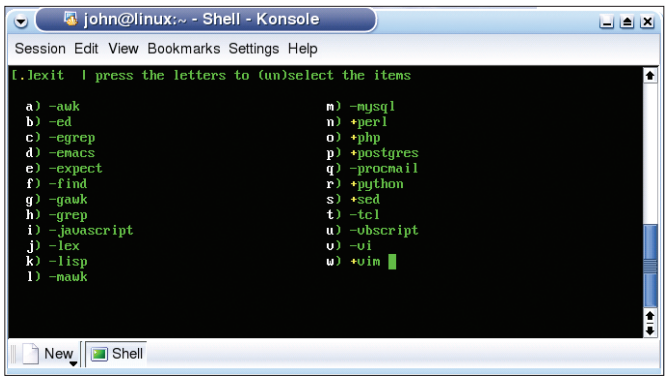

Figure 4: Selecting programming languages.

The program is now ready to run. Type *txt2regex* in a terminal window to launch it.

Options

Let's take a look at the tool's command line options first. There is a basic setting for the background of the virtual text-based console or Xterm. Output from the wizard is more easily legible if you opt for a light colored background; the *--whitebg* parameter takes care of this, as in the following example:

```
~ > txt2regex --whitebg
```

You can remove the color from txt2regex by specifying *--nocolor*. You also need the *--prog* option to specify the programs that the script will display regex's for. The following example tells the tool to display the regular expressions for the vim and emacs editors, as well as the Visual Basic Script and Perl programming languages:

```
~ > txt2regex --prog vim,emacs,➥
vbscript,perl
```

To display all possible regex variants, set the *--all* option instead. You might note that the program quits with a message about your screen not having enough lines, even though you are running it in a large Xterm. You may want to *export COLUMNS LINES* in Bash, before launching *txt2regex --all* to prevent this from happening.

GLOSSARY

Regex: *The abbreviation for regular expression; a combination of special characters that programs can use to search for specific text strings.*

--history is another important option, as it loads a previously defined expression:

```
~ > txt2regex --history '13fl➥
http://www.linux-magazine.com'
```

This displays the *13flhttp://www.www.linux-magazine.com* string as a regex in various programming languages. When you complete an expression, the program displays a shortcut for the history option. You can type *--showinfo* for a list of the program's meta-characters. See Figure 2.

The first line contains the program name, *egrep* in our case, followed by details on the meta-characters, the special characters than need to be escaped, or the Posix classes. For more information on *egrep*, you can access the manpages by typing *man egrep*.

Question and Answer

After launching the wizard (see Figure 3) by typing *txt2regex* in a console, you will note the options at the top: *quit*, *reset*, *color*, and *programs*. Hitting the dot key quits the editor, which then outputs a regex expression that you can access via the *--history* option. You are also shown the text pattern that the regex specifies, for example: *Start match at beginning of line, followed by a string…*

When you press the *0* key to call the *reset* option, the program resets all previous entries. In contrast to this, the asterisk key, *, toggles the *color* on or off. Press the / key to access *Programs*, the tool's most important menu. This will then take you to an overview screen of the programs whose regular expressions you want to use (see Figure 4), and which txt2regex can help with.

A question and answer session comes next, helping you to define your regular expression. The first set of data, *start to match:* applies to where the search pattern starts. This can be *on the line beginning* or *in any part of the line*. Answer by pressing *1* (for the beginning of the line) or *2* (for anywhere). Your input is converted to regex-speak and appears in the regex boxes of the specified applications.

After finishing this selection, the next group is entitled *followed by:*. It has a list of nine numbered items, ranging from *any character*, through *a forbidden characters list*, to *anything*.

After setting the options for permitted or forbidden characters, you are asked *which* of these you want. The editor then goes on to ask you *how many times* the character should occur, offering you seven options ranging from *once* to *up to N* times.

Having completed this step, you are again taken to the overview with the *followed by:* heading. If you then select the *String* option, you are prompted to type the desired text string. Item 7, *POSIX combination*, gives you access to a list in alphabetical order which includes *letters*, *numbers*, *hexadecimal numbers* and *graphic chars*. The dot key takes you back out of the Posix menu.

The wizard uses this approach to compile a text pattern for you, without you needing to worry about the specifics of special characters or the like. When you have finished your regex, just press the dot key.

INFO

[1] Txt2regex: *http://txt2regex.sourceforge.net/*