NAME

efabulor - read text files through **espeak** in a controlled manner

SYNOPSIS

```
[python3] efabulor.py [options] [file]
```

DESCRIPTION

efabulor is a wrapper around **espeak**, a freely-available speech-synthesis engine. It reads a plain-text file, splits it in sentences, and sends them one after the other to be read aloud by espeak. The user can control the reading process (go back and forth, stop, pause and resume the reading, find text, etc.) using the keyboard. Transformation rules can be applied to the text to add newline characters at selected points to tweak the sentence-splitting process. (You can also apply external preprocessing filters to the text to do the same, or to apply any kind of processing to the text). Substitution rules can be applied to each sentence after reading the input file, to further control the reading by **espeak** (e.g., to improve on the results of **espeak**'s text-to-speech-conversion algorithm).

The output of a command can be used as a text source to read non-plain-text files, and a companion program **efabconv.py** is supplied to simplify the conversion of non-plain-text files by assigning preconfigured text-extraction-command templates to different file extensions and/or mimetypes.

Currently **efabulor** is a command-line tool and there are no plans to provide a graphical-user-interface any time soon (however, the program includes the option to be executed in a 'scripted' mode intended to simplify the creation of an external graphical front-end without having to modify the program).

TARGET PLATFORMS

efabulor was developed and tested under Linux (specifically, Debian). It might run under macOS, BSD, etc., but it's not guaranteed to do so. It has been adapted to run (but not thoroughly tested) under Windows.

MINIMAL DEPENDENCIES

Python >= 3.7 (installed in most modern Linux distributions; Windows users can download it from python.org)

espeak (available from standard repositories in Linux; Windows users can download it from espeak.sourceforge.net at the time of this writing)

RECOMMENDED

mbrola and the mbrola voices for your intended languages, to improve the reading performance (available in standard repositories for Linux; Windows users can download it from github.com/numediart/MBROLA at the time of this writing, or kindly do an Internet search for mbrola)

libreoffice and **unoconv/unoserver** to simplify extraction of text from non-plain-text files (available from standard repositories in Linux; for Windows, see libreoffice.org and github.com/unoconv/unoconv).

(Windows: after installing Python, it is recommended to add the psutil package:

```
pip install psutil
```

Alternatively, you can download **pssuspend** from Microsoft's website and copy it to a directory within your PATH.)

INSTALLATION

Copy all six **ef*.py** files to a directory included in your PATH. In Linux, make them executable to be able to call the scripts by their name. In Windows there is also a way to make scripts executable, but otherwise you can call **efabulor** by prefixing 'python3':

```
python3 efabulor.py …
```

Copy the ***.cfg** files to the SAME directory where you put the ***.py** files.

TUTORIAL

I provide a very simple tutorial to give you an idea of what you can do with the program. The full extent of its capabilities is discussed in the manual after this section.

PART 1:

Copy the **tutorial** directory to somewhere in your disk. Open a terminal (in Windows, you must go to **Start->Execute**, write **cmd.exe in** the search box and press **Enter**; if you are not terminal-savvy, kindly look for advice in the Internet or in your Windows documentation).

Change to the tutorial's directory. Copy **tutorial.txt** to **tmp.txt** (we will modify the file later, so it's a good idea to keep the original **tutorial.txt** file).

Now, to start the tutorial, execute (the part in brackets is optional in Linux):

```
[python3] efabulor.py --lang en tmp.txt
```

If everything works as expected, **efabulor** will read the file aloud and stop after reading the last line.

Now press **b** to jump back one line.

Press **a** to start reading again.

Press **v** to jump back to the start of the file; if **efabulor** was reading when you pressed **v**, it will keep reading from there (or you can press **a** again to start the reading again).

Press **n** to jump to the next line (unless you are already at the last line).

Press **m** to jump to the last line.

Try going back and forth and restarting the reading a few times.

Now, while **efabulor** is reading, press the **spacebar**. The reading will be paused (in Windows, it only works if you installed **psutil** or **pssuspend**; see the INSTALLATION section). Press the **spacebar** again; the reading will be resumed where it left.

If instead of the **spacebar** you press **x**, the reading will be stopped and the next time you press the spacebar, it will be resumed from the beginning of the line. (Pressing **a** always starts or restarts the reading from the beginning of the line.)

Press **o**. After a short time, a new window should pop up and you should see the input file in your default editor for plain-text files.

Go back to the console, and press **m** to take **efabulor** to the last line.

Now, go to the editor window and make some changes to the file. After no more than two seconds, **efabulor** should detect that changes were made, jump to the first modified line, and start reading again from there. **efabulor** is designed to give you immediate auditory feedback of changes to the input file as they are being made. (I call this 'tracking'.)

At this point of the tutorial, changes are only being 'tracked' on lines before the current one (that's why I asked you to go to the last line before making any changes).

Press **v** to go to the first line. Press **?** to open a 'tracking-mode' menu. From the menu, select **3** to activate the 'forward' tracking mode.

Make some more changes to the input file. Now **efabulor** will also detect changes made after the current line.

When you are satisfied with experimenting, press **q** to quit the program.

By the way, if you don't like the keys I've chosen for the commands, they are configurable; see [TO BE DONE].

PART 2:

You might have noticed that **efabulor** was splitting lines at the end of paragraphs (or, more technically, wherever there is a ‘newline’ character). It would be nice to have it split the text in smaller sections, wouldn’t it?

Copy **tutorial.txt** to **tmp.txt** again, and execute (it’s one line):

```
[python3] efabulor.py --lang en --transform transformrules.txt
tmp.txt
```

efabulor will read the file **tmp.txt** again, but now it will be split in shorter sentences.

When you are satisfied that **efabulor** is splitting the text in shorter sentences, press **:** (colon).

A new editor window should pop up, with the content of the **transformrules.txt** file. Have a look at it; you will find some introductory information on how **efabulor** can apply transformation rules to the text before splitting it in lines.

If you will be using regular expressions to write your transformation rules (you will!), please note that regular expressions are applied to the text as a whole; that is, they span over paragraphs boundaries (as if you used the **-z** option in **sed**); this gives you more flexibility when defining transformations. More information is given in the [TO BE DONE] section. (And by the way, if you’d prefer to transform the text using other standard tools, such as **sed**, or using any program of your own writing, there is an option to do that; see [TO BE DONE]).

The transformation file provided here is long! Be sure to read it in full, as there is important information at the end. (By the way, in the previous part of this tutorial, you’ve learned that **efabulor** keeps track of changes you make to the input file. **efabulor** will also track changes to the input text indirectly caused by changes to the transformation file; that way, you can edit your transformation rules and have immediate feedback of the effects.

Also note that transformation rules are not the only way to configure the splitting of sentences in **efabulor**. See [TO BE DONE].

Ok, now quit the program (you already know how to do it) so we are ready to start the final part of this tutorial.

PART 3:

If you have modified **tmp.txt** since the last time, copy **tutorial.txt** to **tmp.txt** once again. Now execute (it’s one line):

```
[python3] efabulor.py --lang en --transform transformrules.txt
--subst substrules.txt tmp.txt
```

Pay close attention to the reading. Do you notice any change? If you didn’t, go back to the first line and start the reading again.

Once you detect what was different this time (or if you don’t) press **s**. Once again, an editor window should pop up, this time showing the content of the **substrules.txt** file. Substitution files are similar to transformation files, in that they instruct **efabulor** to edit the input text before sending it to **espeak**. The difference is that transformation rules are applied to the text as a whole before splitting it in sentences; substitution rules are applied on a sentence-by-sentence basis, after the text was split. And if you change the substitution-rules file while **efabulor** is running, a ‘tracking-event’ will be triggered if and only if the new rules introduce changes to the current line.

By this time, you should be starting to have an idea of how you can use **efabulor** to support your workflows. The rest of the manual will provide you a full account of its capabilities.

[TO BE DONE]