Upgrading Your linux Distribution mini−HOWTO

Table of Contents

Table of Contents

Upgrading Your linux Distribution mini−HOWTO

Greg Louis, `glouis@dynamicro.on.ca`

v1.11, 6 June 1996

Hints and tips on upgrading from one linux distribution to another.

7.Make separate backups of each group of files you want to preserve.

8.Prepare root and boot floppies for the new installation.

9.Format floppies for the temporary kernel and the final build.

10.Inhibit logins and back up the /root and /home trees.

11.Boot from the new installation's boot and root floppies.

12.Delete the linux partitions with fdisk and recreate them.

13.Run the new linux installation.

14.With the new linux system booted from the hard disk, edit /etc/fstab

15.Restore configuration data to the /etc directory and subdirectories.

16.Configure and rebuild the linux kernel.

17.Restore the stuff from the backups you made earlier.

18.Review security.

19.Enable logins.

20.Sorry, but once again:

21.Acknowledgements

1.IMPORTANT!!! Disclaimer and Copyright

The procedure to which this document attempts to be a guide is inherently dangerous to the programs and data stored in your computer. You carry out any such procedure entirely at your own risk. The steps described in this document worked for the author; there is no guarantee that they will work for you, nor that you can attempt to follow them without serious damage to your computer's programs and/or data. You are entirely on your own in any use you may make of the information presented herein, and the author shall not be liable in any way whatsoever for any damage or inconvenience of any kind that you may suffer in so doing.

Comments and questions may be directed to the author. Especially welcome, for use in future revisions, are accounts of successful upgrades of complex systems.

2.Changes since version 1.1

- Added this history section
- Added Zoltán Hidvégi's suggestion re mtime and ctime. Thanks, Zoltán!
- Added an Acknowledgements section

3.Introduction

3.1 How to slay and reincarnate your linux box!

The purpose of this document is to offer tips to help you through the destruction and reinstallation of a linux system. It's not a foolproof cookbook by any means; but I hope it will serve as some indication of what you need to think about, and of the order in which to do things. It would have been a help to me, if someone else had written something like this before I did my first upgrade; so I hope it will be a help to you, if you have a linux machine to rebuild.

Don't take it as gospel, though: your mileage will almost certainly vary. Even the directory names in this document may be different from the ones you need to use; some people have /usr/home instead of /home, for example; others call it /u, and some (delicate shudder :) even put all their users directly under /usr itself! I can't be specific about your system, so I've just used the names the way they are in mine.

You'll also notice that I use Slackware distributions, and that I assume you've enough RAM and hard disk space to install linux kernel source and build your own kernel. If your system is different, some of my recommendations won't apply; but I hope you'll still find the general outline to be of assistance in your rebuild project.

3.2 Why would anyone want to do that?

Good question! If it can possibly be avoided, don't do it! (That's the single most important recommendation in this whole guide!!!) But there are times when you may have to.

For example, I installed a 4Gb hard disk and then found out that Slackware 2.0 vintage linux didn't know a hard disk could have more than 2Gb, and it got horribly confused. So I had to upgrade to the then−current Slackware 2.3. That upgrade was a gruelling experience, and it's part of the reason I'm writing these notes. I did just about everything wrong, and only good luck and the fact that I had another running linux box beside me saved me from disaster.

As another example, I found that I just couldn't succeed in building a working a.out linux kernel in the 1.3 series, using an out−of−the−box Slackware 2.3 installation (another machine, not the one I botched before). I took the plunge, bought Slackware 3.0 on CDROM and converted to ELF. This time the reinstallation went better, thanks in part to the previous bitter experience, and it served as the source of most of the ideas I'm offering you here.

3.3 Do you have to ``destroy and reinstall?''

It's safer, oddly enough. If you install over top of an existing linux system, chances are you'll have a mixture of old and new binaries, old and new configuration files, and generally a mess to try to administer. Wiping the system clean, and then putting back only what you know you need, is a drastic but effective way to get a clean result. (Of course we're talking about installing a whole new linux distribution here, not about upgrading one or two packages! The best way to avoid having to do a full reinstallation is, precisely, to keep the individual bits −− especially gcc and its libraries, and binutils −− current. If the stuff you use is reasonably up−to−date, and you can keep it so by bringing in, and if need be compiling, new code from time to time, then there's no need for a mass upgrade.)

As Patrick Volkerding points out (he too recommends the wipe−it−clean procedure for upgrades), installing ELF on top of a running a.out system is a recipe for disaster; at least, if you know enough to try it, you needn't read this guide!

Even without that complication, though, you're better to build from scratch.

3.4 How long will it take?

Depends, of course, on how complex your system is. But I figure that, for the successful upgrade (the other one? −− don't ask! :) I spent about ten hours making backups, six hours rebuilding the system to the point where I could enable logins, and another half day or thereabouts restoring the less−crucial stuff. As time passes I keep discovering little things that still aren't exactly as I want them −− I fix these as they're encountered −− but in the main, twenty hours' work should suffice for a reasonably complex rebuilding job. Maybe less if you're reinstalling from hard disk (I used CDROM) or more if you need to install from floppies. Maybe less if you've got a fast Pentium, more if it's a 386. You get the idea.

So much for the introduction. Here's how to set about it, once you've decided it must be done. Arm yourself with fortitude and Jolt or whatever, and:

4.Write down everything you do.

It's extremely valuable to have a record of what you've done in the process of preparing for, and carrying out, the changeover. Especially important is a list of the backups you'll be making in preparation for the destruction of your existing system.

5.Make a full backup of the existing system.

Generally speaking, backups tend to be written on media that are sequentially accessed. That being so, you won't want to use this complete backup for restoring significant numbers of files; it's got too many files on it that you don't want. It's better to create small backups of individual segments that you know you're going to restore in their entirety. I'll list a bunch of examples later.

Why then should you start with a full backup? Two basic reasons: first, in the event of a catastrophic failure installing the new system, you'll have a way to get back to the starting point with minimum pain. Second, no matter how carefully you prepare for the new installation, there is a very large chance that one or two important files will be overlooked. In that case the clumsiness of restoring those one or two files from the full backup set will be preferable to the inconvenience of doing without them.

To save time and space, if you've still got the distribution medium for your old linux version, you might want to back up only those files the mtime or ctime of which is more recent than the date of the original installation.

6.Back up /etc and its subdirectories on one or more floppies.

This is the other extreme: you won't be restoring these files (for the most part, anyway); you'll be comparing them with the new ones that get created during installation. Why? Because the new ones may have data that the old ones didn't, or may express the old data in new ways. Changes in protocols, addition of new tools, or implementation of new features in existing tools may all dictate changes in the formats of the configuration files and startup scripts that the /etc tree contains, and you'll very likely have to edit your old data into these files so as to preserve the new formats and take advantage of the improvements.

7.Make separate backups of each group of files you want to preserve.

This is the most variable part of the job, and all I can really do to help is to describe what I did in my system, in the hope that it will serve as a rough guide. Basically, you want to look at every directory that contains any

- files that aren't part of your standard linux installation, or
- files that are actually newer than the ones you'll install when you do your new linux installation

and separate out only those files that you want to carry over.

(Another possible strategy is to back up all files with mtime or ctime more recent than the day of the previous linux installation, as mentioned above, and then restore from that. If you do that, you have to take into account that the new linux distribution may contain versions of some files that are newer still than the ones you saved.)

In my case, I ended up making a .tgz file on the backup medium for each of

- /usr/lib/rn
- /usr/lib/smail
- /usr/lib/trn (the rest of /usr/lib would be reinstalled)
- /usr/local/src
- /usr/local/bin
- /usr/local/lib
- /usr/local/lpfont
- /usr/local/man
- /usr/local/sbin

- /usr/local/thot (there were other /usr/local files I didn't need)
- /usr/openwin
- /usr/src/lilo−17 (because my new Slackware still had version 16)
- /usr/src/linux−1.2.13 (because I'd done some customizing)
- /usr/X11R6/lib/X11/app−defaults
- /usr/X11R6/lib/X11/initrc (the rest of Xfree86 was to be reinstalled)
- /var/named
- /var/openwin
- /var/texfonts

My machine was relatively easy in that there were no spool files to worry about. I don't run a news spool on this box, and since there are only two users, it was easiest just to get all the mail read before shutting down. Otherwise, /var/spool directories would have had to be backed up at the last minute. (And, of course, the news library and site directories!)

8.Prepare root and boot floppies for the new installation.

Details of how to do this will be found in the installation guide for your new distribution.

9.Format floppies for the temporary kernel and the final build.

You'll need two, one floppy for each.

After all that's done, you're ready for the Big Moment. The next step removes the system from production.

10.Inhibit logins and back up the /root and /home trees.

This is the last thing to be done on the old system before you destroy it, so as to carry forward the most current user and root information.

11.Boot from the new installation's boot and root floppies.

12.Delete the linux partitions with fdisk and recreate them.

The installation guide will explain how to set about doing this, which will destroy the old system. From now on you're dependent on the quality of the backups you made in the earlier steps! You have been warned!

13.Run the new linux installation.

There are already several good documents describing how to do this, so I'm not going into any detail. Continue from here when the new system can boot from its hard disk.

Along the way, be sure to make a floppy that you can boot as well, since the kernel that the linux setup installs has to be replaced and accidents can happen during that process. Be sure to install the development packages and the kernel source.

14.With the new linux system booted from the hard disk, edit /etc/fstab

and add your swap partition. Then run the command "swapon −a". I don't know why, but Slackware setup doesn't offer to do this for you if your swap partition exists already. Then, when you boot your new system and the rc.S script tries to turn swapping on, it can't find the partition in the fstab file and swapping doesn't get enabled. This step fixes it.

15.Restore configuration data to the /etc directory and subdirectories.

As described above, you can't just copy all of the old files back into /etc and expect things to work properly afterward. Some files you can do that with; for example, /etc/XF86Config (as long as you're using the same version of Xfree86 −− and the same video hardware −− in the new installation as you did in the old). For the most part, though, it's best to use diff to compare the old and new files before doing any copying. Watch out especially for significant changes in the files in /etc/rc.d, which may require you to reestablish your old configuration by hand editing, instead of by copying the old rc scripts from your backup. Once it's all done, reboot.

16.Configure and rebuild the linux kernel.

Even if you don't absolutely have to do this in order to get a kernel that supports your hardware, it's worth doing it in order to get a kernel that doesn't contain masses of drivers for stuff your machine doesn't have. For details, see the Kernel HOWTO. Install the rebuilt kernel on a floppy at first; once that boots ok, install on the hard disk, run lilo if you're using it, and reboot.

17.Restore the stuff from the backups you made earlier.

Some of the binaries may need to be reinstalled from the source directories; I had to do that with lilo, for example, since my version was newer than the one on the Slackware installation and I hadn't bothered to save the binary from /sbin. You'll want to check through your restored programs and confirm the existence and correctness of configuration files, libraries and so on. In some cases, you may have to restore things in a specific order; you did make notes during backup, didn't you? ;−)

18.Review security.

Check file permissions and directory permissions to be sure that access is neither too restricted nor too easy. I find that Slackware tends to lean toward a more open environment than I like, so I go around changing 755's to 711's for binaries in the .../bin directories and stuff like that. Or even 700's in the .../sbin ones. Especial care is needed if you've carried over an ftp server; but then, if you were running an ftp server, you probably thought of that already. :)

19.Enable logins.

You're up and running. Over the next little while, there'll probably be details to clean up; but the bulk of the work is done. Enjoy!

20.Sorry, but once again:

USE THIS INFORMATION AT YOUR OWN RISK!

(See the disclaimer at the start of this document.)

21.Acknowledgements

Thanks for contributing to the content of this mini−HOWTO are gratefully tendered to Zoltán Hidvégi.