Benscomputer.no-ip.org

Getting Started With Linux Part 2
Installing Software

In this Article I will assume a few things, firstly that you have read **Getting Started With Linux Part 1, (http://benscomputer.no-ip.org/Articles/getting_started.html)** and also that you now have some flavour of Linux installed.

A problem apparantly encountered by many on Linux, is the task of installing new programs. To install a program the easiest way is always to use your Package Manager. This can usually be found in the configuration module that comes with the distribution. In Mandriva it is the Mandriva Control Centre, in SUSE it is YAST and so on.

If you use KDE then you could also consider installing KPackage. This interfaces well with most distributions that I have tried it on, and is a fairly quick and easy way to install software.
As long as you have **root access (http://en.wikipedia.org/wiki/Superuser)** then installing via a package manager can be quick and painless. Removing software is also quite easy through a package manager.

However the software you want is not always available via your package manager, this leaves a choice. You can either compile the program from source, or you can try to find a precompiled package for your system. This article will try to cover both these methods.

If you wish to find a precompiled package, then you first need to identify the type of package to get. If you are using an RPM based system (Mandriva, SUSE, Red Hat, Fedora to name but a few) then you can search for an RPM on **rpmfind.net (http://www.rpmfind.net)** or **rpmbone.net (http://www.rpmbone.net)** . These normally have the filename extension .rpm. You need to ensure that you download an RPM for your distro otherwise you may encounter problems later on. Both sites mentioned tell you which distro the RPM is for. Once you have downloaded the package, you can usually just double click it, and your package manager will install it for you. However you may find (and people often do) that you have problems with things called dependancies.

Dependancies are programs or libraries that the package you are installing requires to function correctly. WHen you install something with your Package manager it normally auto selects your dependancies for you.

If you do hit "Dependancy hell" then for an RPM based distro, I would recommend writing down the name of the package or library that is missing, and then search for it in your package manager, if that is not available then search for it on one of the sites mentioned above.

You may find it simpler to use an **Autopackage (http://autopackage.org)** where available, they are not very well adopted just yet, but where available they do make life easier. The first time you try to install an Autopackage, it will offer to install the software necessary to install Autopackages for you. Once this is installed, it will confirm you want to install the Autopackage, and then proceed. This method also gives you a screen similar to that of the Add/Remove programs dialogue in Windows.
It may be worth emailing the developer of some software you like, and asking *politely* if they would consider putting it in an Autopackage.

For those running a Debian derivative (i.e. Debian, Ubuntu etc) there is the option to download debian files, these normally have the filename extension .deb. As with RPMs you can normally double click them to set them installing. In my experience the Debian method is far more apt (forgive the terrible pun) and checking and fetching dependancies.

Not all projects offer Distro specific packages however, in fact many offer only source tarballs. This is basically an archive with the source code to the program inside as well as some other handy tools. You are often given a choice between the type of archive to download;

filename.zip - A zip file, yes exactly like you used to open with Winzip ;-)
filename.tar.gz - This is a Gzipped Tarball, its compression is better than .zip files
filename.tgz - This is exactly the same as .tar.gz just a different way of naming it
filename.tar.bz2 - This is a Bzipped archive, this has the best compression out of the above, which leads to a smaller download

Which one you choose is up to you, most distros have the tools to open any of these archives installed as default.
As you are going to be compiling from source, you will want a terminal of some sort open. If you are running KDE then Konsole is more than sufficient.
Having downloaded the archive foobar (Which for the sake of argument we will say is in your home directory)
in the console run

cd ~
then

tar xvzf foobar.tar.gz

or

tar xvzf foobar.tgz

or

tar jxvf foobar.tar.bz2

or

unzip foobar.zip

followed by

cd foobar

With those to commands you have extracted the tarball and then changed into the directory it has created. if you run the command

ls

you will see which files are available. Remember this article is intended as a guide only and is no substitute for proper documentation. If there is a file called INSTALL there I would recommend opening it in a text editor and having a quick read to see how to install the program.
As a general rule of principle, you use the following commands to compile the program from

source;

./configure
make

The *./configure* runs a script called configure in the current directory, you will have seen it when you ran *ls*, this script checks your system for various dependancies, and also to obtain some configuration variables. If the script fails it will normally tell you what you are missing, assuming this occurs, the first place to go is your package manager, find if the required package is available (You may have to google the name of the missing library to find out the package name), and install it. If it isn't available then you are going to need to hunt on the net for the package, use the resources mentioned above, or pop over to the homepage of the missing package and grab it from there (may mean another source compiation however). Once the erroneos package is installed, run ./configure again (remember you need to be in the directory that was created when you unpacked the archive)

OK if the *./configure* script worked, then you will be looking to run make. Simply put, *make* tells the system to begin compiling the source. If this stage fails, then look through the output for the first error of what is normally a stream of errors and warnings at the beginning. Your best bet is to then put this into google unless you can ascertain what is wrong yourself. Remember to put this error into any posts you make on forums about the subject.

Once that has completed you can run the program from the local directory, run ls to see which files are there and then (assuming the program is called foobar) run ./foobar to run the program.
If however you want a system wide installation of the program (i.e. so every user can run it) then follow the next steps

su
[roots password]

su stands for switch user, it is normally used to temporarily log in as root, however it can be used to switch to other users i.e. *su ben* would lead to you being prompted for bens password. You may find you are unable to run su, it is not always available to all users, if you wish to use su then you need to log in as root and add your user to the group 'wheel'. Alternatively log in as root then run (if su worked you also want to run this)

make install

This will install the program to a directory in your PATH (More on that in a later article), so from any console you will be able to run (in the example)

foobar

without needing to be in the directory in which it is actually installed.
Once this has completed don't forget to run exit to exit the su session.

I have seen many tutorials that tell you how to install from source, but very few that also tell you how to uninstall it if needs be. To uninstall the program (assuming you have run make install, otherwise you can effectively just delete the directory, but read the warning first) as root (or using su) run

make uninstall

Once that has completed the program is uninstalled, you can either then delete the directory

cd ..
rm -r foobar/

or run

make clean

if the reason for uninstallation was to create a clean installation of the program.

I hope this has advanced your understanding of the different packages available, there is a wide variety out there and it can easily become confusing. Remember unless you are after Bleeding edge then your package manager should always be your first port of call.

Getting Started with Linux Introduction - Why use Linux? (http://benscomputer.no-ip.org/Articles/getting_startedintro.html)

Getting Started with Linux Introduction Part 2 - Hardware (http://benscomputer.no-ip.org/Articles/getting_startedintro2.html)

Getting Started With Linux Part One (http://benscomputer.no-ip.org/Articles/getting_started.html)

Getting Started With Linux Part Two - Installing Software (http://benscomputer.no-ip.org/Articles/getting_started2.html)