Author: Amro Diab – adiab@linuxmail.org

Creating a custom Linux Distribution based on Progeny 2.0 Development Edition

Revision Date: 08 March 2005

This HOWTO assumes you have a basic understanding of XML and Debian concepts
It is recommended that you read the componentized Linux guides prior to reading this HOWTO

Preparing The Base System

Download and Install Progeny Debian from
http://archive.progeny.com/progeny/linux/iso-i386/

You only require the first two images – the binary ISO's
You will need all packages except those under the server category

Next you will need to create your repository.
Login as root. Create a folder in your root '/' partition – call it the name of your new distribution – e.g. sgnix

Within that directory create the following directories: dists, build, and work

```
mkdir -p /sgnix{dists, build, work}
```

Now download the progeny components into the structure:
```
rsync -av rsync://archive.progeny.com/progeny/linux/dists/cl/
/Sgnix/
```

Creating Components

Locate or create any additional components or packages you wish to add. You can find a good selection at http://cvs.linex.org/linex2004/dists/cl/
More appropriately, you should acquire the packages from the progeny archives and build your own components. To browse through what is available look at http://archive.progeny.com, but you will be downloading the required packages another way

If, for example, you wished to make a thunderbird component to add to your repo, you would do the following:

a) create a componenty structure in your repo.
```
cd /Sgnix/dists/cl
mkdir -p mozilla-thunderbird/{binary-i386, source}
```

b) Copy over a sample comps.xml file to use as a base, the hello package comps.xml file is created for this purpose. Now edit it to include the packages you require for your new component and have the relevent titles and descriptions etc.

c) Next run comps2repo to populate the binary directory.
Now from the /Sgnix directory generate the Packages file.
```
cd ./sgnix; dpkg-scanpackages dists/cl/mozilla-thunderbird/binary-
i386 /dev/null > dists/cl/mozilla-thunderbird/binary-i386/Packages
```

Make sure there is no trailing '/' after the first binary-i386, as this will create a double // in the Packages file. Also note that Packages is case sensitive - it DOES start with an uppercase P

Be sure that dependencies are satisfied within the component, or are available within other components.

Configuring the installation procedure

Next, you will need to acquire the PICAX XML scripts using subversion and then configure them to suit your needs

```
svn co svn://svn.progeny.com/cl/debian/trunk /sgnix/build
```

Now you will need to edit the xml files to configure the way anaconda is installed.
You should find the following files:

picax.xml
comps.xml
pkglist.xml

Firstly Edit the picax.xml file

```
vi picax.xml
```

In-between the cd-label tags insert your Distro name or whatever you wish the cd to be called – eg Sgnix
Change inst-comps to the path of your comps file e.g. /SGNIX/build/comps.xml
Change inst-help-path to the path of your help directory eg /Sgnix/build/anaconda-help
The inst-* options should point to the appropriate xml file path eg:

```
   <inst-comps>/Sgnix/build/comps.xml</inst-comps>
```

The source option will make a source cd, if required.

Add repository components to include the components you wish to have installable from the cd.

E.g. <repository component="games" distribution="cl" />
And remove the one's you no longer require.

Make sure you include all of the required components as failure to do so will result in runtime errors in anaconda.
Remember to add any components that you created or downloaded.

Now edit the comps file

```
vi comps.xml
```

An easy way of maintaining this file is by starting with the comps-skel.xml which should be in your build directory and merge in exsiting comps files to create the final version.
You can achieve this with the compsmerge tool.
E.g.

```
cd /Sgnix/build
compsmerge ./comps-skel.xml ../dists/cl/mozilla-firefox/comps.xml
../dists/cl/mozilla-thunderbird/comps.xml ../dists/cl/gnome-
2.0/comps.xml ../dists/cl/xfree86-4.3/comps.xml > ./comps.xml
```

The above will merge the comps.xml files from firefox, thunderbird, gnome, and Xfree86 with the skeleton file and create a new combined file called comps.xml in the build dir.

You will need a group for every component that you specified in picax.xml. For each component you will need to specify the modules that you wish to be available and how they are to be available.
If you wish the component to be visible to the user in anaconda then change the uservisible tag data to true. The default tag specifies whether or not it will be selected to be installed by default. It is important to note the difference between the id and name tags. Id specifies the name of the package found in your repository and nname specifies the name to be shown to the user.

The Mandatory and Optional properties specify whether or not it is possible to select/deselect the package at run time. Obviously packages which are required for the basic operation for the distribution such as Bash etc. should be made mandatory.

Example snippet:
```
- <group>
  <id>basic-desktop</id>
  <uservisible>true</uservisible>
  <default>false</default>
  <name>Basic Desktop</name>
  <description>X window system</description>
- <packagelist>
  <packagereq type="mandatory">x-window-system</packagereq>
  </packagelist>
  </group>
- <group>
```

The category section towards the bottom allows you to specify the package structure as shown in anaconda.

Finally there's the pkglist file
You will probably not want to edit this, it specifies what you need at runtime for the installer – so unless you need to have some extra functions, or remove the anaconda GUI for example, I would leave it as it is.

Creating and customizing anaconda from source

Next you will need to acquire, compile and install the lastest versions of anaconda and picax.

```
Cd /Sgnix/work
apt-get install subversion-tools
apt-get install anaconda picax
cd /Sgnix/work
svn co svn://svn.progeny.com/anaconda/trunk/ anaconda
svn co svn://svn.progeny.com/anaconda/picax/trunk picax
```
this will create 2 directories – anaconda and picax.

First of all go into the anaconda dir and edit constants.py
You will find a variable called BETANAG – change the value to 0.

Then you will need to compile.
```
dpkg-buildpackage
```

do the same in the picax directory

```
cd ../picax; dpkg-buildpackage.
```

If compilation is successful you will be left with 2 deb files in the /Sgnix/work directory

Copy these into the correct place in your repo

```
cp /Sgnix/work/*.deb ../dists/cl/cl-dists/binary-i386
```

Make sure you remove any previous versions of the same packages, both source and binaries

You may need customize anconda using the source. It is built using Python. The most important files are

```
./language.py, ./timezone.py, ./dispatch.py

./iw/bootloader_advanced_gui.py
./iw/osbootwidget.py

./textw/language_text.py
./textw/bootloader_text.py
```

I, personally, edited the source without any previous knowledge of python, but I'm guessing you would need to know what you're doing if you will be hacking away at the code at any high level.

The dispatch file specifies which forms or screens are to be shown and the order in which they are displayed. In the installsteps section you can add or remove sections, as appropriate.

Language and timezone will allow you to specify defaults for the appropriate values in the relevent screens.
the boot files allow you to modifiy boot settings eg. default label in grub or lilo.

Remember anaconda comes in both text mode and GUI mode, so any changes you make in one you will probably want to make in another.
Before compiling your distro, you will need to:

a)Build the package
make sure you are in the source root.

```
dpkg-buildpackage -b
```

b) Move the deb package into the binary directory.

```
mv ../*.deb ../../binary-i386
```

c)create the Package file

```
  cd /Sgnix
  dpkg-scanPackages dists/cl/cl-devel/binary-i386 /dev/null >
dists/cl/cl-devel/binary-i386/Packages
```

You will get a message listing the packages you have exported to the Packages file, and a warning that the override file is missing. You can safely ignore this message as long it is has successfully written the correct number of entries to the Packages file.

d)Install the current version on the base system.

```
  dpkg -i ./anaconda_*.deb
```

Customizing the Distribution

Now you waill need to create the customizations. First of all, you will need to get a package containing the progeny packages. Download everything from the following page into /Sgnix/work
http://archive.progeny.com/progeny/linux/debian/dists/cl/progeny/
in the source directory extract the 2 g'zipped tarballs. The progeny-branding package contains the images required for gnome, grub, gdm, bootsplash etc. These can be modified to your liking. There is also a file called progeny.xml for the default theme. Once you've modified it to your requirements go back to the root of the source directory and recompile the package in the same way. It can be renamed, if you wish.
First of all move the changelog into the source directory
And then compile

```
Dpkg-buildpackage
```

This will build the deb file as well as the dsc file.
You can then move that to an appropriate location in your repo – eg /Sgnix/dists/cl/branding/binary-i386.

The source can also be added to something like

```
/Sgnix/dists/cl/branding/source
```

then extract the sources.gz file

```
gunzip sources.gz
```

so that it can be recognized by picax
modify as appropriate

The same should be done with the progeny-defaults. This contains some scripts and variables etc.

Be sure to add the newcomponent to the picax.xml file under repository distribution and also add to the comps.xml as a group with both of the packages as mandatory as they contain dependencies for other components.

Creating the Images

Move to the build directory
```
cd /sgnix/build
```
And execute the following command

```
Picax –build-config=picax.xml /Sgnix/
```

This should create an iso in /sgnix called img-bin01.iso, and so on, if it required further images.
The build directories will be named /Sgnix/bin1/ Sgnix/bin2, and so on, if for any reason you need to rebuild the ISO's then you will need to remove, or at least rename these directories first, otherwise picax will fail. I would also recommend removing any existing ISO's to avoid confusion. You will probably receive quite a few dependency errors at first, especially if you have a large and heavily edited comps.xml file. You may want to redirect the pixar output to a file so that you can examine it later, as it tends to be very large, even without any problems.