Seting rappture up in your computer

David Saenz
August 22, 2011

Keywords: nanoHUB, rappture, linux

1 Introduction

As it's creator describes it, rappture (**R**apid **app**lication infrastruc**ure**) is a tool kit that facilitates the development of powerful scientific applications. With C++ at it's core, rapture can be combined with a variety of programming languages including C, MATLAB, Fortran, Octave, Perl, Python and Tcl. It provides several tools for the user so to speed the development of such applications. One of it's most interesting and useful resources is a powerful XML based Tk Graphical User Interface generator. Rappture made applications are easy to deploy on the nanoHUB and share with a large community of researchers and students throughout the web. Still one of the main problems with rappture application development is the restriction of some services through rappture remote development environment like MATLAB. This is the main motivation to prepare a local installation of rappture. More information about this tool kit can be found on the project web site: rappture.org.

2 Setting up rappture on your local machine

The objective of this tutorial is to create a functional MATLAB enabled rappture development environment. In order to achieve this, The following will be needed:

- Working Linux OS installation in your machine (preferably Ubuntu).
- The latest rappture build from rappture.org.
- MATLAB license and installation disks.

Each of the steps necessary to set up rappture is described below.

2.1 Installing Linux

If rappture development is intended to be performed in a computer lab with a working Linux installation, it is preferable to contact the system administrator to perform the installation. The procedure described here is guaranteed to work only on the latest releases of Ubuntu. While rappture has been successfully installed in other Linuxdistributions, full functionality is not guaranteed (Note that the Linux version running on the nanoHUB is Debian GNU/Linux 5.0 "Lenny").

You can set up Ubuntu in one of the following ways [1]:

Native installation - HD Drive partition and native installation is time consuming but takes full advantage of computer hardware. You will need to prepare a bootable USB or burn a disk for installation. Refer to ubuntu.com for specific instructions on how to perform this kind of installation

Virtual Installation - Virtual Machine Installation is time efficient but has hardware constrains (graphics). You can use programs like Virtual Box, VMWare or Parallels for mac. They are all very good, but in this tutorial, VMWare Fusion was used (see Figure 1). Plenty documentation is available online.

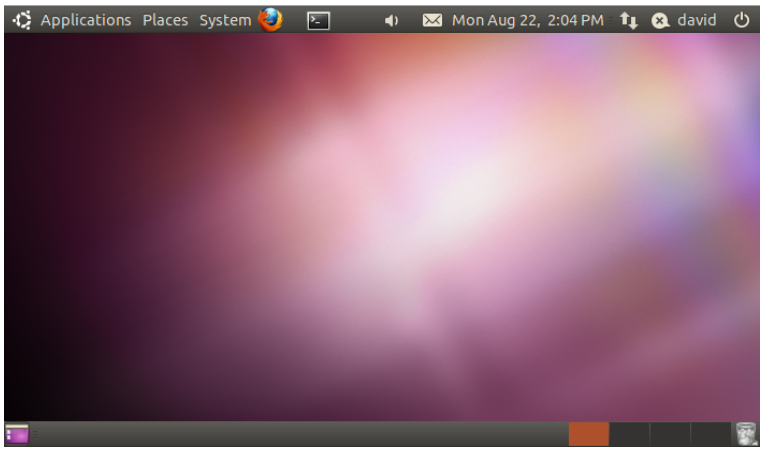

Figure 1: Ubuntu desktop.

[1] Make sure you download the latest Ubuntu installation image (Ubuntu 11.04 at the time of writing this tutorial, but Ubuntu 10 will also work).

2.2 Installing Matlab

After performing all necessary updates and making sure Ubuntu works, you can proceed to install MATLAB. Installing MATLAB in Linux is an easy but tricky task. Make sure you follow the installation manual before running any terminal command. A typical MATLAB installation on Linux starts by sourcing the installation script within the main MATLAB installation folder (you might need administrative privileges to perform the installation):

```
sudo ./install
```

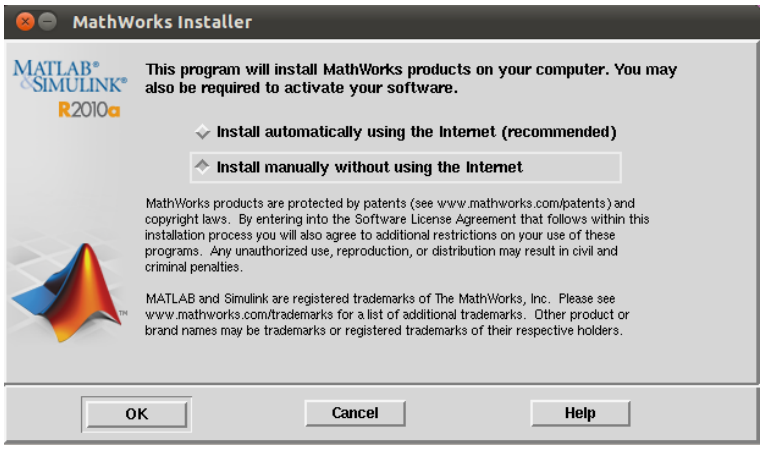

Figure 2: MATLAB installation wizard.

After typing your password, the MATLAB installation wizard should come up and guide you through (see Figure 2). Common problems at installation times include not giving executable permission to critical installation files, or not having writing privileges at the installation directory. To fix this, you can temporally enable all permissions to the MATLAB installation folder by typing the following command in the terminal:

```
sudo chmod -R 777 /matlabInstallDir
```

Where "matlabInstallDir" is the path on which your installation files are. After successfully installing MATLAB, try running it from the terminal by typing its name as a command:

```
matlab
```

After pressing enter, the MATLAB editor should pop out. If after typing the command you receive the message "command not found", that could mean that the MATLAB binaries are not in your executable search path. So you will have to add them yourself. This or create a symbolic link to it in one of your default binary search folders. You can add the MATLAB executable to your path, by typing the following command at the prompt:

```
export PATH=$PATH:/matlabInstallDir/bin
```

This will only enable MATLAB for this terminal on which you typed the command. If you want bash to remember to add MATLAB to your path every time you open a terminal, you have to modify it's configuration file. For bash (default shell for Ubuntu), this file is called ".bashrc" and is located in your home directory. You can find it by typing the command

```
ls -a
```

in your home folder. Then you can modify it using your favorite text editor and add the "export" command previously mentioned at the end of the file. What happens, is that when starting the terminal, it runs every command in this script. This is the way you can pre-configure any setting on your shell. You can find more about configuration files of bash and other kinds of shells in Wikipedia.

Furthermore, if you would rather create a symbolic link to the executable file, navigate to your default binary folder:

```
 cd /usr/local/bin
```

and then type the following command:

```
sudo ln -s /matlabInstallDir/bin/matlab matlab
```

The difference between these two types of set up, is that in the first one, when we added the installation directory to the PATH variable. We are telling the shell which folder to look for matlab explicitly. Whereas in the second one, we are redirecting it from a place on which it found a pointer to the MATLAB executable. The second one is more common, because later on, if a newer installation of MATLAB is present in the machine, we can just redirect the pointer without touching the path variable. This is simpler and more organized than if we were edit our path every time we install or update software on our machine. You can find more information on symbolic links and executables in Wikipedia.

2.3 Setting rappture up

Installing software packages in Linux could be quite complicated. Specially when a source compilation is needed. It is better, if we can get access to a working build of the software, so that we don't have to worry about compiler libraries and many other things. This is why, we are going to download the prebuilt rappture package from the project web site. Go to rappture.org and download the package that corresponds to your machine architecture. Make sure you get the stable version. Then, uncompress it and will get a folder with a lot of numbers as a title as a product. Then, you can do one of the following to fit rappture into your system:

Edit script install directory - Follow the instructions on the rappture installation website.

Correct package location - Create the specified directory and move rappture in there. (this is explained below).

If the second choice is chosen, In the scripts you are instructed to change within rappture(i.e. rappture.env, rappture, simsim, etc..) there is a certain installation path. it looks something like:

```
destdir=/apps/rappture/20091009
```

and it is located near the beginning of the scripts. So go to your root directory, and create these folders. You might need administrative privileges because we are working within a protected zone. Type the following commands in the terminal to create the directory:

```
cd /
sudo mkdir apps
sudo mkdir apps/rappture
```

then, move all the package you just downloaded to this location using the "mv" command (assuming you decompressed the file within your downloads folder on your home directory):

```
mv ~/Downloads/20091009 /apps/rappture/
```

Then, we need to test if rappture is working. We do this in a similar way as we did it with MATLAB. We should be able to type "rappture" on the terminal and invoke rappture. But before we can do this, we need to tell our terminal where to look for the rappture binaries. As before, we can add a symbolic link in our /usr/local/bin folder to point to the rappture executable script. Do so by typing the following commands on the terminal:

```
cd /usr/local/bin
sudo ln -s /apps/rappture/20091009/bin/rappture rappture
```

After doing this, you should be able to run rappture from the terminal. Test the installation by executing the "rappture" command. If it works, then a message saying: "tool.xml not found" should come up on the terminal. This indicates it's working.

To test rappture and matlab together, you can try to run some of the rappture examples within the rappture installation package. They are located within the rappture directory, on a folder called "examples". You can go to this directory and run one of them by typing:

```
cd /apps/rappture/20091009/examples/app-fermi/matlab/uncompiled
rappture
```

Then the rappture interface should pop out showing the fermi sample application (see Figure 3). Click simulate to see if rappture calls matlab the correct way. then, if everything is fine, you should see rappture calling matlab and at the end, a nice fermi-dirack distribution curve on the rappture plot screen. You can also try other examples to see how different rappture objects behave. Navigate a few directories up in the examples folder to reach the top, and then go into the "zoo" folder. Here you will find a variety of rappture objects and examples on how to use them.

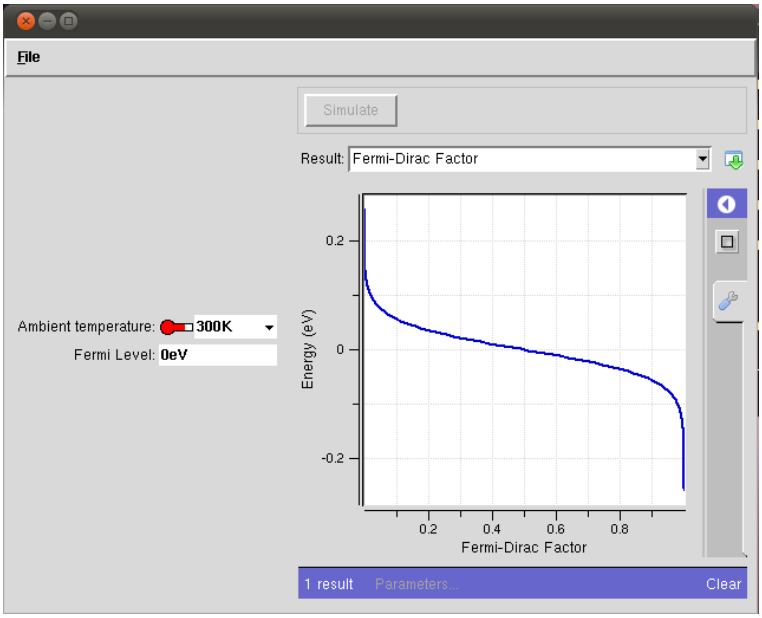

Figure 3: Running rappture.

3 Final remarks.

There are some objects which doesn't work under the previously explained installation. Such is the case of most rendered objects or other visual appealing graphs like clouds. Still, if you are lucky, some of them will run. You will have to test each of them to see which ones work and which ones doesn't. But the main objective, which is to be able to come up with simple graphs out of MATLAB generated data is accomplished. Enjoy!

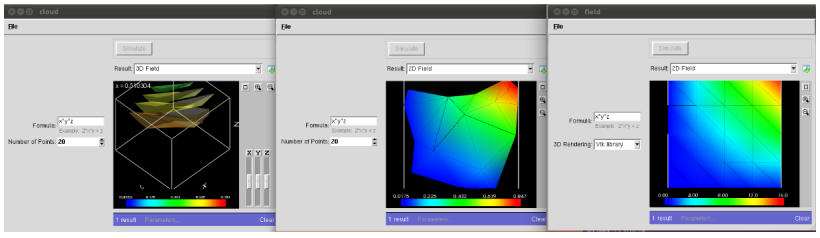

Figure 4: Rappture rendering.