Development for Multiple Linux Distributions mini−HOWTO

Ed Hill

<ed@eh3.com>

2001−03−01

Revision History

Revision 0.9.1	2002−08−14	Revised by: EH3
Small update for using X11 (sockets).		
Revision 0.9	2001−12−03	Revised by: EH3
Initial version.		

This document outlines a quick hack to aid in developing and testing user−space programs for multiple Linux distributions.

Table of Contents

1. Introduction

As a developer, there is occasionally a need to code and/or test programs on multiple Linux distributions. This mini−howto outlines a neat chroot trick that can, in many cases, obviate the need to reboot into different Linux distributions to do testing or development.

In a nutshell, the "trick" is to place all the files from a particular linux distribution into a single directory and then chroot into that directory to develop/test/debug your program. This approach will work provided that:

1. The kernel version of the "main" or "primary" distribution that you are running is (at least somewhat) compatible with the "secondary" or chroot'ed distributions, and
2. Your application is primarily user−space (ie. no kernel modules) and does not depend intimately on any particular kernel features (ie. `/proc` behavior).

1.1. Copyright and License

2. Setup

We assume that you already have:

1. One "main" or "primary" Linux distribution installed that is running a 2.4−series or newer kernel,
2. a free partition with 2−−4Gigs available, and
3. sufficient disk space for the secondary installs (and any subsequent development/testing work).

Given those assumptions, the following outline is one way to setup the multiple secondary Linux distributions that will be used for development, testing, or other purposes:

1. First, create or locate an unused partition on one of your hard drives that is large enough (usually 2−−4Gb is sufficient) to do a basic install of one of the secondary Linux distributions.
2. Install the secondary distro into this partition but do not add it to your boot configuration. Note that this install need only include the packages that your target application (or tests) require. For instance, you may be able to ignore applications such as the X server or other space−hogs.
3. Reboot your primary Linux distribution and mount the partition containing the freshly−installed "secondary" distro. Copy (preferably using `tar -cp` or some other method that preserves permissions) all the files from the secondary distro to a location such as `/opt/distros/DISTRO_NAME`.
4. Repeat steps 2−−3 for any additional distributions that you wish to install. The result should be a directory structure resembling:
   ```
   /opt/distros/redhat_6.2/
                suse_7.2/
                mandrake_8.1/
                debian-potato/
                slackware_8.0/
   ```
   where each directory contains the complete set of files resulting from each distribution install.

3. Using the Secondary Distributions

With the secondary distributions installed, the steps to use them are:

1. It is important to use a 2.4−series or newer Linux kernel in the "primary" Linux distribution in order to take advantage of the multiple mount points feature that the 2.4−series permits. For many tasks, the `/proc` and `/tmp` filesystems will have to be remounted in the secondary distribution using:
   ```
   mount --bind /proc /opt/distros/redhat_6.2/proc
   mount --bind /tmp /opt/distros/redhat_6.2/tmp
   ```
2. Also, it can be helpful to remount (rather than copy) the source tree from the primary to the secondary distro:
   ```
   mount --bind /home/USER/src/PROJECT /opt/distros/redhat_6.2/USER/src/PROJECT
   ```
3. Become root and use:
   ```
   xhost +localhost chroot /opt/distros/redhat_6.2
   /bin/bash
   ```
   to obtain a shell with one of the secondary distros. Note that the `xhost` command is only necessary if you intend to use X−windows applications.
4. Finally, create a user (if necessary) within the `chroot`−ed shell and develop, build, and/or test your application within this "separate" Linux distribution! Not that you may also have to specify your `DISPLAY` environment variable if you'd like to use X applications.

Done! You now have a shell that is, for all practical purposes, running within the secondary Linux distribution of your choice.

4. Credits

The ideas contained in this mini−HOWTO are not originally mine. They are culled from posts by Ben Reed (of OpenNMS) to one of the TriLUG mailing lists. I found the information to be so useful that I wanted to document the idea for others.