Saving Space mini−HOWTO

Table of Contents

Saving Space mini−HOWTO

By Guido Gonzato, mailto:guido@ibogeo.df.unibo.it

v1.0.1, 7 April 1999

This mini−HOWTO gives you directions for squeezing your Linux installation into the least possible space. It's particularly aimed at notebook users.

1.Introduction

I've got a notebook I installed Linux on, beside Windows 95 that was pre−−installed. I squeezed the Windows partition to 500 Mb, making space for a 240 Mb Linux one. Small though the latter may seem, I've managed to install a fairly complete Linux system on it, based on Red Hat 4.1 and some magic to save as much space as possible.

If you're in need for space, the indications you'll find in the following sections will free up a considerable chunk of hard disk. The only caveat I'm giving you is: don't blame me if something goes wrong! We'll be using a couple of programs that worked fine for me, but are inherently dangerous. You've been warned.

2.Software requirements

You need:

- a fully working Linux system (any version should be OK);
- the common `gzip` compression utility, or alternatively `bzip2` that you find on ftp://sunsite.unc.edu:/pub/Linux/utils/compress; this tool compresses better than `gzip`, but it's also a wee bit slower and consumes a lot of memory;
- the `upx` executable compressor, whose home page is at http://cdata.tvnet.hu/~ml/upx.html and http://wildsau.idv.uni−linz.ac.at/mfx/upx.html;
- the `zlibc` package, available on ftp://sunsite.unc.edu:/pub/Linux/libs/compression. It's called `zlibc−X.X.tar.gz`, where `X.X` is the latest version.

There are other executable compressors. One is `gzexe` (forget it), while `tcx` was probably the best before `upx` became available; yet another is `tzx`, in theory better than `tcx`. The thing is, it screwed everything up when I tried it on a spare PC, though I'm not sure if I did something wrong. You had better stick with `upx`: it's reliable, efficient, and very handy.

There are kernel patches that provide transparent file system compression a la Stacker, but as of this writing none of them has yet a reputation for stability and reliability. For safety's sake, steer clear.

3.The procedure

3.1 Removing the Kernel Sources

The kernel sources take up more than 20 Mb, and you may want to remove them. If so, I suggest that you compile a new kernel that is tailored to your machine once and for all, then get rid of the sources. Take care, though.

I *don't* recommend that you remove the kernel sources unless you are sure that your machine is properly configured. Besides, you need the kernel #includes to compile C programs. Think twice!

If you do decide so, *do not* remove the include/linux tree unless you know you'll never compile applications on your machine.

3.2 Pruning Out Applications

Now, decide which applications you *really* need. Some may prove redundant: for instance, are you sure you can't live without `emacs`? You could use `jed` instead.

It's up to you to decide what you want to keep. Some general points:

- `gcc` is a fairly large package. It's needed to recompile the kernel and all the applications for which there are no pre−−built Linux binaries. It's also needed, of course, if you write your own C or Fortran (with `f2c` or `g77`) programs; consider your needs before removing it. Other compilers like `lcc` are fine, but not up to `gcc` level. I suggest that you keep it;
- X11 is awfully large, but it's a nice thing to have. If you decide you can't give it up, at least try to make do with as little as necessary: only the right X server, one simple window manager, only one `xterm`, no 100 dpi fonts, and so on;
- TeX and accompanying packages are very large indeed. Giving up LaTeX and sticking to plain TeX saves a lot of space; giving up X11 previewers like `xdvi` and `ghostview` is possible if you use `dvitty`, `dvivga` and some such. Eliminating the need for X11 dviware could even make X11 redundant;
- games are never ``necessary''.

3.3 Stripping Binaries

Let's start reducing the size of binaries. Move to /usr/bin and issue the command

```
machine:/usr/bin# strip *
```

which will do away with symbols embedded in binaries. Repeat this step in /usr/X11R6/bin/ and other directories containing executables you may have (don't forget to locate TeX's and `gcc`'s binaries), but *do not* run it under /sbin, /bin, or /usr/sbin/ if you value your installation!

3.4 Compressing with `upx`

First, install `upx` and read its documentation. Then go to /usr/bin and run it with the command `upx *`; it will compress all executables, including suid ones (`tcx` wouldn't). Repeat this step in other directories as seen above.

Remember to compress the executables whenever you install a new package!

3.5 Gzipping around

There are lots of other files that can be compressed once and for all. Let's start with /usr/doc/; move to this directory and issue the command

```
machine:/usr/doc# find . −type f −exec gzip −9 {} \; 2> /dev/null
```

Remember to compress the docs whenever you install a new package!

Repeat this step in the directory containing the documentation for TeX (on my system, /usr/lib/texmf/texmf/doc/. If you're *really sure*, remove these directories altogether.

Now, install `zlibc` and compile it. If your system is like mine, the compilation process will abort complaining about a missing (static) libc. Never mind; you'll find a file called `uncompress.o` that is what's needed. Move it to /usr/local/lib/ and add this line to your /etc/profile:

```
export LD_ELF_PRELOAD=/usr/local/lib/uncompress.o
```

Now, you can compress with `gzip` not only documentation, but also data files: the applications that use them will be able to use them nonetheless. In theory, the trick should work with most applications, but in practice your degree of success may vary. Mine was rather unimpressive.

4.A Real Life Example

This is what I got applying the above procedure to one of my machines. Before the treatment, `df` reported I used 398,798 1024−−blocks:

- I didn't remove the kernel sources and the kernel headers;
- I uninstalled several applications and all of the games, but I left X11, X11 development, C and Fortran development, Tcl/Tk, networking tools, and a few other standard applications. `df` reported 244,668 used blocks;
- I ran `upx` on /usr/bin, /usr/X11R6/bin, /usr/lib/texmf/bin/i586−linux, and /usr/lib/gcc−lib/i386−linux/2.7.2.1. 226,270 used blocks;
- I compressed the documentation under /usr/doc and /usr/lib/texmf/texmf/doc: 198,745 used blocks.

To sum up, I started with 398,798 blocks and finished with 198,745. Think of the stuff you can shove in those 200,000 spared blocks! I would have saved even more if I had used `bzip2` instead of `gzip`.

On average, if you are careful from the beginning and install only the necessary applications, compressing executables and documents will save you some 20 Mb. On a notebook, this can be a lifesaver.

5.The End

5.1 Copyright

5.2 Disclaimer

``Saving Space mini−HOWTO''was written by Guido Gonzato, REMOVE_MEguido@ibogeo.df.unibo.it.

This document is provided ``as is''. I put great effort into writing it as accurately as I could, but you use the information contained in it at your own risk. In no event shall I be liable for any damages resulting from the use of this work.

Feedback is welcome. For any requests, suggestions, flames, etc., feel free to contact me.

Enjoy Linux and life,

Guido =8−)
