Hey, Researcher! Open Source Tools For You!

By Amit Kumar Saha

Disclaimer:
Your author is writing this article after having just finished his latest task - conducting numerous experiments which involved processing a lot of data, and finally writing the paper on it in just under five days. He wonders how he would do it without some of the tools he will be talking about in this article, leaving aside the pleasure of working on Linux systems. His views are biased and are meant to be taken *cum grano salis*, but I hope you will give him a fair chance.

What am I addressing in this article?

My meeting with Linux happened accidentally 10 years ago, with the fun of wiping out and putting in a Linux distribution numerous times on my first computer without having any clue about how to work on it. Finally, after a lot of such iterations I finally settled in, working on it full time for the last 5 years. The accident that happened 10 years ago stood me in good stead both when I was working at a software company and now when I'm working in a research lab. In a research lab, we need to plot figures, run batch jobs remotely, perform scientific and numerical computing, and last but not least, publish results by writing papers. Just a few days before I wrote to the Editor-in-Chief about this article idea, I helped a Windows friend draw some plots (fixing them for publication) for her paper, wondering all the while "Jeez, how do Windows users do it?" Most tools that I used were primarily for Linux. On Linux, things are so much easier. You can focus on the real job instead of bothering about setting up your tools. This article is therefore an unabashed claim about the coolness of working with these tools while you focus on your research. Some of these tools can be set up on a non-Linux system too; however, I shall address their usage on a Linux system only and make no attempt to cover any non-Linux issues. The primary focus here is to introduce my readers to these tools, and not to convert them to Linux. *It would, in fact, be a good exercise to get all these tools that I talk about here on Windows!*

In this article, I shall take a look at (in no particular order) getting up and on with LaTex, *gnuplot*, *GNU octave*, Python scientific libraries, *Beamer*, and some other tools and libraries.

Hello Linux

Before I start talking about using the scientific tools, I shall write a few lines about how you can set up a Linux system for yourself - if you don't already have one, that is. While bare-metal Linux installations are the best case scenario, it may not always be possible because your primary work may be on Windows. In such cases, the best solution is to go for a *virtual Linux installation*. In my humble opinion, I have found using VirtualBox to be a breeze when you need to set up a virtual machine. Just download a Linux distribution like Ubuntu and install it using VirtualBox. Once you have the basic installation, take a little time to gain some hands on familiarity with the terminal, using the package manager to install packages, and using a text editor.

Here are some resources which can help you getting started:

- Ubuntu community documentation on VirtualBox
- Documentation for the latest stable version of Ubuntu

Giving LaTex a try

LaTex is a document preparation system for high-quality typesetting and is widely used for writing scientific papers, technical manuals, and even creating presentations *(more on this later)*. *TeX Live* is an easy way to get up and running with LaTex on your Linux system. A file written in LaTex is usually

saved as a *.tex* file. Once you have created your *.tex* file, you will compile the file using the *latex* command, which gives you a *dvi* file, which you can convert to a PDF file using a tool like *dvipdf*. You can also directly get a PDF output using the tool *pdflatex*.

You can edit LaTex in your text editor of choice. The widely used *VIM* and *Emacs* both support LaTex editing with syntax-highlighting, code-completion, etc.

The best way to get started with LaTex is by getting started. Start your text editor, type in some Latex and see the results. Here are some great resources to help you master it:

- The (Not So) short introduction to LaTex
- Text Processing using LaTex
- Draw the LaTex symbol and get the corresponding command

gnuplot

If you are doing experimental work, there is a fair chance that you are going to present your results in a graphical plot. Please welcome *gnuplot*. It is a command-line driven graphing utility for creating 2-dimensional as well as 3-dimensional graphical plots. It supports output to many file formats with *eps* and *fig* being of special interest to the researcher. The easiest way to install *gnuplot* is using the package manager of you Linux distribution.

Here are a couple of links to help you get started with *gnuplot*:

- gnuplot documentation
- gnuplot not so Frequently Asked Questions

Calling gnuplot *from your C program*

A cool way to use *gnuplot* is to use it to display continuously changing data that your program may be generating. For this example, let's assume that you have a program written in the C programming language and you want to send the data it's generating to *gnuplot*. The simplest way to do this is to use the *popen* system call. Please refer to this *2-cent tip* published here sometime back.

There is also an ANSI C interface to *gnuplot* available here, but I haven't personally used it.

GNU Octave

GNU Octave (referred to as Octave from now on) is a software tool intended for numerical computation. Octave has extensive tools for solving common numerical linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential-algebraic equations. Octave is available for use on Linux, Solaris, Mac OSX, and Windows. If your study or play involves numerical computations, Octave is for you. Octave has been designed with MATLAB compatibility in mind, so with careful design you could write scripts which run on both MATLAB and Octave. It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

Octave is most definitely available in the package repository of your Linux distribution and that is the easiest way to get it.

GNU Octave comes with extensive documentation, which is the best place to get started.

In case you happen to like this author's writings, he is currently writing an article series on Octave elsewhere. Please refer to these blog post for PDFs of the articles.

Add Python to your toolkit

I shall not use this section of the article for Python language advocacy. Rather, I shall point you to some ways you may consider using Python in your research, if you are already using it, or think of using it in the future as just another of your many experiments. Python is already installed on most of the Linux distributions. Let us now see some of the ways you can use Python:

As a scientific calculator

With its *math* module, Python can be a very useful command-line scientific calculator. For example:

```
$ python
Python 2.6.4 (r264:75706, Dec  7 2009, 18:45:15)
[GCC 4.4.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> math.factorial(10)
3628800
>>>
```

As you can guess, you will have to import the math module every time you want to use it. You can avoid doing this by automatically importing it at startup:

1. Create a file: .pythonrc in your $HOME and place this line:
2. *import math*
3. Now in your .bashrc or similar:
4. export PYTHONSTARTUP=$HOME/.pythonrc

Now, every time you start Python interactively, you should have the *math* module already imported.

```
$ python
Python 2.6.4rc1 (r264rc1:75270, Oct 10 2009, 02:40:56)
[GCC 4.4.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
 >>> math.pi
3.1415926535897931
```

The *decimal* module is another interesting module which provides support for decimal fixed point and floating point arithmetic. *fractions* is another interesting module, which can be used for rational number arithmetic.

The beauty is that you do not need to know the Python *programming language* to use these modules. Just fire up the Python interpreter and import the module(s) you want to use, then call the appropriate function.

Scientific libraries

If you are already conversant with Python, then you would definitely like to know about some scientific libraries which you may find useful in your research:

- **NumPy and SciPy**: *NumPy* is an extension to the standard Python programming language, adding support for large multi-dimensional arrays and matrices. *SciPy* builds upon *NumPy* to provide various mathematical functions for Integration, Optimization, Interpolation, Linear Algebra, Statistics, and others.
- **matplotlib**: This is a plotting library for Python, which is also used by a lot of other Python scientific tools for plotting data.
- **NetworkX**:NetworkX is a Python library for the creation, manipulation, and study of complex networks. Even if you do not have the need to build very complex networks, it is fun to play around with it to create simple graphs too.

Beam it with Beamer!

You have done your experiments, sent in your findings, they've been accepted, and now you are going to present them at a conference. You want to prepare some slides to give you company as you talk about your work. OpenOffice.org Impress, KOffice? How about taking a look at something different? You will surely like the results, I promise. *Beamer* is a LaTex class for making presentation slides. In LaTex parlance, it is just another Document Class. You shouldn't even make an attempt to use Beamer, if you have no idea what LaTex is. However, if you already use LaTex, chances are you swear by it, then Beamer is for you.

The easiest way to install is to use the package manager in your Linux distribution. You will see that some more packages providing other extra LaTex classes need to be installed. Prominent among these are 'latex-xcolor' and 'pgf'.

There is, of course, a way to install the LaTex class manually. It's easy to do it, but not easier than the previous method. The LaTex Beamer user guide shows how to do it.

With the danger of being pulled up by the chief editor for self-publicity, the author would like to point to his article *Typesetting Presentations with Beamer*, published elsewhere. The PDF copy is available from here.

Others you may find useful

In this section, I shall talk about some of the other tools, which are useful for scientific computing that I know of:

- **Sage**: *Sage* is an open-source mathematics software *system* which brings together a huge number of open-source mathematical libraries and tools under one common umbrella. In their own words, the Sage community is building a car, and not reinventing the wheel. The Sage tutorial is a good place to start exploring it.
- **Xfig**: Xfig is a drawing tool which may be used to create figures, import and export figures in various other formats. *fig* is the native format that Xfig uses to save its images. A useful way to use *Xfig* that I have learnt is to export your plots from *gnuplot* as *.fig* images, open them in *Xfig* to add the relevant text/labels to your plots, then export the plot in a format you desire.
- **GNU Scientific Library**: The GNU Scientific Library is a C/C++ numerical library providing support for Linear Algebra, Complex Numbers, Numerical Differentiation and others.

The small ones

One category of tools that I haven't talked about yet is some of the Linux utilities, the ones that ship along with the Linux system - *Shell scripting*, *awk* and *sed*. My knowledge of each is very limited, but I have a working knowledge of them all, and I can get things done with them, which is what matters a lot of the time.

- **Shell scripting**: Shell scripting is very useful for running batch jobs: fire up a job and it should finish without requiring any intermediate user intervention. It's all the more useful when you have to run an experiment of yours a large number of times. Writing a shell script is often as simple as just writing several separate commands into one file in a *for* loop and running that file. Of course, that's an oversimplification, but sometimes it *is* just that simple to set up a batch job. Do not get put off by the title, but the Advanced Bash-Scripting guide is a good place to get started with scripting for BASH shells.
- **awk**: The only way I have used *awk* so far (and a lot of times) is for text extraction from huge files containing numerical and textual data. Even with this limited use, it can be a useful tool when you want to extract text that is stored in a tabular format. The GNU Awk User's Guide is the place to get started to explore *awk*.
- **sed**: *sed* is a stream editor and is useful to carry out batch editing as well as editing of individual text files. This three-part series on sed is a comprehensive treatment of the features of *sed*.

Conclusion

It's often the case that we do not know of a great tool which would simplify our work as we go about our research work everyday. My hope as a novice researcher is to acquaint the reader with the kind of software tools available in the Open Source world. I have made no attempt to be exhaustive here. There are a lot of other tools out there, which I haven't yet used, so I have refrained from writing about them. I hope you have enjoyed reading this article as much as I have writing it!

Acknowledgements

Credits are due to the folks on the mailing lists and the documentation of all the projects that I have talked about today, because they are the source of my learning.