PROGRAMMING WITH PYTHON ubuntu 14, Windows10, ubuntu 16 Build the Environment Part 2

emerging-mind lab (EML)
eJournal ISSN 2567-6466
31.Oct - 2.Nov 2017
info@emerging-mind.org

Gerd Doeben-Henisch
gerd@doeben-henisch.de
FRA-UAS - Frankfurt University of Applied Sciences
INM - Institute for New Media (Frankfurt, Germany)

November 2, 2017

Contents

Abstract

The original plan was, to set up under ubuntu 14.04 a programming environment which can use python3, spyder3 and as additional software a speech-synthesis software like pyttsx. While everything worked fine (see the first part of this article), it came to a 'show down' when trying to combine python3 under ubuntu 14.04 with spyder3 and pyttsx3. All trials to overcome the problem led to new problems (see below). Finally I decided to give up the axiom of having ubuntu 14.04 on account of ros (robot operating system), because the primary tool is in this phase the programming language python. Python offers a 'universe of concepts' on its own. And, who knows, there will be a day where ros will work with ubuntu 16.04 too :-). Then I installed ubuntu 16.04 (as upgrade from the internet) and checked the combination of python3, spyder3 and pyttsx3. It worked.

The pages 1-15 of the article describe a first update still using ubuntu 14.04. Then it cam to the crash when trying to combine python3 + spyder3 + pyttsx3. This led to the upgrade to ubuntu 16.04 and the new story. Before it came to the upgrade there was an important interaction with windows 10 helping to understand some limits and possibilities.

1 Why an Update (of ubuntu 14.04)?

There are two reasons:

(i) Depending from the order how you install program packages it can be, that you will need certain additional packages (or not) depending from your installation history. In this update the installation history it is described from scratch again, to keep a certain sequence of installations and dependencies. Mentioned are here only those points which have to be modified compared to the preceding paper in which the installation history was a little bit different.

(ii) The preceding paper started the description after a re-installation of the whole system because the de-installation of the 'built-in' python 2.7.6 version caused a complete system-destabilization and the system was no longer able to work. Later, after the parallel installation of python 3.4.3 it happened, that the directory 'site-packages', which is a standard in a 'normal' python implementation, was under ubuntu 14.04 not where it should be, and different attempts to locate it, failed (but all python3 depending programs worked!). This caused the idea to re-install python 3.4.3. And because there was still python 2.7.6 on the system, it was assumed that one can de-install python3 without problems. But what happened was the same as in the case of the de-installation of python2: the whole system destabilized and stopped working after login.

If you didn't really practice a regular update-policy before, after these experiences you will do (probably).

2 Download And Install Ubuntu 14.04

No change.We again used an installation with an usb-stick.

3 Prepare The SW Environment

The list of programs to be installed was this time the following one (in the real order):

1. A *terminal* program to allow console commands from the administrator.
2. From the console the packet manager *synaptic*.
3. For to write documents with the language LaTeX the *TEXStudio* program has been installed.
4. For a convenient handling of files between different locations the file-manager *krusader* has been installed.
5. As main 'tool for everything', especially for the drawing of figures, diagrams etc. the *open office* suite from the apache foundation has been installed (ubuntu 14.04 has pre-installed *libre office*).

6. Although open-office contains a writer-program there are many situations working with the console where it is helpful to have a small and quick editor, which is a bit more user-friendly than 'vi', but not so powerful we open-office. This is the editor *gedit*. He can be installed with the synaptic packet-manager.

7. To be able to do screen-shots the program *shutter* has been selected.

8. The images from drawing and from screen shots need mostly some post-processing. This is dome with the *gimp*-program.

9. Python 2.7.6 is pre-installed in ubuntu 14.04. But we install additionally the newer version *python 3.4.3* in parallel. (python 3 has its own package management system called 'pip').

10. Together with python3 we install the integrated development environment (IDE) *spyder3*.

11. To improve the documentation of using programs the screen-recorder software *simplescreenrecorder* has been installed. (Before this re-installation two videos have already successfully been produced with this software).

12. For to check the results of your video-production you can use the *vlc*-media-streamer (videos and sound).

13. To enhance the actor-environment simulations it is nice to use a speech-synthesizer to enable the actors to 'speak' to the audience. This is done by the program *pyttsx.py*, which uses the synthesis engine *espeaker*.

For the selection of the programs it was important, that the main programs work also under windows 10 and work as standalone versions, without depending from some cloud. The capability to work as a *local assistance* is important. This is one of the main requirements of the *Local Learning Environments* concept (see other papers)[1].

4 Terminal, Packet-Manager 'Synaptic'

To be able to install the packet-manager *synaptic* which supports You in most cases very gratefully you will need a *terminal* to be able to enter install-commands as super-user. You can select the terminal program from

[1]To be local is not a contradiction to be 'in the web' at the same time. But the 'focus' of the 'local assistance' is very different as if the focus would be located in a cloud driven by completely different motives and interests.

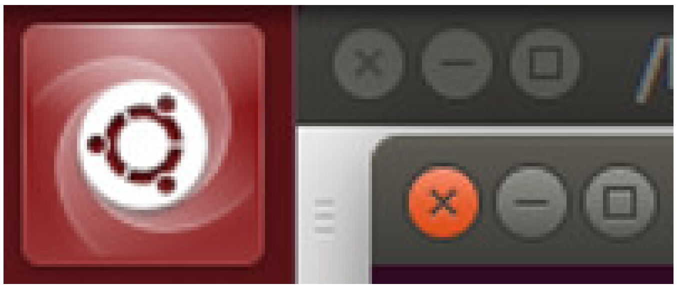

Figure 1: Icon for installed programs

the list of already installed programs in the left-upper corner of your screen (cf. figure 1).

If you select the program *terminal* it will appear in the left bar of your screen. When you click on it with the right-mouse-button you can select to keep it there. This is recommended for all programs which you will use very often.

Within the terminal you can enter the command *sudo apt-get install synaptic*. This will install the program *synaptic* which you will normally use in the future to manage installations or de-installations.

5 Manage Programs

No change.

6 Type Setting with TEXStudio

The *TEXStudio* program is 'on top' of the LaTex-distribution. This distribution is very large. Depending from the kind of document you want to generate you will need different parts from the LaTeX-distribution. To avoid boring messages that some part is missing for the compilation of your document it is recommended to install most of the LaTeX-distribution from the beginning. This can be arranged with *synaptic*.

With synaptic you install *textstudio-qt4* and all the main *textlive*-packages. One of them which you should not miss is the *texlive-latex-extra*-package.

As already mentioned there are numerous options how to organize your LaTeX-document. And because this poses a challenge to nearly everybody there have been many people who invested some time and effort to elaborate *templates* for different types of layouts. One group of contributors of templates are the big publishing institutions and companies. I used very often LaTeX-templates from the IEEE-Institute and from Springer, the LNCS-series of books. All these templates are free to download. This actual document does only use the normal 'article'-mode from LaTeX.

7 Filemanager 'Krusader'

No change.

8 Open Office Suite (e.g. for Drawing)

To install open-office under ubuntu is a bit tricky because ubuntu 14.04 comes by default with 'Libre Office' and the synaptic-packet manager does not offer an option for open-office. Libre-office is in some sense the same as 'Open Office', but, I did use open office long before libre office did exist[2], and therefore I am used to it, and – somehow – I like it more.

When You want to download and install open office in parallel and against libre office, follow the clear instructions given at
https://wiki.openoffice.org/wiki/Documentation/FAQ/Installation/How_do_I_install _OpenOffice.org_on_Linux%3F#Linux_DEB-based_Installation
and
https://www.openoffice.org/download/common/instructions.html.

There is one tricky point in the beginning which has to be handled; it is the 'aggressive' behavior of libre-office to occupy the link to open-office by its own and send every call to open-office to libre office. The first step is to look, where this pointer to open-office is by using the command 'whereis' (attention: the short form of open-office is 'soffice',inherited from the 'old-times' when it was called 'star-office'):

```
gerd@gerd-linux1:~$ whereis soffice
soffice: /usr/bin/soffice /usr/bin/X11/soffice
gerd@gerd-linux1:~$
```

Going into these directories with *cd /usr/bin/* one can detect the following:

[2]Libre office is an off-spring of open-office

```
lrwxrwxrwx 1 root root 34 Apr 29 2017 soffice -> ../lib/libreoffice/program/soffice
```

There is this ’occupying’ link of libre-office redirecting ’soffice’ to libre-office.

The same you will find if You enter the other directory with *cd /usr/bin/X11/*.

One can counter-act by ’un-linking’ this symbolic link with the command *sudo unlink soffice*. This deletes the link in all directories, where it has an occurrence. You can check this whether the entry ’soffice’ has vanished.

Because ubuntu uses a source-code with the debian package-format ’deb’ it is good to find such a source. One is here: `http://www.openoffice.org/download/`. Here one selects an appropriate option. I have selected the 64-Bit linux version with the deb-format. Additionally I have downloaded the md5-checksum to check whether the downloaded source-code is exactly the same as it is intended to provide. After downloading you can do the check as follows:
First check the open-office source-code:

```
gerd@gerd-linux1: /Dokumente/EMP/SW$ md5sum Apache_OpenOffice_4.1.4_Linux_x86-64_install-deb_de.tar.gz
```

This produces the following number:

```
c4c01becf75e2ef18bcef9ed49d5d910
```

Then take an editor; at this moment ’vi’ is the only editor installed, because ’vi’ is at the ’heart’ of every linux system.

```
gerd@gerd-linux1: /Dokumente/EMP/SW$ vi Apache_OpenOffice_4.1.4_Linux_x86-64_install-deb_de.tar.gz.md5
```

... then the editor opens, you make an ’enter’ command and you will see the same number (if everything is correct). Then you can leave the editor with ’shift+:’ and then enter ’q’ for quit.

Now one can ’un-pack’ the tar-ball with the following command:

```
gerd@gerd-linux1: /Dokumente/EMP/SW$ tar xvfz Apache_OpenOffice_4.1.4_Linux_x86-64_install-deb_de.tar.gz
```

This generates a folder for the language you have selected. In my case I had selected ’german’ resulting in a folder ’de’. Enter this folder. There you will find another folder named 'DEBS', where the deb-formatted packages are collected. To install now the packages you have to enter the command *... de/DEBS$ sudo dpkg -i *.deb*.

Finally one has to integrate open-office into the desktop. This can be reached by entering the directory *desktop-integration* within the *DEBS*-directory and using the command *sudo dpkg -i *.deb* again.

Before one now can start one has (i) to find the actual *location* of soffice, the (ii) one has to provide a *java-runtime-environment (JRE)* which is presupposed by soffice, and finally (iii) one has to set a new symbolic link in the directories /usr/bin, /usr/local/bin and /usr/bin/X11 for the new location.

The answer of task (i) is given as: */opt/openoffice4/program/soffice*. It has to be commented that there have been no official hints, where to search. The command 'whereis soffice' does not work.

The task (ii) is solved with the packet-manager synaptic enter 'jre' and selecting 'default-jre'. This installs the actual jre-software.

The task (iii) is solved by entering the mentioned directories and set a symbolic link as follows: *sudo ln -s /opt/openoffice4/program/soffice soffice*

After these preparations open-office starts in every director with the command 'soffice'. It appears an icon in the left bar of the screen and if one clikcs (recommended) with the right mouse-button on the icon one can select 'keep in the start-menu'. Then one can use open-office in the future only by clicking on the icon (clearly I have de-selected in the same manner libre-office :-)).

9 Simple Editor 'gedit'

Simply install with 'synaptic'.

10 Screenshots with 'shutter'

Simply install with 'synaptic'.

11 Image Post-Processing with 'gimp'

Simply install with 'synaptic'.

12 Python2+3, spyder3

By default is python 2.7.6 and python 3.4.3 installed. You can test it if You enter the command python or python3 in the console. Here the output in the case of python3:

```
gerd@gerd-linux1:~$ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If You are looking with *whereis python3* which dirctories are associated with the name you will get the following list:

```
gerd@gerd-linux1:~$ whereis python3
python3:
/usr/bin/python3.4
/usr/bin/python3.4m
/usr/bin/python3
/etc/python3.4
/etc/python3
/usr/lib/python3.4 (<-- DistPackages)
/usr/lib/python3 (<-- DistPackages)
/usr/bin/X11/python3.4
/usr/bin/X11/python3.4m
/usr/bin/X11/python3
/usr/local/lib/python2.7 (<-- DistPackages + Site-Packages (leer))
/usr/local/lib/python3.4 (<-- DistPackages)
/usr/share/python3
/usr/share/man/man1/python3.1.gz
```

The only place where a folder with the name 'site-packages' occurs is connected to */usr/local/lib/python2.7*. What is the reason for this absence of this standard folder? Is it the absence of external programs because python3 is only installed with very basic stuff until now?

The *integrated development environments (IDE) spyder and spyder3* are not installed in the beginning.

To extend the python3 collection of programs as well as the connection with spyder3 we want to use the python3-packet management tool *pip*. But this must first be installed (although it should be included in 3.4.3).

After installing *pip* with synaptic we find pip not in the directory */usr/lib/python3.4*, but in the directory */usr/lib/python3*:

```
/usr/lib/python3.4 (<-- No Dist-packages, no pip)
/usr/lib/python3 (<-- DistPackages + inside: pip)
```

Thus it seems that python3 is the general term for all 3.x-versions. Besides this if we install spyder3 with synaptic then the spyder3 libraries and additional packages like 'numpy', 'scypi' etc. will be stored under 'python3'. Comparing this with the directory-structure if python3 in a windows10 environment it seems that under ubuntu the concept of the site-packages has been replaced with the directory *dist-packages*. A simple test confirms this hypothesis. If we load the graphics module *graphocs.py* from John Zelle by starting the filemanager *krusader* from the command line with *sudo krusader*) to copy the module into the */usr/lib/python3/dist-packages* directory, then the python-console within spyder3 recognizes 'graphics' when we give the command 'import'.

It followed many hours of trying to implement a spyder version. With python 2.7.6 and spyder 2 it worked but it was really poor. No integrated python console in spyder2. Working with pip was not possible. The usage of pyttsx3 wasn't possible either.
Thus, wwe stopped working with python3 under ubuntu 14.04.

13 Recording with the 'simplescreenrecorder'

When we start programming the it is important to be able to communicate the results to an audience. Explaining complex programs is always difficult. Thus showing the behavior of a program in some graphical representation can be of help. For this we install the *simplescreenrecorder* from Maarten Baert, a wonderful tool. Because the synaptic package-manager does not now the program *simplescreenrecorder* we use another method to install programs: we extend the set of possible web-addresses from where we can download with apt-get:

Adding a new repository to the list: *gerd@gerd-linux1: $ sudo add-apt-repository ppa:maarten-baert/simplescreenrecorder*

Check for possible updates until now: *gerd@gerd-linux1: $ sudo apt-get update*

Install the program: *gerd@gerd-linux1: $ sudo apt-get install simple-screenrecorder*

Then check whether it works with a direct command in the terminal: *simplescreenrecorder*

It starts correctly and we fix the icon at the left bar of the screen to be able to click for a start in the future.

Now, two steps are left :-).

14 Video-Streamer 'vlc'

Simply install with 'synaptic'.
You can take any video or autio file to check whether it works.

15 Speech Synthesis with 'espeak' and 'pyttsx'

Today we are quite used to the possibility that programs can recognize spoken language (*'speech recognition', (SR)*) or that programs can generate speech (*'text-2-speech synthesis' (TTS)*). Mostly these applications are backed up by programs located in a cloud which collects millions of speech events and computes statistical models which words are associated with which words and how these are pronounced. This supports better and better quality. The disadvantage of this strategy is that you depend completely on an active connection to the Internet and that your own program has no control about the content and shape of these data. With regard to the philosophy for the *Local Learning Environment* paradigm we favor local applications which can connect by themselves to the Internet, if necessary.

For the beginning we will use an open-source local speech-synthesis solution based on the programs *espeak* and *pyttsx*.

The program *espeak* can esily be installed with *synaptic*.

Information to the program *pyttsx* can be found on this site:
https://pyttsx.readthedocs.io/en/latest/#

Installation is done with the aid of the python packet-manager *pip* as follows: *sudo python3 -m pip install pyttsx*. This installs the program successfully in the directory */usr/local/lib/python3.4/dist-packages*.

There were lots of syntax-errors posted during the installation (nearly in all files). Trying to run the test examples from the website didn't work (although the same examples under windows 10 are working). Comparing the files from the ubuntu installation and the win10 installation shows that the files in the window-installation don't exhibit the syntax errors from the linux installation. Thus,the application of pyttsx3 under ubuntu 14.04 has been canceled.

16 Ubuntu 14 and Windows 10 as Environment for Python3

By several reasons I am using python3 + spyder3 + pyttsx3 also in parallel in the windows10 environment.

Under Windows 10 you have the newest versions of python3 (3.6.2) and spyder3 available, and they are running. The speech-synthesis Software pyttsx3 is running. The two available (German and US-English) voices are feminine. They have a good quality. A runnable demo-program will be attached to this posting as a download.

Still battling with ubuntu 14.04 I detected that the structure of the python-directory is different than under windows. While under windows 10 you have the standard directory 'site-packages' for all the external modules you will not find such a directory under ubuntu 14.04. There all modules are located in a directory called 'dist-packages'. Furthermore you have two directories for python3: the actual release with python3.5 and the 'collector directory' python3. Both are used together.

Therefore, if one is only interested in using the newest version of python3 with spyder3 then windows offers you more (I did not yet clarify the whole coverage of modules supported by windows 10 compared to ubuntu 14.04, ubuntu 16.04 until now). If you need the ubuntu environment because it offers you a very flexible, open working environment, then you go to linux (or use 'both' on demand :-)).

17 Opting for ubuntu 16.04 besides ubuntu 14.04

When I decided to install ubuntu 16.04 *besides* the existing ubuntu 14.04 on a second disk in my computer I had some work to 'free' this second disk because I had installed the LVM-System (logical volume manager) with ubuntu 14.04. This is a very versatile management tool, if you have more than one storage device. Now I had to de-install lvm for the second disk. This was no problem. Then I could install from my usb-stick first ubuntu 14.04 with the idea afterwards directly to upgrade to ubuntu 16.04.

Everything worked fine until I mad a first restart to boot the new system. The boot manager offers both disks as separate options. But surprise: selecting the second disk didn't work. After some trials I tried the first disk and there it was – the new system?! This was really strange.

Until now I had no time to investigate more what went wrong. I upgraded from 14.04 to 16.04 directly from internet. This worked fine, because I wanted to check whether under ubuntu 16.04 my target programs python3 + spyder3 + pyttsx3 would work

18 Rebuild the Environment with ubuntu 16.04

Using the experience from the first 15 pages of this article I installed directly:

1. A *terminal*.
2. By command line in the terminal *synaptic*.
3. File-manager *krusader*.
4. With synaptic *python3* and *spyder3*.
5. With python3 and the aid of the *pip*, the spyder packet management system, I could install the speech-synthesis program *pyttsx3*.
6. I could use the python-pyttsx-test-program from the windows 10 environment under ubuntu 16.04 directly (see downloadable file in the post). The only difference is the speech-engine used under ubuntu 16.04 – ('espeak') compared to ('sapi5') from windows 10 – and the voices (under ubuntu 16.04 you have masculin voices with bad quality, but you have many more languages!)
7. *TEXStudio* with nearly all packages from *texlive*.

Everything worked fine until now... :-)
Upcoming there will be more reports working with ubuntu 16.04 and the python-team of software modules. Also some more videos...