LINUX—a free unix-386 kernel

Linus Torvalds (torvalds@kruuna.helsinki.fi)

October 10, 1991

1 Introduction to LINUX

1.1 What is LINUX?

LINUX is a free unix-like kernel for 386-AT computers, coming with full source code. It is meant for hackers/computer science students to use, learn and enjoy. It is written mostly in **C**, but parts of it are in gnu-format assembler, and the boot-sequence is in intel 086 assembly language. The **C**-code is relatively ANSI, with a few GNU enhancements (mostly `__asm__` and `inline`).

While there are a number of unices available for 386 computers, most of them cost a lot of money, and come with no source. Thus, they are ideal for actually *using* your computer, but if you want to learn how they work, you are f--ked.

There are also a few unices available with source. Minix, the learning tool written by Andrew S. Tanenbaum, has been used in universities as a teaching tool for years. The BSD-386 system comes with source, but has a restrictive copyright and costs a lot of money ($995 is the starting price, I think). The GNU kernel (Hurd) will be free, but is currently not ready, and will be too big to understand and learn.

LINUX most closely resembles Minix, in that it is small and not very sophisticated, and thus easy (well...) to understand. LINUX was also written under Minix, so there are quite a bit of similarities, and any Minix hacker will feel relatively at home with LINUX. None of the Minix code was used in the project though, so the Minix copyright doesn't cover the new system. It also is *completely free*, and has a very loose copyright. Thus there is no need for megabytes of diffs like under Minix.

1.2 The LINUX copyright

While being freely distributable, I do restrict the use of LINUX in a few ways:

- You may freely copy and redistribute the source and binaries, **as long as:**

  - complete source is available. Thus binaries may not be distributed by themselves, even if you have made changes to them.
  - you do not profit from the distribution. In fact even "handling costs" are not acceptable.
  - you keep the appropriate copyrights intact.

- You may change the source to your liking, but if you distribute parts of the new system (or just binaries), all the new code must be included.
- You may make small excepts from the code without including copyrights. This is up to you, but a reference to me or the code would be appreciated.

This should be loose enough not to cause any worry in anybody using or expanding the system. If you have a friend who really doesn't want the source, but just a working binary, you may of course give it to him without worrying whether I will sue you. Keep it between friends, though.

1.3 Hardware/software needed to get LINUX running

LINUX was written on a 386-AT running Minix. As LINUX is a real operating system, and goes directly to the hardware to do things, you have to have a very similar system to get it going without problems:

- 386-AT (PS/2's are different enough that things won't work)
- VGA or EGA screen hardware.
- Standard AT hard-disk interface, IDE disks work fine (in fact that's what I use).
- Normal real-mode BIOS. Some machines seem to use virtual-86 mode to run the bootup program, and on such machines LINUX won't boot up and run correctly.

While LINUX will be expanded to be a self-sufficient system, Minix-386 is currently needed to get the ball rolling. You need Minix to make the initial root file system, and to compile the OS binary. After that LINUX is a self-sufficient system, but Minix is recommended in order to make file system checking (`fsck`) and to recompile the system after making changes.

1.4 Getting LINUX

LINUX can currently be gotten by anonymous `ftp` from `'nic.funet.fi'` in the directory `'/pub/OS/Linux'`. This directory contains the full source to the operating system, as well as a couple of binaries so that you can actually use the system.

NOTE! The binaries are mostly GNU software, and are under a stricter copyright (the GNU copyleft) than the LINUX sources. Thus you may not redistribute them without distributing their source, found in `/pub/GNU`. See any GNU software package for more information on the GNU copyleft.

The various files found in this directory are:

- `linux-0.03.tar.Z`—complete source to the operating system, in a 16-bit compressed tar archive.
- `Linux.tex`—LaTeX source for this file.
- `bash.Z`—bash binary to run under LINUX. This binary should be put under tha name `/bin/sh` in the file system reserved for LINUX (see installation).
- `update.Z`—update binary, to be put in `/bin/update`.
- `gccbin.tar.Z`—GNU cc binaries needed to get a working compiler. This tarred archive contains the compiler, loader, assembler and support programs (nm, strip etc). It also contains a small library sufficient for most programs.
- `include.tar.Z`—include-files necessary to get gcc working.
- `unistd.tar.Z`—source to the unistd library routines (ie system call interface). With this you can build a bigger library by using system-independent library sources.
- `utilbin.tar.Z`—binaries to various GNU utilities, including GNU fileutils, make and tar. Also contains the emacs-clone `uemacs`.
- `README, RELNOTES-0.01, INSTALLATION`—ascii files containing some (somewhat out-of-date) information about LINUX.

The absolute minimum needed to get a system going is the OS source and the bash and update binaries. You won't be doing much with just these though.

2 Installation

After you have gotten the necessary LINUX files, you need to compile the system and make a root directory. The necessary binaries need to be put in the root file system. Do this:

1. Back up your software. While LINUX never has destroyed any of my files, nothing is certain. Better safe than sorry.

2. Choose/make a standard Minix HD-partition to be the new LINUX root file system.

3. Make the necessary device nodes on the new root. LINUX uses the same type of nodes as Minix, so use the Minix `mknod` command to make the following devices:

   - `/dev/tty`
   - `/dev/tty[0-2]`
   - `/dev/hd[0-9]`

   Node numbers are the same as in Minix.

4. Move the necessary files to the new root partition. The files should be in the following directories:

   - `/bin`:
     - `sh`, ie `bash.Z`.
     - `update`
   - `/usr/bin`:
     - Contents of `utilbin.tar.Z`
   - `/usr/include`:
     - Contents of `include.tar.Z`
   - `/usr/local/lib`:
     - Contents of `gccbin.tar.Z`, excepting `gcc`
   - `/usr/local/bin`:
     - `gcc`
     - links to files in `/usr/local/lib` of your choice. I link `ld`, `as`, `nm`, `strip` and `size` to their counterparts in `/usr/local/lib/gcc-XXX`.
   - Edit the `linux/include/linux/config.h` file for your system. This file contains the system-specific information: memory space, disk types, root partition number (again the numbering is the same as in Minix), keyboard type (currently just US and Finnish) etc.
   - Compile the LINUX sources. A simple `make` should do the trick, after you have edited the `makefiles` to suit your system (ie, removed the `-mstring-insns` flag and changed the paths to suit you.) Users of `gcc` versions earlier than 1.40 will probably have to add `gnulib` to the '`LIBS =`'-line in the `makefile`.
   - Copy the resultant `Image`-file to a floppy (ie, `cp Image /dev/PS0` or similar.)

- Reboot with the new floppy. The startup screen should tell you that the system is booting (`Loading system ...`), then some vital root file system information (`xxx/XXX inodes/blocks free`), followed by a `Ok`. and the bash prompt (initially `bash#` if you have no `.bashrc` file).

Hopefully you now have a functioning unix, and you are logged in as `root`. LINUX currently has no `'init'` process, and as soon as you log out, the system will `sync` and just wait. Use the three-finger-salute to reboot your machine.

3 Things missing/incomplete in LINUX

While LINUX is meant to be a fully selfsufficient kernel, this is currently not the case. As already mentioned, you need Minix to set things up, and to check the file system once it's running. There are a number of other deficiencies:

- Incomplete hardware-support. Some of the standard features of an AT are not currently supported. Most notably are floppy disk drives, making using LINUX for real work (backing up etc) currently not possible. Also some of the features of the serial lines aren't yet implemented (hard-wired to 2400bps, no hang-up notification etc).

- Incomplete standard C library. The `libc.a` found in the gcc distribution package is not complete, and I'm very much interested in freely distributable library functions.

- Some of the system calls are not fully implemented. This concerns mostly "seldom-used" features like debugging (yeah, who needs it anyway, don't all your programs work the first time :-) and some other features.

- As mentioned, no `login` and `init` processes. Currently LINUX boots up in single-user mode, with the root as console-user. This is enough for some porting work, but not really practical.

- 387-support is not yet implemented, although some skeleton routines are present. The gcc-binary found on `'nic.funet.fi'` will correctly use soft-float (ie emulation function calls) for the four basic math operations. 387-support will materialize as soon as a 387 finds its way into my computer. Hopefully in a month or two.

- None of the important system-administration commands has yet been written for LINUX. These include things like `mkfs`, `format`, `fsck`, `mknod` etc. Some of these need kernel features not yet implemented (`format`, `mknod`), some just need to be written. As with the library, I'd welcome any freely distributable files.

As you can see, LINUX is as yet not a complete system. Your help is appreciated to make it better. I'm not interested in Minix-commands rewritten for LINUX, unless you have written them yourself from scratch. You are of course free (and encouraged) to use everything you have in your Minix-distribution for your own LINUX-system, but due to the Minix copyrights, they cannot be distributed to a wider audience.

Some of the probelms mentioned here will be fixed by me (ie serial lines/387/floppy support) as soon as possible, but I'm hoping to get help with the libraries etc. Bug-reports/patches and wish-lists will be appreciated, and if you actually have the patch to the problem, I'll try to implement it right away. Small changes will be sent out as patches to the mailing list and be set up on `'nic.funet.fi'`, but after heavy rewrites or bigger patches, the whole system will be updated at `'nic.funet.fi'`.

4 Porting software for LINUX

LINUX was designed to make porting relatively easy. Thus the full `termios`-implementation, and the somewhat `POSIX` library. The (admittedly relativly few) programs I've ported posed no problems.

Even though LINUX resembles Minix a great deal, Minix programs are not generally easier to port than programs designed for some other unix. Thus I wouldn't recommend starting from a Minix-version of a particular program, but instead trying to port the "virgin" program from scratch. Being closer to SYSV than BSD means that most programs port easily when given a `-DUSG` or `-DSYSV` flag.

One of the more difficult problem in porting can be missing library functions. These must be written by you, or copied from some other source (Minix being one possibility for those that have it). Alternatively, some programs (notably GNU) have various flags making it possible to define which functions aren't available (the GNU fileutils will compile quite nicely, once a sufficient number of `-DXXX_MISSING` flags are added to the `Makefile`.)

4.1 Programs already ported

These programs have already been ported to LINUX:

- GNU cc (gcc, cc1, cpp)
- GNU assembler (as386)
- GNU binutils (ld, ar, nm, size, strip, ranlib)
- GNU compress (16-bit)
- GNU tar

- GNU make
- GNU bash (Bourne Again SHell)
- GNU sed
- GNU bison (yacc-lookalike)
- GNU awk
- GNU fileutils (ls, cp, rm, mkdir, rmdir, tail etc)
- less
- uemacs

All of the above sources can be found on `'nic.funet.fi'` (mostly in `'/pub/gnu'`), and most of the LINUX-binaries can be found in the directory `'/pub/OS/Linux'`. All of these programs compiled without changes, even though gcc (cc1) has some enhancements of my own. Mail me for diffs/sources, but try first to compile them yourself.

Additionally I have reports that GNU diff compiles cleanly and works.

5 Technical help

LINUX currently has a mailing-list, which you can subscribe to by mailing to the address: **Linux-activists-request@niksula.hut.fi** and asking to be included into the list. You can then ask questions by mailing them to **Linux-activists@niksula.hut.fi**, which will duplicate your question/answer/whatever and send them to all persons on the list.

Note the difference between `Linux-activists` and `Linux-activists-request` — the first sends mail to all persons on the list, the second is used only to subscribe and unsubscribe from the list.

Naturally, you can also mail me directly at **torvalds@kruuna.helsinki.fi**. I'll try to answer all questions within a day or two.

Although `'nic.funet.fi'` will probably be kept reasonably up-to-date, there are a few problems with it (ie, I cannot personally get to the files, but have to go through a couple of persons). Thus people on the mailing-list will get patches/binaries faster if they ask for them.

6 Thanks

I'd like to thank the academy ...

Seriously, this system never would have seen the light of day or would have been much worse without the help of some others. Bruce Evans helped me find

the places needed to be changed in order for `gcc` to correctly handle floating point, and came with a lot of useful ideas/suggestions (and his Minix-386 was used to build the system). Also, Earl Chew's `estdio` package was used for the standard IO-library. More freely distributable packages like this!

Alain W Black and Richard Tobin made the gcc for Minix, without which I couldn't have compiled the thing. GNU made most of the programs I use under LINUX. Alfred Leung sent the US keyboard patches.

PS. "Thanks" to `wirzeniu@kruuna.helsinki.fi` for his "constructive" criticism and "witty" comments. He was also my first α-tester, and should be given a medal for courage.