Linux Old Documents

- RELNOTES
- INSTALLS
- CHANGES
- INFO-SHEETS

Assembled by
Zhao Jiong
gohigh@sh163.net

2002-10-2

1. RELNOTES-0.01

```
        Notes for linux release 0.01


        0. Contents of this directory

linux-0.01.tar.Z - sources to the kernel
bash.Z          - compressed bash binary if you want to test it
update.Z    - compressed update binary
RELNOTES-0.01       - this file


        1. Short intro


This is a free minix-like kernel for i386(+) based AT-machines.  Full
source is included, and this source has been used to produce a running
kernel on two different machines.  Currently there are no kernel
binaries for public viewing, as they have to be recompiled for different
machines.  You need to compile it with gcc (I use 1.40, don't know if
1.37.1 will handle all __asm__-directives), after having changed the
relevant configuration file(s).

As the version number (0.01) suggests this is not a mature product.
Currently only a subset of AT-hardware is supported (hard-disk, screen,
keyboard and serial lines), and some of the system calls are not yet
fully implemented (notably mount/umount aren't even implemented).  See
comments or readme's in the code.

This version is also meant mostly for reading - ie if you are interested
in how the system looks like currently.  It will compile and produce a
working kernel, and though I will help in any way I can to get it
working on your machine (mail me), it isn't really supported.  Changes
are frequent, and the first "production" version will probably differ
wildly from this pre-alpha-release.

Hardware needed for running linux:
    - 386 AT
    - VGA/EGA screen
    - AT-type harddisk controller (IDE is fine)
    - Finnish keyboard (oh, you can use a US keyboard, but not
      without some practise :-)
```

The Finnish keyboard is hard-wired, and as I don't have a US one I cannot change it without major problems. See kernel/keyboard.s for details. If anybody is willing to make an even partial port, I'd be grateful. Shouldn't be too hard, as it's tabledriven (it's assembler though, so ...)

Although linux is a complete kernel, and uses no code from minix or other sources, almost none of the support routines have yet been coded. Thus you currently need minix to bootstrap the system. It might be possible to use the free minix demo-disk to make a filesystem and run linux without having minix, but I don't know...

2. Copyrights etc

This kernel is (C) 1991 Linus Torvalds, but all or part of it may be redistributed provided you do the following:

- Full source must be available (and free), if not with the distribution then at least on asking for it.

- Copyright notices must be intact. (In fact, if you distribute only parts of it you may have to add copyrights, as there aren't (C)'s in all files.) Small partial excerpts may be copied without bothering with copyrights.

- You may not distibute this for a fee, not even "handling" costs.

Mail me at "torvalds@kruuna.helsinki.fi" if you have any questions.

Sadly, a kernel by itself gets you nowhere. To get a working system you need a shell, compilers, a library etc. These are separate parts and may be under a stricter (or even looser) copyright. Most of the tools used with linux are GNU software and are under the GNU copyleft. These tools aren't in the distribution - ask me (or GNU) for more info.

3. Short technical overview of the kernel.

The linux kernel has been made under minix, and it was my original idea to make it binary compatible with minix. That was dropped, as the differences got bigger, but the system still resembles minix a great deal. Some of the key points are:

- Efficient use of the possibilities offered by the 386 chip. Minix was written on a 8088, and later ported to other machines - linux takes full advantage of the 386 (which is nice if you /have/ a 386, but makes porting very difficult)

- No message passing, this is a more traditional approach to unix. System calls are just that - calls. This might or might not be faster, but it does mean we can dispense with some of the problems with messages (message queues etc). Of course, we also miss the nice features :-p.

- Multithreaded FS - a direct consequence of not using messages. This makes the filesystem a bit (a lot) more complicated, but much nicer. Coupled with a better scheduler, this means that you can actually run several processes concurrently without the performance hit induced by minix.

- Minimal task switching. This too is a consequence of not using messages. We task switch only when we really want to switch tasks - unlike minix which task-switches whatever you do. This means we can more easily implement 387 support (indeed this is already mostly implemented)

- Interrupts aren't hidden. Some people (among them Tanenbaum) think interrupts are ugly and should be hidden. Not so IMHO. Due to practical reasons interrupts must be mainly handled by machine code, which is a pity, but they are a part of the code like everything else. Especially device drivers are mostly interrupt routines - see kernel/hd.c etc.

- There is no distinction between kernel/fs/mm, and they are all linked into the same heap of code. This has it's good sides as well as bad. The code isn't as modular as the minix code, but on the other hand some things are simpler. The different parts of the kernel are under different sub-directories in the source tree, but when running everything happens in the same data/code space.

The guiding line when implementing linux was: get it working fast. I wanted the kernel simple, yet powerful enough to run most unix software. The file system I couldn't do much about - it needed to be minix compatible for practical reasons, and the minix filesystem was simple enough as it was. The kernel and mm could be simplified, though:

- Just one data structure for tasks. "Real" unices have task information in several places, I wanted everything in one place.

- A very simple memory management algorithm, using both the paging and segmentation capabilities of the i386. Currently MM is just two files - memory.c and page.s, just a couple of hundreds of lines of code.

These decisions seem to have worked out well - bugs were easy to spot, and things work.

4. The "kernel proper"

All the routines handling tasks are in the subdirectory "kernel". These include things like 'fork' and 'exit' as well as scheduling and minor system calls like 'getpid' etc. Here are also the handlers for most exceptions and traps (not page faults, they are in mm), and all low-level device drivers (get_hd_block, tty_write etc). Currently all faults lead to a exit with error code 11 (Segmentation fault), and the system seems to be relatively stable ("crashme" hasn't - yet).

5. Memory management

This is the simplest of all parts, and should need only little changes. It contains entry-points for some things that the rest of the kernel needs, but mostly copes on it's own, handling page faults as they happen. Indeed, the rest of the kernel usually doesn't actively allocate pages, and just writes into user space, letting mm handle any possible 'page-not-present' errors.

Memory is dealt with in two completely different ways - by paging and segmentation. First the 386 VM-space (4GB) is divided into a number of segments (currently 64 segments of 64Mb each), the first of which is the kernel memory segment, with the complete physical memory identity-mapped into it. All kernel functions live within this area.

Tasks are then given one segment each, to use as they wish. The paging mechanism sees to filling the segment with the appropriate pages, keeping track of any duplicate copies (created at a 'fork'), and making copies on any write. The rest of the system doesn't need to know about all this.

6. The file system

As already mentioned, the linux FS is the same as in minix. This makes crosscompiling from minix easy, and means you can mount a linux partition from minix (or the other way around as soon as I implement mount :-). This is only on the logical level though - the actual routines are very different.

NOTE! Minix-1.6.16 seems to have a new FS, with minor modifications to the 1.5.10 I've been using. Linux won't understand the new system.

The main difference is in the fact that minix has a single-threaded file-system and linux hasn't. Implementing a single-threaded FS is much easier as you don't need to worry about other processes allocating buffer blocks etc while you do something else. It also means that you lose some of the multiprocessing so important to unix.

There are a number of problems (deadlocks/raceconditions) that the linux kernel needed to address due to multi-threading. One way to inhibit race-conditions is to lock everything you need, but as this can lead to unnecessary blocking I decided never to lock any data structures (unless actually reading or writing to a physical device). This has the nice property that dead-locks cannot happen.

Sadly it has the not so nice property that race-conditions can happen almost everywhere. These are handled by double-checking allocations etc (see fs/buffer.c and fs/inode.c). Not letting the kernel schedule a task while it is in supervisor mode (standard unix practise), means that all kernel/fs/mm actions are atomic (not counting interrupts, and we are careful when writing those) if you don't call 'sleep', so that is one of the things we can count on.

7. Apologies :-)

This isn't yet the "mother of all operating systems", and anyone who hoped for that will have to wait for the first real release (1.0), and even then you might not want to change from minix. This is a source release for those that are interested in seeing what linux looks like, and it's not really supported yet. Anyone with questions or suggestions (even bug-reports if you decide to get it working on your system) is encouraged to mail me.

8. Getting it working

Most hardware dependancies will have to be compiled into the system, and there a number of defines in the file "include/linux/config.h" that you have to change to get a personalized kernel. Also you must uncomment the right "equ" in the file boot/boot.s, telling the bootup-routine what kind of device your A-floppy is. After that a simple "make" should make the file "Image", which you can copy to a floppy (cp Image /dev/PS0 is what I use with a 1.44Mb floppy). That's it.

```
Without any programs to run, though, the kernel cannot do anything. You
should find binaries for 'update' and 'bash' at the same place you found
this, which will have to be put into the '/bin' directory on the
specified root-device (specified in config.h). Bash must be found under
the name '/bin/sh', as that's what the kernel currently executes. Happy
hacking.


        Linus Torvalds       "torvalds@kruuna.helsinki.fi"
        Petersgatan 2 A 2
        00140 Helsingfors 14
        FINLAND
```

2. RELNOTES-0.12

```
        RELEASE NOTES FOR LINUX v0.12


This is file mostly contains info on changed features of Linux, and
using old versions as a help-reference might be a good idea.


        COPYRIGHT

The Linux copyright will change: I've had a couple of requests to make
it compatible with the GNU copyleft, removing the "you may not
distribute it for money" condition.  I agree.  I propose that the
copyright be changed so that it confirms to GNU - pending approval of
the persons who have helped write code.  I assume this is going to be no
problem for anybody: If you have grievances ("I wrote that code assuming
the copyright would stay the same") mail me.  Otherwise The GNU copyleft
takes effect as of the first of February.  If you do not know the gist
of the GNU copyright - read it.


        INSTALLATION

This is a SHORT install-note. The installation is very similar to 0.11,
so read that (INSTALL-0.11) too. There are a couple of programs you will
need to install linux: something that writes disk images (rawrite.exe or
NU or...) and something that can create harddisk partitions (fdisk under
xenix or older versions of dos, edpart.exe or something like that).

NOTE! Repartitioning your harddisk will destroy all data on it (well,
not exactly, but if you know enough to get back the data you probably
didn't need this warning).  So be careful.

READ THIS THROUGH, THEN READ INSTALL-0.11, AND IF YOU ARE SURE YOU KNOW
WHAT YOU ARE DOING, CONTINUE.  OTHERWISE, PANIC.  OR WRITE ME FOR
EXPLANATIONS.  OR DO ANYTHING BUT INSTALL LINUX - IT'S VERY SIMPLE, BUT
IF YOU DON'T KNOW WHAT YOU ARE DOING YOU'LL PROBABLY BE SORRY.  I'D
RATHER ANSWER A FEW UNNECESSARY MAILS THAN GET MAIL SAYING "YOU KILLED
MY HARDDISK, BASTARD.  I'M GOING TO FIND YOU, AND YOU'LL BE SORRY WHEN I
DO".

1) back up everything you have on your harddisk - linux-0.12 is still in
   beta and might do weird things.  The only thing I guarantee is that
   it has worked fine on /my/ machine - for all I know it might eat your
   harddisk and spit it out in small pieces on any other hardware.

2) Test out the linux boot-disk with the root file system.  If it
   doesn't work, check the hardware requirements, and mail me if you
   still think it should work.  I might not be able to help you, but
   your bug-report would still be appreciated.

   Test that linux can read your harddisk at least partly: run the fdisk
   program on the root-disk, and see if it barfs.  If it tells you about
   any partitions at all, linux can successfully read at least part of
   your harddisk.

3) Make sure that you have a free /primary/ partition.  There can be 4
   primary partitions per drive: newer DOS fdisks seem to be able to
   create only 2 (one primary and one extended).  In that case use some
   other partitioning software: edpart.exe etc.  Linux fdisk currently
   only tells you the partition info - it doesn't write to the disk.

   Remember to check how big your partition was, as that can be used to
   tell which device Linux thinks it is.

4) Boot up linux again, fdisk to make sure you now have the new
   partition, and use mkfs to make a filesystem on one of the partitions
   fdisk reports.  Write "mkfs -c /dev/hdX nnn" where X is the device
   number reported by linux fdisk, and nnn is the size - also reported
   by fdisk.  nnn is the size in /blocks/, ie kilobytes.  You should be
   able to use the size info to determine which partition is represented
   by which device name.

5) Mount the new disk partition: "mount /dev/hdX /user".  Copy over the
   root filesystem to the harddisk, eg like this:

    # for i in bin dev etc usr tmp
    # do
    # cp +recursive /$i /user
    # done

   You caanot use just "cp +recursive / /user", as that will result in a
   loop.

6) Sync the filesystem after you have played around enough, and reboot.

    # sync
    <wait for it to sync>
    ctrl-alt-del

   The folklore says you should do this three times before rebooting:
   once should be enough, but I admit I do it three times anyway :) THIS
   IS IMPORTANT! NEVER EVER FORGET TO SYNC BEFORE KILLING THE MACHINE.

7) Change the bootdisk to understand which partition it should use as a
   root filesystem.  See INSTALL-0.11: it's still the word at offset
   508 into the image. You should be up and running.


That's it. Go back and read the INSTALL-0.11


        New features of 0.12, in order of appearance
            (ie in the order you see them)

    Linux now prints cute dots when loading

WoW. Run, don't walk, to see this :). Seriously, it should hopefully now
load even on machines that never got off the ground before, but
otherwise the loading hasn't changed. Implemented by drew.

    Super-VGA detection for extended alphamun modes

I cannot guarantee it, I didn't write it, but it works great on a ET400
SVGA card.  I'm addicted to the new look with 100x40 character editing,
instead of a cramped 80x25.  This only works on VGA-cards that support
higher text-resolutions, and which are correctly identified. Implemented
by d88-man.

    Job Control.

Ok, everybody used to typing ^Z after they started a long command, and
forgot to put it in the background - now it works on linux too.  Bash
knows the usualy job-control commands: bg, fg, jobs & kill.  I hope
there will be no nasty surprises.  Job control was implemented by
tytso@athena.mit.edu.

    Virtual consoles on EGA/VGA screens.

You can select one of several consoles by pressing the left alt-key and
a function key at the same time. Linux should report the number of
virtual consoles available upon bootup. /dev/tty0 is now "the current"
screen, /dev/tty1 is the main console, and /dev/tty2-8 can exist
depending on your text-mode or card.

NOTE! Scrolling is noticeably much slower with virtual consoles on a
EGA/VGA. The reason is that no longer does linux use all the screen
memory as a long buffer, but crams in several consoles in it. I think
it's worth it.

The virtual consoles also have some new screen-handling commands: they
```

```
confirm even better to vt200 control codes than 0.11. Special graphic
characters etc: you can well use them as terminals to VMS (although
that's a shameful waste of resources).

    pty's

Ok. I have to admit that I didn't get the hangup-code working correctly,
but that should be easy to add. The general things are there.

    select

I've never used it, so I cannot say how well it works. My minor testing
seems to indicate that it works ok. vc's, pty's and select were
implemented by pmacdona, although I hacked it heavily.

    387-emulation.

It's not complete, but it works well enough to run those gcc2.0 compiled
programs I tested (few).  None of the "heavy" math-functions are
implemented yet.

    Symbolic links.

Try out a few "ln -s xx yy", and ls -l. Note that I think tar should be
recompiled to know anout them, and probably some other programs too. The
0.12 rootimage-disk has most of the recompiled fileutilities.

    Virtual memory.

In addition to the "mkfs" program, there is now a "mkswap" program on
the root disk.  The syntax is identical: "mkswap -c /dev/hdX nnn", and
again: this writes over the partition, so be careful.  Swapping can then
be enabled by changing the word at offset 506 in the bootimage to the
desired device.  Use the same program as for setting the root file
system (but change the 508 offset to 506 of course).

NOTE! This has been tested by Robert Blum, who has a 2M machine, and it
allows you to run gcc without much memory.  HOWEVER, I had to stop using
it, as my diskspace was eaten up by the beta-gcc-2.0, so I'd like to
hear that it still works: I've been totally unable to make a
swap-partition for even rudimentary testing since about christmastime.
Thus the new changes could possibly just have backfired on the VM, but I
doubt it.

    And that's it, I think.

Happy hacking.

            Linus
```

3. RELNOTES-0.95

```
        RELEASE NOTES FOR LINUX v0.95
        Linus Torvalds, March 7, 1992


This is file mostly contains info on changed features of Linux, and
using old versions as a help-reference might be a good idea.


        COPYRIGHT

Linux-0.95 is NOT public domain software, but is copyrighted by me.  The
copyright conditions are the same as those imposed by the GNU copyleft:
get a copy of the GNU copyleft at any major ftp-site (if it carries
linux, it probably carries a lot of GNU software anyway, and they all
contain the copyright).

The copyleft is pretty detailed, but it mostly just means that you may
freely copy linux for your own use, and redistribute all/parts of it, as
long as you make source available (not necessarily in the same
distribution, but you make it clear how people can get it for nothing
more than copying costs).  Any changes you make that you distribute will
also automatically fall under the GNU copyleft.

NOTE! The linux unistd library-functions (the low-level interface to
linux: system calls etc) are excempt from the copyright - you may use
them as you wish, and using those in your binary files won't mean that
your files are automatically under the GNU copyleft.  This concerns
/only/ the unistd-library and those (few) other library functions I have
written: most of the rest of the library has it's own copyrights (or is
public domain).  See the library sources for details of those.


        INSTALLATION

This is a SHORT install-note.  The installation is very similar to 0.11
and 0.12, so you should read INSTALL-0.11 too.  There are a couple of
programs you will need to install linux: something that writes disk
images (rawrite.exe or NU or...) and something that can create harddisk
partitions (fdisk under xenix or older versions of dos, edpart.exe or
something like that).

NOTE! Repartitioning your harddisk will destroy all data on it (well,
```

not exactly, but if you know enough to get back the data you probably didn't need this warning). So be careful.

READ THIS THROUGH, THEN READ INSTALL-0.11, AND IF YOU ARE SURE YOU KNOW WHAT YOU ARE DOING, CONTINUE. OTHERWISE, PANIC. OR WRITE ME FOR EXPLANATIONS. OR DO ANYTHING BUT INSTALL LINUX - IT'S VERY SIMPLE, BUT IF YOU DON'T KNOW WHAT YOU ARE DOING YOU'LL PROBABLY BE SORRY. I'D RATHER ANSWER A FEW UNNECESSARY MAILS THAN GET MAIL SAYING "YOU KILLED MY HARDDISK, BASTARD. I'M GOING TO FIND YOU, AND YOU'LL BE SORRY WHEN I DO".

Minumum files needed:

```
    RELNOTES-0.95 (this file)
    INSTALL-0.11 (+ any other docs you might find: the FAQ etc)
    bootimage-0.96.Z
    rootimage-0.95.Z
    rootimage-0.12.Z  (for tar+compress)
    rawrite.exe
    some disk partitioner
```

1) back up everything you have on your harddisk - linux-0.95 is still in beta and might do weird things. The only thing I guarantee is that it has worked fine on /my/ machine - for all I know it might eat your harddisk and spit it out in small pieces on any other hardware.

2) Test out the linux boot-disk with the root file system. If it doesn't work, check the hardware requirements, and mail me if you still think it should work. I might not be able to help you, but your bug-report would still be appreciated.

   Linux-0.95 now has an init/login: there should be 4 logins started on the first 4 virtual consoles. Log in as root (no password), and test it out. Change to the other logins by pressing left-alt + FN[1-4]. Note that booting up with a floppy as root is S..L..O..W.. - the floppy driver has been optimized for sequential access (backups etc), and trashes somewhat with demand-loading.

   Test that linux can read your harddisk at least partly: run the fdisk program on the root-disk, and see if it barfs. If it tells you about any partitions at all, linux can successfully read at least part of your harddisk.

   NOTE! Harddisk device names and numbers have changed between versions 0.12 and 0.95: the new numbering system was needed for the extended partitions, and a new naming scheme was in order so that people wouldn't cunfuse the old devices with the new ones.

   The new harddisk device names are: /dev/hd followed by an 'a' for the

```
   first drive, or a 'b' for the second one.  After that comes the
   partition number, 1-4 for the primary partitions, 5- for possible
   extended partitions.  No number means the complete disk. Like this:

    /dev/hda the whole first harddisk (old: /dev/hd0)
    /dev/hdb3    partition nr 3 on the second disk (old: /dev/hd8)

3) Make sure that you have a free /primary/ partition.  There can be 4
   primary partitions per drive: newer DOS fdisks seem to be able to
   create only 2 (one primary and one extended).  In that case use some
   other partitioning software: edpart.exe etc.  Linux fdisk currently
   only tells you the partition info - it doesn't write to the disk.

   Remember to check how big your partition was, as that can be used to
   tell which device Linux thinks it is.

   NOTE! Linux-0.95 /might/ recognize extended partitions: but the code
   for this is utterly untested, as I don't have any of those.  Do NOT
   use the extended partitions unless you can verify that they are
   indeed correctly set up - if my routines are wrong, writing to the
   extended partitions might just overwrite some other partition
   instead.  Not nice.

4) Boot up linux again, fdisk to make sure you now have the new
   partition, and use mkfs to make a filesystem on one of the partitions
   fdisk reports.  Write "mkfs -c /dev/hdX nnn" where X is the device
   number reported by linux fdisk, and nnn is the size - also reported
   by fdisk.  nnn is the size in /blocks/, ie kilobytes.  You should be
   able to use the size info to determine which partition is represented
   by which device name.

5) Mount the new disk partition: "mount /dev/hdX /mnt".  Copy over the
   root filesystem to the harddisk, eg like this:

    # for i in bin dev etc usr tmp
    # do
    # cp +recursive /$i /mnt
    # done

   You caanot use just "cp +recursive / /mnt", as that will result in a
   loop.

6) Sync the filesystem after you have played around enough, and reboot.

    # sync
    # lo

    (none) login: sync
    <wait for it to sync>
    ctrl-alt-del
```

   THIS IS IMPORTANT! NEVER EVER FORGET TO SYNC BEFORE KILLING THE MACHINE.

7) Change the bootdisk to understand which partition it should use as a root filesystem. See INSTALL-0.11: it's still the word at offset 508 into the image. You should be up and running.

8) When you've successfully started up with your harddisk as root, you can mount the older rootimage (rootimage-0.12) from a floppy, and copy over any files you find there that weren't on the newer root-image.

   Mounting a floppy is easy: make the directory /floppy, and write:

```
    # mount /dev/PS0 /floppy (if you have a 3.5" drive)
```

   or

```
    # mount /dev/at0 /floppy (for 5.25" floppies)
```

   After that the files can be copied to your harddisk, eg:

```
    # cp /floppy/usr/bin/compress /usr/bin
    # ln -s /usr/bin/compress /usr/bin/compress
    # cp /floppy/usr/bin/tar.Z /usr/bin
    # uncompress /usr/bin/tar.Z
```

That's it. Now go back and read the INSTALL-0.11, until you are sure you know what you are doing.

New features of 0.95, in order of appearance (ie in the order you see them)

Init/login

Yeah, thanks to poe (Peter Orbaeck (sp?)), linux now boots up like a real unix with a login-prompt. Login as root (no passwd), and change your /etc/passwd to your hearts delight (and add other logins in /etc/inittab etc).

Bash is even bigger

It's really a bummer to boot up from floppies: bash takes a long time to load. Bash is also now so big that I couldn't fit compress and tar onto the root-floppy: You'll probably want the old rootimage-0.12 just in order to get tar+compress onto your harddisk. If anybody has pointers

```
to a simple shell that is freely distributable, it might be a good idea
to use that for the root-diskette.

Especially with a small buffer-cache, things aren't fun. Don't worry:
linux runs much better on a harddisk.

    Virtual consoles on any (?) hardware.

You can select one of several consoles by pressing the left alt-key and
a function key at the same time. Linux should report the number of
virtual consoles available upon bootup. /dev/tty0 is now "the current"
screen, /dev/tty1 is the main console, and /dev/tty2-8 can exist
depending on your text-mode or card.

The virtual consoles also have some new screen-handling commands: they
confirm even better to vt200 control codes than 0.11. Special graphic
characters etc: you can well use them as terminals to VMS (although
that's a shameful waste of resources), and the PF1-4 keys work somewhat
in the application-key mode.

    Symbolic links.

0.95 now allows symlinks to point to other symlinks etc (the maximum
depth is a rather arbitrary 5 links). 0.12 didn't like more than one
level of indirection.

    Virtual memory.

VM under 0.95 should be better than under 0.12: no more lockups (as far
as I have seen), and you can now swap to the filesystem as well as to a
special partition. There are two programs to handle this: mkswap to set
up a swap-file/partition and swapon to start up swapping.

mkswap needs either a partition or a file that already exists to make a
swap-area. To make a swap-file, do this:

    # dd bs=1024 count=NN if=/dev/hda of=swapfile
    # mkswap swapfile NN

The first command just makes a file that is NN blocks long (initializing
it from /dev/hda, but that could be anything). The second command then
writes the necessary setup-info into the file. To start swapping, write

    # swapon swapfile

NOTE! 'dd' isn't on the rootdisk: you have to install some things onto
the harddisk before you can get up and running.

NOTE2! When linux runs totally out of virtual memory, things slow down
```

```
dramatically. It tries to keep on running as long as it can, but at
least it shouldn't lock up any more. ^C should work, although you might
have to wait a while for it..

    Faster floppies

Ok, you don't notice this much when booting up from a floppy: bash has
grown, so it takes longer to load, and the optimizations work mostly
with sequential accesses.  When you start un-taring floppies to get the
programs onto your harddisk, you'll notice that it's much faster now.
That should be about the only use for floppies under a unix: nobody in
their right mind uses floppies as filesystems.

    Better FS-independence

Hopefully you'll never even notice this, but the filesystem has been
partly rewritten to make it less minix-fs-specific. I haven't
implemented all the VFS-patches I got, so it's still not ready, but it's
getting there, slowly.

    And that's it, I think.

Happy hacking.

            Linus (torvalds@kruuna.helsinki.fi)
```

4. RELNOTES-0.95a

```
Please FIRST read the RELNOTES-0.95 file, then read this.  This is only
a listing of the differences between this release and the last. [-mkj]

CHANGES IN THE LINUX v0.95a ROOT DISKETTE
Jim Winstead Jr. - March 17, 1992

This file mostly contains info about the changes in the root diskette
from Linux v0.95/0.12 to Linux v0.95a.

CHANGES

With the release of Linux v0.95a, the maintenance of the root diskette
has been assumed by Jim Winstead Jr. (jwinstea@jarthur.Claremont.EDU).
This means there are a few large changes between the Linux 0.95 and
0.12 root floppies and the Linux 0.95a root floppy.  These are
detailed (as much as I remember them) below:

-       'bash' has been replaced with 'ash', the BSD 4.3 /bin/sh.  This
        freed up nearly 200k on the root floppy.  However, there are
        some problems with 'ash' that haven't been resolved:

        - sometimes the backspace key will not work on a virtual
          console.  I've found that it usually works on all _but_ one
          console, so this is only a minor hinderance.

        - 'ash 'supports BSD-style job control, and this has not yet been
          adapted to Linux's more POSIXish job control.  This means
          that 'ash' does not yet support job control, but it's being
          worked upon.

-       'tar' and 'compress' are back on the root floppy.  'tar' is
        compressed, and both utilities are in /bin.

-       'pfdisk', a disk partitioner, was added to the root floppy.
        This makes it (almost) possible to install Linux on a machine
        without looking at another OS.

-       the file pager 'more' has been added to the floppy.  This was
        added because of the addition of some documentation files on
        the root floppy.

-       'cat' has been added to /bin.
-       many utilities have been moved from /usr/bin to /bin, to
        conform to the Linux Directory Structure Standard (v1.0).
        These utilities are ones that are 'vital to the restoration of
        other file systems in the case of a corrupting crash.'

-       'init' and 'update' have been moved to /etc from /bin.  This
        was done because neither program should be executed from the
        command line by any user, including root.  (That means don't
        put /etc in your PATH!)  This has been a matter of some
        controversy, but this is how it will stand until the Linux
        Standards mailing list/committee decides otherwise.

-       tty64, tty65, etc, have been renamed to ttys1, ttys2, etc.

-       the directory /INSTALL was added, which contains some
        documentation, and three simple shell scripts to make
        installing Linux on a hard drive partition easier.  These are:

        - 'mktree', which makes a directory tree on the specified
          mounted device.
        - 'mkdev' which creates the standard devices in the dev
          directory of the specified mounted device
        - 'install' which installs the programs on the root diskette
          to the specified mounted device

        These programs will normally be called with '<name> /mnt'.

-       rootdev is different than the one on v0.95.  A couple of days
        after the release of 0.95, a program called 'rdev' was posted
        to alt.os.linux that duplicated and extended the functionality
        of rootdev.  This was renamed to rootdev and replaces the old
        rootdev.

-       agetty was renamed to getty, to be consistent with common Unix
        practice.

-       an improved fdisk was added that correctly reports extended
        partitions,  (Thanks to Linus!)

-       /dev is complete, or at least more complete than the last few
        releases of the root diskette, which always seemed to be a
        major complaint.  :)

-       /etc/issue and /etc/motd have been expanded to be a little
        more informative.  (Yeah, I know, big deal! :)

-       chgrp was removed.  You can use chown to get the same effect,
        but you just have to specify an owner, too.
```

```
Many of these changes were discussed on alt.os.linux, or the Linux
Standards group, so they may look familiar.

If you have questions, problems, or complaints about the root
diskette, either post to alt.os.linux, or send mail to me at
jwinstea@jarthur.Claremont.EDU.

If you have questions, problems, or complaints about the boot diskette
or the kernel itself, post to alt.os.linux or send mail to Linus
Torvalds at torvalds@cc.helsinki.fi.

Remember, the only stupid questions are the ones you don't ask.

FUTURE CHANGES

I'm already anticipating some changes for the next release, so here's
a sneak preview:

-       shared libraries.  These are currently in alpha testing, and
        will hopefully free up some more room on the root floppy for
        more goodies.

-       a generic mtools might be added to the root floppy.

-       a better fdisk to replace the current fdisk/pfdisk pair.  You
        won't need to know your drive's geometry for this, and it will
        know about Linux extended partitions.

-       an improved sh.  I'm working on the backspace problem, and
        adding job control.  I'm also going to look at using the GNU
        readline library for input, as long as it doesn't add
        substantially to the size of sh.

-       init/getty/login may be removed from the root floppy.  The
        main reason they'll still on there is the backspace problem
        with ash.

-       improved installation documentation.  People have started work
        on this already - read alt.os.linux for previews.

-       more robust installation scripts.  The current ones are quick
        and dirty, and work well, but I'd like to add better ones.

-       miscellaneous utilities added.  I'd really like to add an
        editor to the root disk, but I haven't found one small enough.
        Any suggestions?

-       various other things that I can't remember right now.
```

```
Again, mail your questions, comments and suggestions about the root
diskette to me at jwinstea@jarthur.Claremont.EDU.
--
Jim Winstead Jr. (CSci '95)    | "Catch a fish!"
Harvey Mudd College            |  -Geddy Lee,
jwinstea@jarthur.Claremont.EDU |   San Diego Sports Arena
Disclaimer: Mine, not theirs!  |   January 20, 1992
```

5. RELNOTES-0.95c+

```
This is release 0.95c+ of the linux kernel - it contains some
enhancements and bugfixes to the 0.95a kernel, as well as some minor
fixes relative to the last alpha-patch (0.95c).  The release is
available as

- binary     (bootimage-0.95c+.Z)
- full source        (linux-0.95c+.tar.Z)
- patches rel. to 0.95c  (diff-0.95c.c+.Z)
- patches rel. to 0.95a  (diff-0.95a.c+.Z)

NOTE TO PATCHERS!! Before patching, do this:
 - make an empty include-file linux/include/checkpoint.h
 - rename linux/kernel/math/math_emulate.c as just emulate.c
That is, from the linux source directory do:

    $ > include/checkpoint.h
    $ mv kernel/math/math_emulate.c kernel/math/emulate.c

Also note that patching from the 0.95a version is probably not worth it
as it's easier to get the complete new sources.

Although I'm making binaries and the full source available, this isn't
really a major release: there is no new rootdisk, and this is more "my
current kernel" and not really tested (I put in the last changes 5
minutes before packing all this up).

The reason I'm making this available is that with the advent of gcc-2.1
and the VFS-library the old kernel doesn't really do everything the new
libraries want: the readdir system call is needed to get things working.
The default compiler after this release is considered to be gcc-2.0 or
higher (although 1.40 still works - you don't /have/ to change).  People
who are unable or unwilling to patch a new kernel shouldn't be unable to
run the new binaries.

This kernel should be totally backwards compatible, so no binaries
should break.  I resisted adding the changed mount() system call into
this release: the next major release will have a third parameter for
mount() - the filesystem type name (ie mount /dev/xxx /mnt minix).

Fixes relative to 0.95c:

- corrected two minor bugs in readdir() (thanks to R Card)
- lp-patches are in.  I've edited them a bit, and will probably do some
  more editing in the future, but they seem to work fine.

- 8-bit ISO latin output to the console (ie part of Johan Myreens
  general latin-1 patches: the keyboard patches aren't there)

- other minor bug-fixes (thanks to HH Bergman for noticing the
  timer-table bug)

Things I haven't had time to look into yet:

- select still has some problems
- reports that VC-output sometimes isdiscarded (never seen it myself)
- probably other things I've simply forgot...

            Linus
```

6. RELNOTES-0.97

```
Changes in 0.97:

 - The VESA-support was removed.  I'd be happy to put it back once it
   works on all hardware.  Instead of the VESA-code, I finally put in
   the automatic SVGA setup patches.  See the top-level Makefile.

 - The IRQ code has solidified, and should work on all machines.  Not
   all of the SCSI drivers use it yet, so I expect patches for that..

 - Serial interrupts are handled slightly differently, and performance
   should be up.  I've sent out a few alpha-releases, and testing seems
   to indicate that's actually true this time.  Reactions have ranged
   from "nice" to "wonderful" :-)

 - The buffer-cache and memory management code has been edited quite a
   bit.  ps/free etc programs that reads kernel memory directly no
   longer work, and even a recompilation won't be enough.  They actually
   need editing before they work.

   The buffer-cache now grows and shrinks dynamically depending on how
   much free memory there is.  Shift+PrintScreen will give some memory
   statistics.  (Ctrl+PrSc gives task-info, ALT+PrSc gives current
   register values).

   The mm code changes removed some race-conditions in the VM code, and
   I also tried to make the Out-of-swapspace error less severe (better
   thrashing-detection etc).

 - The super-block code has been cleaned up.  Especially the extended fs
   needs to be edited a bit to take advantage of the new setup, and I
   expect Remy Card will have a patch out eventually.

 - include-files have been moved around some more: there are still some
   names that clash with the standard headers, but not many.

 - Unswappable processes implemented: by default only 'init' is
   unswappable.  This is a bit safer in low-memory conditions, as at
   least init won't die due to low memory.  I also made killing init
   impossible: if init doesn't recognize a signal, it simply won't get
   it.  Some other changes ("while (1) fork();" won't kill the machine
   for non-root users etc)

 - The new SCSI drivers are in.  These make the kernel noticeably
```

```
   bigger, but you can leave them out if you don't want them.

 - The floppy- and hd-drivers print out more debugging-info in case of
   errors: this might be irritating if you have hardware that works, but
   often gives soft-errors.  On the other hand, some old debugging-info
   was removed - notably for user-level protection errors etc.

 - Various minor fixes.  I haven't made cdiffs (and I haven't gotten any
   requests for them, so I probably never will), but they would be
   pretty big.

Things that I didn't have time for:

 - I wanted to rewrite the tty drivers to be more "streams-like" (ie not
   an actual streams-implementation, but some of the ideas from
   streams).  I never got around to it: there was simply too much else
   to do.

 - I got a lot of patches, and some went in, others didn't.  If you
   think your patch was important, please re-send it relative to the new
   version.

I'd like comments on the new system: performance / clarity of code etc.
0.97 should correct all known bugs (at least the ones I know about), but
I guess that's just wishful thinking.

Note that the dynamic buffer-code also handles differently-sized
buffers, but that the rest of the system (block device drivers,
filesystem code etc) cannot yet take advantage of this - there is still
some coding needed.

        Linus
```

7. INSTALLATION.old

```
        Installing Linux on your system

Ok, this is a short guide for those people who actually want to get a
running system, not just look at the pretty source code :-). You'll
certainly need minix for most of the steps.


    0.  Back up any important software.  This kernel has been
working beautifully on my machine for some time, and has never destroyed
anything on my hard-disk, but you never can be too careful when it comes
to using the disk directly.  I'd hate to get flames like "you destroyed
my entire collection of Sam Fox nude gifs (all 103 of them), I'll hate
you forever", just because I may have done something wrong.

Double-check your hardware.  If you are using other than EGA/VGA, you'll
have to make the appropriate changes to 'linux/kernel/console.c', which
may not be easy.  If you are able to use the at_wini.c under minix,
linux will probably also like your drive.  If you feel comfortable with
scan-codes, you might want to hack 'linux/kernel/keyboard.s' making it
more practical for your [US|German|...] keyboard.


    1.  Decide on what root device you'll be using.  You can use any
(standard) partition on any of your harddisks, the numbering is the same
as for minix (ie 0x306, which I'm using, means partition 1 on hd2).  It
is certainly possible to use the same device as for minix, but I
wouldn't recommend it.  You'd have to change pathnames (or make a chroot
in init) to get minix and linux to live together peacefully.

I'd recommend making a new filesystem, and filling it with the necessary
files: You need at least the following:

    - /dev/tty0       (same as under minix, ie mknod ...)
    - /dev/tty        (same as under minix)
    - /bin/sh         (link to bash)
    - /bin/update         (I guess this should be /etc/update ...)

Note that linux and minix binaries aren't compatible, although they use
the same (gcc-)header (for ease of cross-compiling), so running one
under the other will result in errors.
```

```
    2.  Compile the source, making necessary changes into the
makefiles and linux/include/linux/config.h and linux/boot/boot.s.  I'm
using a slightly hacked gcc-1.40, to which I have added a -mstring-insns
flag, which uses the i386 string instructions for structure copy etc.
Removing the flag from all makefiles should do the trick for you.

NOTE! I'm using -Wall, and I'm not seeing many warnings (2 I think, one
about _exit returning although it's volatile - it's ok.) If you get
more warnings when compiling, something's wrong.


    3.  Copy the resultant code to a diskette of the right type.
Use 'cp Image /dev/PS0' or equivalent.


    4.  Boot with the new diskette.  If you've done everything right
(and if *I've* done everything right), you should now be running bash as
root.  You can't do much (alias ls='echo *' is a good idea :-), but if
you do run, most other things should work.  I'd be happy to hear from
anybody that has come this far - and I'll send any ported binaries you
might want (and I have).  I'll also put them out for ftp if there is
enough interest.  With gcc, make and uemacs, I've been able to stop
crosscompiling and actually compile natively under linux.  (I also have
a term-emu, sz/rz, sed, etc ...)

The boot-sequence should start with "Loading system...", and then a
"Partition table ok" followed by some root-dev info. If you forget to
make the /dev/tty0-character device, you'll never see anything but the
"loading" message. Hopefully errors will be told to the console, but if
there are problems at boot-up there is a distinct possibility that the
machine just hangs.


    5.  Check the new filesystem regularly with (minix) fsck.  I
haven't got any errors for some time now, but I cannot guarantee that
this means it will never happen.  Due to slight differences in 'unlink',
fsck will report "mode inode XXX not cleared", but that isn't an error,
and you can safely ignore it (if you don't like it, do a fsck -a every
once in a while).  Minix "restore" will not work on a file deleted with
linux - so be extra careful if you have a tendency to delete files you
don't really want to.

Logging out from the "login-shell" will automatically do a sync, and
will leave you hanging without any processes (except update, which isn't
much fun), so do the "three-finger-salute" to restart dos/minix/linux or
whatever.


    6.  Mail me and ask about problems/updates etc.  Even more
```

```
welcome are success-reports (yeah, sure), and bugreports or even patches
(or pointers to corrections).


NOTE!!! I haven't included diffs with the binaries I've posted for the
simple reason that there aren't any - I've had this silly idea that I'd
rather change the OS than do a lot of porting.  All source to the
binaries can be found on nic.funet.fi under /pub/gnu or /pub/unix.
Changes have been to makefiles or configuration files, and anybody
interested in them might want to contact me. Mostly it's been a matter
of adding a -DUSG to makefiles.

The one exception if gcc - I've made some hacks on it (string-insns),
and have got it (with the gracious help of Bruce Evans) to correctly
emit software floating point. I haven't got diffs to that one either, as
my hard-disk is overflowing and I cannot accomodate both originals and
changes, but as per the GNU copyleft I'll make them available if
someone wants them. I hope nobody want's them :-)


        Linus       torvalds@kruuna.helsinki.fi
```

8. INSTALL-0.10

Warning: I have personally not done this the hard way, so I don't know what problems could surface. In general, this version is still meant for people with minix: they are more used to the system, and can do some things that DOS-based persons cannot. If you have only DOS, expect some troubles. As the version number suggests, this is still not the final product.

This is a "fast hack", meant as a minimal guide to what you must do. I'll expand this as soon as people tell me what they have problems with etc etc. If somebody who has successfully installed the system wants to write something better, I'd be delighted. This guide stinks to high heaven.

        Installing Linux-0.10 on your system

There are 5 major steps in installing linux on your system:

1 - BACK UP ANY IMPORTANT DATA. Linux accesses your hardware directly, and if your hardware differs from mine, you could be in for a nasty surprise. Doublecheck that your hardware is compatible: AT style harddisk, VGA controller. (If somebody has EGA, please tell me if the screen driver should happen to work)

2 - Make a file-system on your harddisk. This is easy if you have minix, but if you haven't got minix, you'll have to get the minix demo-disk from somewhere (plains.nodak.edu is one place), and use that. There should be a manual accompanying the demo-disk, and you had better read that carefully. Although this version of linux will boot up without minix, a knowledge of minix would help. Especially if you have never done any unix work, you'll be very confused.

Making a filesystem means getting a empty partition (with DOS fdisk or similar), and using the 'mkfs /dev/hdX nnn' command to write out a empty file-system.

3 - copy the diskimages to two floppies. Again, under minix (or any unix), this is easy, as you can just do a simple 'dd' to a floppy, but from within MS-DOS this might be a bit trickier. 'debug' should be able to write diskettes directly, or you could get the sources to "raw-write"

from the same place as you got the minix demo disk, and modify them to write out any disk image (or do they do that already?).

NOTE! The floppies MUST be of the same type: even though the boot-image will fit nicely on a 360kB floppy, you have to write it to the same type of floppy as the root-image. That means a 1.2M or 1.44M floppy. The reason is that the floppy-type is determined at boot-time from the boot-floppy. Thus the same binary works on both 3.5" and 5.25" drives.

4 - boot up from floppy. This should be obvious. Having a floppy as root-device isn't very fast (especially on a machine with less than 6MB total ram -> small buffer cache), but it works (I hope). Test the programs on the root-floppy (cat mkdir etc).

5 - Mount the harddisk partition (I do it on /user: ie 'mount /dev/hdX /user'), and copy the file system over to the new partition. The following is a example of how to do this:

```
$ cd /user
$ mkdir usr
$ for i in bin etc usr/bin usr/root mtools
> do
> mkdir $i
> cp `ls -A /$i` $i
> done
$ mkdir dev
$ cd dev
$ for i in 0 1 2 3 4 5 6 7 8 9
> do
> mknod 'hd'$i b 3 $i
> done
$ mknod tty c 5 0
$ mknod tty0 c 4 0
$ mknod tty1 c 4 1
$ mknod tty2 c 4 2
```

You should now have a filesystem you could boot from. Play around a bit, try to get aquainted with the new system. Log out when you've had enough.

6 - Changing the boot-diskette use your new harddisk partition as root. The root device to be used for linux is encoded in a word at offset 508 in the boot image. Normally this is 0, meaning that the root is to be the same type of floppy as was used in the boot process. This can be changed to whatever you like.

```
Use a short program like the one at the end to change the word (I assume
everybody has access to some kind of C compiler, be it under dos or
unix).  You can then write out the new bootdisk, and boot from it, now
using the harddisk as root (much faster).  Once you have successfully
done that you might want to install additional programs (gcc etc) by
reading them from a dos-floppy with 'mcopy'.


        Linus (torvalds@kruuna.helsinki.fi)


------  example program: use 'a.out < oldboot > newboot' ----
#include <unistd.h>
char tmp[512];

void main(void)
{
    int i;

    if (512 != read(0,tmp,512))
        exit(1);
    if (0xAA55 != *((unsigned short *)(tmp+510)))
        exit(2);
    *((unsigned short *)(tmp+508)) = NEW_DEV;
    if (512 != write(1,tmp,512))
        exit(3);
    while ((i=read(0,tmp,512)) > 0)
        if (i != write(1,tmp,i))
            exit(4);
    exit(0);
}
-------

        Devices:

Harddisks:
0x301 - /dev/hd1 - first partition on first drive
...
0x304 - /dev/hd2 - fourth partition on first drive

0x306 - /dev/hd1 - first partition on second drive
...
0x309 - /dev/hd2 - fourth partition on second drive

0x300 - /dev/hd0 - the whole first drive. BE CAREFUL
0x305 - /dev/hd5 - the whole second drive. BE CAREFUL


Floppies:
```

```
0x208 - 1.2M in A
0x209 - 1.2M in B
0x21C - 1.44M in A
0x21D - 1.44M in B
```

9. INSTALL-0.11

```
        Using Linux v0.11
        Linus Torvalds 08.12.91

NOTE: Users of 0.10, please check the "changed" list before using 0.11.

        Booting linux

Linux-0.11 can easily be booted by getting the 2 files bootimage-0.11.Z
and rootimage-0.11.Z from the linux archive, uncompressing them and
writing them out to disks of the same size (ie 2 1.44M floppies or 2
1.2M floppies). Writing the disks is done with the "rawrite.exe" program
from dos, or with "dd" from unix. Linux is then booted simply by
inserting the bootdiskette in drive A, and rebooting the machine. If
everything goes well, linux will ask you to insert the root-disk after
loading the system. Hopefully linux will then correctly load the shell
executable, and leave you as root on the new system (prompt '# ').

        Using it.

You can get a complete list of available commands by pressing <tab>
twice: the root-disk contains mostly setup-programs needed to install
the system on a harddisk.  You can test them a bit, reading directories
etc.

In order to install linux on the harddisk, first check out your harddisk
by executing the command "fdisk" - it should show you all the partitions
available.  If you have only 1 AT-harddisk, you should get a
errormessage, just ignore it.  At my system fdisk reports the following:

/dev/hd1:  20476 blocks minix
/dev/hd2:  19975 blocks minix
/dev/hd3:   1020 blocks minix
/dev/hd4:    170 blocks active 16-bit DOS (>=32M)
/dev/hd6:  41641 blocks active minix

The partition type given (12-bit DOS, minix etc) doesn{t really mean
anything, unless it's a "extended partition", in which case you
shouldn't use that partition for anything: linux doesn't yet understand
them. When later using "mkfs" to make a linux file system, it won't
change the output of fdisk, so fdisk may well report "DOS", while in
fact you have made it a linux partition.
```

```
If fdisk doesn't print out anything but errors, linux is unable to read
your harddisk, and you are f**ked.  Play around with the floppy version,
but you won't be able to do anything real.

        Making a filesystem

In order to really use linux, you will have to make a filesystem on your
harddisk. This starts by deciding which partition you can use. Look
again at what fdisk reports, and try to figure out which of the
partitions you are using for DOS, OS/2 etc. /dev/hdX where X={1,2,3,4}
always refers to the first harddisk, X={6,7,8,9} always refers to the
second disk. /dev/hd0 and /dev/hd5 are special: they are all of the
drive, and mkfs will refuse to use them for a filesystem.

When you are certain you know which device points to which partition,
you make a filesystem on the partition of your choice by writing:

    mkfs -c /dev/hdX blocks

where "-c" means that you want mkfs to check for errors, "dev/hdX" is
the free partition you intend to use for linux, and "blocks" is the
number of blocks fdisk reports for that particular partition. NOTE! mkfs
will overwrite the partition you selected, so be doubly (or triply) sure
that you don't mind that.

Note that when using the "-c" flag, mkfs will read through the entire
partition: this can take some time. If there are read errors, mkfs will
mark the particular block as bad, and continue: linux will also print a
little message "harddisk I/O error". After running mkfs these messages
should never occur again: if they do, your data may be corrupted.

        Mounting the filesystem

After mkfs has exited, it's time to mount the file-system, and do the
necessary things to make it a root file system. Mount the new filesystem
on /user by writing:

    cd /
    mount /dev/hdX /user

If you get errors for this, mkfs failed, and there is probably something
seriously wrong.

After mounting the device, you want to move all the files on the current
floppy-root to the new fs. This can most easily be done by writing:

    cd /user
    for i in bin dev etc usr tmp floppy
    do
    cp +recursive +verbose /$i $i
    done
    sync
```

which will also tell you what it is doing (/bin/sh -> bin/sh etc).

After that, you should have a new filesystem that contains the bare necessities to start hacking linux. Play around some more, and exit linux by writing "logout or exit". This should result in

```
    child 4 died with error code 0000
    #
```

Do a couple of syncs (3 is a magic number), and reboot the machine. ALWAYS remember to sync before rebooting: terrible things happen if you don't.

Using the harddisk as root

Once you have happily made a new root, you will want to boot up with it. This is done by changing a word at offset 508 in the boot-image. The word (in 386-order, ie low byte first) tells the system which device to use as root: it is initially 0, which means that we want to use a floppy of the same type as the boot-disk (and this is the reason that you may not use a 360kB boot-disk even though the system fits on one: it has to be the same type as the root-diskette).

In order to use the harddisk as root, this value has to be changed to point to the correct device. Harddisks have a major number of 3 under linux, and the minor nr is the same as the number X in /dev/hdX. The complete device number is then calculated with

```
    DEV_NO = (major<<8)+minor
```

or alternatively major*256+minor. Thus /dev/hd1 is (3<<8)+1 = 0x301, /dev/hd6 = 0x0306 etc. Assuming the partition you made into the new root was /dev/hd2, you will have to write 0x0302 into the boot-image. That is, you should change the 508th byte in the image to 0x02, and the 509th byte to 0x03. There is a sample program for this in some of the older INSTALL-notes, if you don't understand what it's all about.

Ok, I got the root on hd, what now?

As you have probably noticed, you cannot get very far with the binaries found on the original root-diskette. So the first thing you want to do is to import some new binaries. To do this you need to tell linux what kind of floppies you have, as that's the easiest way to import things.

As with harddisk, floppies have device numbers, but this time major = 2 instead of 3. The minor number is not as easy: it's a composite that tells which drive (A, B, C or D) and what type of drive (360kB, 1.2M, 1.44M etc). The formula is 'minor = type*4+nr', where nr is 0-3 for A-D, and type is 2 for 1.2M disks, and 7 for 1.44M disks. There are other types, but these should suffice for now.

Thus if you have a 1.2M A-drive, and want to call it "floppy0", you have to tell linux so. This is done with the "mknod" command. mknod takes 4 paramters: the unix name of the device, a "b" or a "c" depending on whether it's a Block of Character device, and the major and minor numbers. Thus to make "floppy0" a 1.2M A-drive, you write:

```
    mknod /dev/floppy0 b 2 8
```

b is for Block-device, the 2 is for floppy, and the 8 is 4*2+0, where the 2 is 1.2M-drive and the 0 is drive A. Likewise to make a "floppy1" device that is a 1.44M drive in B, you write:

```
    mknod /dev/floppy1 b 2 29
```

where 29 = 4*7 + 1. There are a couple of standard names, for users that are used to minix (major, minor in parentheses): /dev/PS0 is a 1.44M in A (2,28), /dev/PS1 a 1.44M in B (2,29), /dev/at0 is a 1.2M in A (2,8), /dev/at1 is a 1.2M in B (2,9). Use mknod to make those that fit your computer.

After you have made these special block devices, you can now read a floppy under linux. The easiest way to import things into linux is by writing a tar-file to a floppy with rawrite.exe, and then using:

```
    tar xvf /dev/floppy0
```

to untar it under linux. This way you can get the gcc binaries etc available from the linux-carrying sites.

        Changes from 0.10:

- /bin/update is no longer automatically executed upon bootup: instead the file /etc/rc is evaluated by the shell. This file can then start the update process, mount andy needed filesystems, possibly fsck'ing them first. A minimal /etc/rc looks like this:

```
    /bin/update &
    > /etc/mtab
    echo " Ok."
```

- init() restarts the shell every time it is exited: logout from the login shell results in a "child xxx died with error code yyy", a sync and then a new shell as root.

```
- floppies work a lot better than in 0.10. Even using two floppies at
the same time seems to work out ok. Reading big chunks at a time is also
faster then in 0.10 (I think).

- harddisk errors are handled better. Use the "-c" option in mkfs to map
out all errors.

- linux accepts most video-cards: harcules, MDA, CGA etc seem to work.

- ^G beeps on the console, so command completion under bash etc will
notify of errors.

- sticky directories, corrected handling of uid/gid bits, and better
handling of protections when not root. Most of these won't be noticeable
until we get a init/login.
```

10. INSTALL-0.95a

```
INSTALL NOTES FOR LINUX v0.95a
Jim Winstead Jr. - March 17, 1992

This file contains basic instructions for installing Linux v0.95a.
More detailed instructions are being written by others.  Read
alt.os.linux for details on this, and to see preliminary drafts.

COPYRIGHT

Linux 0.95a is NOT public domain software, but is copyrighted by Linus
Torvalds (torvalds@cc.helsinki.fi).  The copyright terms follow the
GNU Copyleft.  See the file COPYING from any GNU software package for
the finer points.  Note that the unistd library functions and all
library functions written by Linus Torvalds are exempt from this
copyright, and you may use them as you wish.

INSTALLATION

1)  First, and absolutely the most important step, MAKE BACKUPS OF YOUR
    SYSTEM!  This system won't do anything nearly as nasty as coredump all
    over your harddrive (see 386BSD v0.0), but it is quite easy to
    accidently screw something up while installing.

2)  Test out the Linux v0.95a boot disk with the Linux v0.95a root
    disk.  If you are unable to get the boot disk to work properly on
    your system, try posting to alt.os.linux, or contacting Linus.

    Notice that Linux (as of v0.95) contains an init/getty/login suite,
    and this will start up 'login' on the first four virtual consoles,
    accessed by Left-Alt-F[1234].  If you experience problems on one
    virtual console, it should be possible to switch to another one.

    (There is a good chance the backspace key will not work with
    /bin/sh on your first virtual console, as this how it often behaves
    on my machine.  I've noticed that it usually works in the other
    virtual consoles, however.)

3)  Run the 'fdisk' program on the root floppy.  This will tell you how
    each of your harddrives is partitioned.  Note that the names of the
    hard drive partitions has changed from v0.12, and 'fdisk' now
    properly reports the new device names (unlike the fdisk with v0.95).

    If 'fdisk' tells you about any partitions at all, Linux can
    successfully read at least part of your harddisk, and you will most
    likely be able to install Linux on your harddrive.

    If you have used previous versions of Linux, you will notice that
    'fdisk' now recognizes extended partitions.  Support for this in
    the kernel, however, is largely untested.  If you're feeling brave,
    go ahead and try, and report any problems to Linus.

4)  Make sure you have a free (preferably primary) partition on your
    hard drive.  If you want to repartition your harddrive, you can use
    the pfdisk program on the root floppy.  See pfdisk.man in the
    /INSTALL directory for more details on using this program.  (NOTE:
    you will need to know your hard drives disk geometry to use pfdisk.
    You can find this out by examining your CMOS setup on most computers.)

5)  If you have used pfdisk to change your partition table, be sure to
    reboot Linux now, so the new partition table will be recognized by
    Linux.

6)  Use 'fdisk' again to check the partitions on your hard drive, and
    use 'mkfs' to make a Linux (minix) filesystem on the partition you
    want to be using for Linux.  The proper command is "mkfs
    /dev/hdX nnn" where X is the partition (i.e. a1, a2, b3, etc.) and
    nnn is the size in blocks (kilobytes) of the partition as reported
    by fdisk.  You should be able to use the size of the partitions to
    tell them apart.

7)  Mount the new filesystem.  This can be done by using "mount
    /dev/hdX /mnt", which will mount the partition into the directory
    /mnt.

8)  Run the script in /INSTALL called 'mktree'.  This will create a
    bare directory tree built down from the specified directory.  So,
    for a standard installation, you would use "mktree /mnt", which
    would build the bare directory tree starting from /mnt.

9)  Run the script in /INSTALL called 'mkdev'.  This will create the
    standard Linux devices in the directory 'dev' in the specified
    directory.  For a standard installtion, this would mean typing
    'mkdev /mnt' to create the devices in /mnt/dev.

    NOTE: This step is really optional, since the 'install' script
          (next step) will do this if it sees you haven't.

10) Run the script in /INSTALL called 'install'.  This will copy over
    the binary programs from the root disk to the directory tree on
    the specified directory.  This means typing 'install /mnt' for a
    standard installation.
    NOTE: (for those upgrading from previous versions of Linux)

    The 'install' script uses the +interactive switch for copying
    files from /etc, which means you can tell it whether or not to
    overwrite any of these files.  'install' will also go through
    your /usr/bin and /bin directories and ask you if it should
    remove any incorrectly placed files.  (Such as /bin/update and
    /bin/init, which have both been moved to /etc.)

11) You should now have a complete (but very basic) root filesystem on
    your harddrive.  To be able to boot from floppy with this as your
    root filesystem, you will have to edit the boot diskette.  This is
    done by modifying the word at offset 508 (decimal) with a program
    such as Norton's Disk Editor, or use pboot.exe (available where
    you got this file, the boot disk and the root disk, hopefully.)

    This word is in 386-order (that is, least-significant byte first),
    which means it should look like one of the following:

       LSB MSB   -   device
       --------------------------
    01 03    -   /dev/hda1    LSB = Least-Significant Byte
    02 03    -   /dev/hda2    MSB = Most-Significant Byte
    03 03    -   /dev/hda3
    04 03    -   /dev/hda4

    41 03    -   /dev/hdb1
    42 03    -   /dev/hdb2
    43 03    -   /dev/hdb3
    44 03    -   /dev/hdb4

    The numbers are in hex, and if you're editing the boot diskette by
    hand, these two bytes should initially be 00 00 (and are followed
    by two non-zero bytes).

    Note that pboot.exe predates Linux 0.95a, so some of the
    information it presents is inaccurate (it refers to the old hd*
    naming scheme).  The codes to use are as above, but with the most-
    significant byte first.  (So /dev/hda1 = 0301, /dev/hda2 = 0302,
    etc.)

12) You should now be able to boot from this diskette and it will use
    your new Linux partition as the root partition.  You'll notice,
    however, that you can't do a whole lot with just the programs on
    the root diskette.  You'll need to get further packages from
    whereever you got the root and boot diskettes, and read these from
    a floppy using tar and compress.  (Simple instructions:  Download
    the file to DOS, use rawrite to write the tar file to diskette.
    Use 'tar zxvf /dev/<floppy>' to read the file from floppy, where
    <floppy> is the appropriate floppy device.  (PS0 is a 1.44 meg
    3.5" as A:, PS1 is a 1.44 meg as B:, at0 is a 1.2 meg as A:, at1
    is a 1.2 meg as B:.)

13) Before you ever reboot your machine when it's running Linux, you
    should run 'sync'.  This flushes Linux's disk buffers, making sure
    everything has been written to disk.  Failing to do this could
    result in badly corrupted filesystems.

----------------------------------------------------------------------------

These instructions are not the best, but should be enough to get you
going.  If you have more questions, either post on alt.os.linux, or
send mail to me (jwinstea@jarthur.Claremont.EDU), or to Linus
(torvalds@cc.helsinki.fi).  Remember, the only stupid questions are
the ones that you don't ask.
```

11. INSTALL-0.96

```
INSTALL NOTES FOR LINUX v0.96
Jim Winstead Jr. - July 4, 1992

This file contains basic instructions for installing Linux v0.96.
More detailed instructions have been written by others.  Read the
Linux FAQ for some suggestions, and for pointers to other installation
documents.

COPYRIGHT

Linux 0.96 is NOT public domain software, but is copyrighted by Linus
Torvalds (torvalds@cc.helsinki.fi).  The copyright terms follow the
GNU Copyleft.  See the file COPYING from any GNU software package for
the finer points.  Note that the unistd library functions and all
library functions written by Linus Torvalds are exempt from this
copyright, and you may use them as you wish.

WARNING

    The 0.96 root disk requires the 0.96b or later kernel.  A bootable
    image of this kernel should be available where you got the image
    for the 0.96 root disk.

INSTALLATION

1)  First, and absolutely the most important step, MAKE BACKUPS OF YOUR
    SYSTEM!  This system won't do anything nearly as nasty as coredump all
    over your harddrive (see 386BSD v0.0), but it is quite easy to
    accidently screw something up while installing.

2)  Test out the Linux v0.96b boot disk with the Linux v0.96 root
    disk.  If you are unable to get the boot disk to work properly on
    your system, try posting to comp.os.linux, or contacting Linus.

    Notice that Linux (as of v0.95) contains an init/getty/login suite,
    and this will start up 'login' on the first four virtual consoles,
    accessed by Left-Alt-F[1234].  If you experience problems on one
    virtual console, it should be possible to switch to another one.

3)  login as 'install', and the system will walk you through the
    process of installing Linux on a hard drive partition.  The
    process is fairly automated, but the process requires that you go
    through the steps of creating a partition for Linux usage.  Some
    tips follow:

    Read the efdisk file from the intro login, which will explain
    the basic concepts of hard disk partitions, and how to use
    efdisk.

    You may find it useful to login to one virtual console as
    intro, so you can access the on-disk documentation, and
    another as install, so you can do the installation and easy
    access the documentation.

    The maximum size of a Minix filesystem (the type created by
    mkfs) is 64 megabytes.  This is not a limitation of mkfs or
    Linux, but a limitation of the Minix filesystem that is used.
    With the release of Linux v0.97, a new 'extended' filesystem
    will be released that will support 4 terabyte (!) partitions,
    and extended filenames.

4)  You should now have a complete (but very basic) root filesystem on
    your harddrive.  To be able to boot from floppy with this as your
    root filesystem, you will have to edit the boot diskette.  This is
    done by modifying the word at offset 508 (decimal) with a program
    such as Norton's Disk Editor, or use pboot.exe (available where
    you got this file, the boot disk and the root disk, hopefully.)

    This word is in 386-order (that is, least-significant byte first),
    which means it should look like one of the following:

       LSB MSB   -   device
       --------------------------
    01 03    -   /dev/hda1    LSB = Least-Significant Byte
    02 03    -   /dev/hda2    MSB = Most-Significant Byte
    03 03    -   /dev/hda3
    04 03    -   /dev/hda4

    41 03    -   /dev/hdb1
    42 03    -   /dev/hdb2
    43 03    -   /dev/hdb3
    44 03    -   /dev/hdb4

    The numbers are in hex, and if you're editing the boot diskette by
    hand, these two bytes should initially be 00 00 (and are followed
    by two non-zero bytes).

    Note that pboot.exe predates Linux 0.95a, so some of the
    information it presents is inaccurate (it refers to the old hd*
    naming scheme).  The codes to use are as above, but with the most-
    significant byte first.  (So /dev/hda1 = 0301, /dev/hda2 = 0302,
    etc.)
```

```
5)  You should now be able to boot from this diskette and it will use
    your new Linux partition as the root partition.  You'll notice,
    however, that you can't do a whole lot with just the programs on
    the root diskette.  You'll need to get further packages from
    whereever you got the root and boot diskettes, and read these from
    a floppy using tar and compress.  (Simple instructions:  Download
    the file to DOS, use rawrite to write the tar file to diskette.
    Use 'tar zxvf /dev/<floppy>' to read the file from floppy, where
    <floppy> is the appropriate floppy device.  (PS0 is a 1.44 meg
    3.5" as A:, PS1 is a 1.44 meg as B:, at0 is a 1.2 meg as A:, at1
    is a 1.2 meg as B:.)

6)  To reboot your machine when running Linux, you should use the
    'reboot' command.  This makes sure to flush all caches to disk,
    and notifies other users that the system is going down (well, the
    last bit isn't real important).

    FAILURE TO DO THIS COULD RESULT IN BADLY CORRUPT FILESYSTEMS.

----------------------------------------------------------------------------

These instructions are not the best, but should be enough to get you
going.  If you have more questions, either post on comp.os.linux, or
send mail to me (jwinstea@jarthur.Claremont.EDU), or to Linus
(torvalds@cc.helsinki.fi).  Remember, the only stupid questions are
the ones that you don't ask.
```

12. CHANGE-0.95a

```
CHANGES IN THE LINUX v0.95a ROOT DISKETTE
Jim Winstead Jr. - March 17, 1992

This file mostly contains info about the changes in the root diskette
from Linux v0.95/0.12 to Linux v0.95a.

CHANGES

With the release of Linux v0.95a, the maintenance of the root diskette
has been assumed by Jim Winstead Jr. (jwinstea@jarthur.Claremont.EDU).
This means there are a few large changes between the Linux 0.95 and
0.12 root floppies and the Linux 0.95a root floppy.  These are
detailed (as much as I remember them) below:

-   'bash' has been replaced with 'ash', the BSD 4.3 /bin/sh.  This
    freed up nearly 200k on the root floppy.  However, there are
    some problems with 'ash' that haven't been resolved:

    - sometimes the backspace key will not work on a virtual
      console.  I've found that it usually works on all _but_ one
      console, so this is only a minor hinderance.

    - 'ash 'supports BSD-style job control, and this has not yet been
      adapted to Linux's more POSIXish job control.  This means
      that 'ash' does not yet support job control, but it's being
      worked upon.

-   'tar' and 'compress' are back on the root floppy.  'tar' is
    compressed, and both utilities are in /bin.

-   'pfdisk', a disk partitioner, was added to the root floppy.
    This makes it (almost) possible to install Linux on a machine
    without looking at another OS.

-   the file pager 'more' has been added to the floppy.  This was
    added because of the addition of some documentation files on
    the root floppy.

-   'cat' has been added to /bin.

-   many utilities have been moved from /usr/bin to /bin, to
    conform to the Linux Directory Structure Standard (v1.0).
    These utilities are ones that are 'vital to the restoration of
    other file systems in the case of a corrupting crash.'

-   'init' and 'update' have been moved to /etc from /bin.  This
    was done because neither program should be executed from the
    command line by any user, including root.  (That means don't
    put /etc in your PATH!)  This has been a matter of some
    controversy, but this is how it will stand until the Linux
    Standards mailing list/committee decides otherwise.

-   tty64, tty65, etc, have been renamed to ttys1, ttys2, etc.

-   the directory /INSTALL was added, which contains some
    documentation, and three simple shell scripts to make
    installing Linux on a hard drive partition easier.  These are:

    - 'mktree', which makes a directory tree on the specified
      mounted device.
    - 'mkdev' which creates the standard devices in the dev
      directory of the specified mounted device
    - 'install' which installs the programs on the root diskette
      to the specified mounted device

    These programs will normally be called with '<name> /mnt'.

-   rootdev is different than the one on v0.95.  A couple of days
    after the release of 0.95, a program called 'rdev' was posted
    to alt.os.linux that duplicated and extended the functionality
    of rootdev.  This was renamed to rootdev and replaces the old
    rootdev.

-   agetty was renamed to getty, to be consistent with common Unix
    practice.

-   an improved fdisk was added that correctly reports extended
    partitions,  (Thanks to Linus!)

-   /dev is complete, or at least more complete than the last few
    releases of the root diskette, which always seemed to be a
    major complaint.  :)

-   /etc/issue and /etc/motd have been expanded to be a little
    more informative.  (Yeah, I know, big deal! :)

-   chgrp was removed.  You can use chown to get the same effect,
    but you just have to specify an owner, too.

Many of these changes were discussed on alt.os.linux, or the Linux
Standards group, so they may look familiar.
```

```
If you have questions, problems, or complaints about the root
diskette, either post to alt.os.linux, or send mail to me at
jwinstea@jarthur.Claremont.EDU.

If you have questions, problems, or complaints about the boot diskette
or the kernel itself, post to alt.os.linux or send mail to Linus
Torvalds at torvalds@cc.helsinki.fi.

Remember, the only stupid questions are the ones you don't ask.

FUTURE CHANGES

I'm already anticipating some changes for the next release, so here's
a sneak preview:

-   shared libraries.  These are currently in alpha testing, and
    will hopefully free up some more room on the root floppy for
    more goodies.

-   a generic mtools might be added to the root floppy.

-   a better fdisk to replace the current fdisk/pfdisk pair.  You
    won't need to know your drive's geometry for this, and it will
    know about Linux extended partitions.

-   an improved sh.  I'm working on the backspace problem, and
    adding job control.  I'm also going to look at using the GNU
    readline library for input, as long as it doesn't add
    substantially to the size of sh.

-   init/getty/login may be removed from the root floppy.  The
    main reason they'll still on there is the backspace problem
    with ash.

-   improved installation documentation.  People have started work
    on this already - read alt.os.linux for previews.

-   more robust installation scripts.  The current ones are quick
    and dirty, and work well, but I'd like to add better ones.

-   miscellaneous utilities added.  I'd really like to add an
    editor to the root disk, but I haven't found one small enough.
    Any suggestions?

-   various other things that I can't remember right now.

Again, mail your questions, comments and suggestions about the root
diskette to me at jwinstea@jarthur.Claremont.EDU.
```

```
CHANGES-0.96

CHANGES IN THE LINUX v0.96 ROOT DISKETTE
Jim Winstead Jr. - 4 July 1992

This file mostly contains info about the changes in the root diskette
from Linux v0.95a to Linux v0.96.

CHANGES

With the release of Linux v0.95a, the maintenance of the root diskette
has been assumed by Jim Winstead Jr. (jwinstea@jarthur.Claremont.EDU).

This continues with the release of the Linux 0.96 release diskette.
The changes between the Linux 0.96 and Linux 0.95a root diskettes are
detailed below:

-   bash is back!  /bin/sh is now a symlink to /bin/bash.  ash was
    simple too buggy for general use as /bin/sh.  (This was likely
    a result of a sloppy port to Linux rather than any flaws with
    ash, but it seems silly to worry about ash when bash fits.)

-   GNU tar is not on the root disk.  Instead, the POSIX-defined
    utility 'pax' is included, which handles tar _and_ cpio
    archives.  There are symlinks from /bin/cpio and /bin/tar to
    /bin/pax to allow using the tar and cpio interfaces to pax.

    (The big change you'll notice is that pax does not support a
    'z' option for compressed tar files.  You will have to pipe
    them through 'uncompress' first.)

    This was done because pax is roughly 1/3 the size of GNU tar,
    and GNU tar offered nothing significant beyond what pax does.

-   the install script has been completely rewritten.  Now, it is
    much more intelligent, and tries to guide you along the path
    of installing Linux on your system.

-   split /etc/rc into /etc/rc and /etc/rc.local.  /etc/rc.local
    is the only one you should ever have need to change.

-   mount has been improved to accept a -a option.  This reads
    /etc/fstab and mounts the filesystems specified within,
    including swapping partitions.  See /etc/fstab to see how it
    works.

    Similar changes have been made to swapon to allow the 'swapon'
    of a single swap file/partition from /etc/fstab.
    As a result of these two improvements, /bin/mount -a and
    /bin/swapon -a have both been added to /etc/rc, and you
    shouldn't need to add additional mount commands to rc.local -
    use /etc/fstab instead.

    Thanks to Doug Quale for writing the new mount and swapon.

-   uncompress is really a link to compress this time, I screwed
    up last time.  oops!

-   I recompiled everything with GCC 2.2.2, and they are linked
    against shared libraries (located in /lib) - it is important
    that /lib be part of your root partition!

-   many of the small utilities are linked as 'impure'
    executables.  This saves a great deal of disk space, at the
    expense that they can't be demand-loaded or shared.  Most, if
    not all, of the utilities linked this way are very small and
    infrequently used, however, so the benefits far outweigh the
    small disadvantage there.

-   rootdev really is rdev this time.

-   /dev/MAKEDEV is a fairly generic script for making devices.
    It supercedes /INSTALL/mkdev from the 0.95a root disk, and
    really should be kept even after installation, because such
    things as the scsi tape devices are not made by default - this
    script allows you to make them when needed.

-   added the lp devices, scsi devices, and miscellaneous other
    devices.

-   included a new termcap file based upon the termcap file
    released with the setterm-0.96b utility.  Also included are
    the termcap entires for X terminals and generic vt100 entries.

If you have questions, problems, or complaints about the root
diskette, either post to comp.os.linux, or send mail to me at
jwinstea@jarthur.Claremont.EDU.

If you have questions, problems, or complaints about the boot diskette
or the kernel itself, post to comp.os.linux or send mail to Linus
Torvalds at torvalds@cc.helsinki.fi.

Remember, the only stupid questions are the ones you don't ask.

FUTURE CHANGES

I'm already anticipating some changes for the next release, so here's
```

a sneak preview:

- you probably won't notice, but I plan on cleaning up the source of some of the utilities, most noticeably shutdown, passwd and mkfs. Those are all pretty ugly.

- the install script will be improved. The current one was written rather rapidly, so there are parts of it I'm not entirely happy with.

- I'd like to write an update script that will allow people who have already installed Linux to update their binaries from the latest root disk. The install script could serve as a base for this, but is a little destructive at present. (It would simply copy over old binaries, etc.)

- the documentation on disk will be cleaned up, and possibly added to.

- fill in the gaps in the MAKEDEV script. (SCSI tapes, more pty devices.)

- the release after the extended filesystem is added to the Linux kernel, the root disk will use it. That means v0.98, if things go according to current plans. This is to allow time for bugs in the extended filesystem to filter out, and for the new mkfs and fsck to stabilize. (For those that don't know, the extended filesystem supports 4 terabyte partitions and long filenames, and is currently in alpha testing.)

Again, mail your questions, comments and suggestions about the root diskette to me at jwinstea@jarthur.Claremont.EDU.

13. CHANGES-0.97

```
CHANGES IN THE LINUX v0.97 ROOT DISKETTE
Jim Winstead Jr. - 4 August 1992

This file mostly contains info about the changes in the root diskette
from Linux v0.96 to Linux v0.97.

BUGS

    'mount' is broken in strange ways, particularly in passing
    options '-o whatever'.  I'm working on this.

CHANGES

With the release of Linux v0.95a, the maintenance of the root diskette
has been assumed by Jim Winstead Jr. (jwinstea@jarthur.Claremont.EDU).

This continues with the release of the Linux 0.97 release diskette.
The changes between the Linux 0.97 and Linux 0.96 root diskettes are
detailed below, and the changes in earlier releases are summarized
after that:

    -   many small binaries were added, including:

            cmp cut date env find head id install logname nice
            nohup pathchk printenv printf sed setserial sort sum
            tac tee tr tty uname uniq wall wc who whoami write yes

        (Some of these may have been on previous root disks -
        I don't have the motivation to double check that.  In any
        case, they are definitely on 0.97.  :)

    -   ps, w, uptime, and related utilities were removed.
        Because these programs rely very closely upon the
        kernel being used, they can be outdated quite quickly.

    -   migrated mount/umount/swapon from /bin to /etc.
        This conforms to common usage (only root can use these
        programs), and current standards.

    -   moved 'rootdev' to /usr/bin and renamed to 'setroot'.
        This reflects more common usage of the utility - it is no
        longer needed for inserting the root device in /etc/mtab, but
        it is still useful to change the root device of a kernel image.
    -   removed /lib/libhard.2.2.2 and moved /lib/libsoft.2.2.2 to
        /lib/libm.2.2.2, instead of using a symlink.

    -   upgraded efdisk and renamed to fdisk.
        efdisk was upgraded to v0.93, from Owen LeBlanc's MCC 0.96c
        interim release, with some small changes from me to support the
        -l flag, allowing it to completely replace fdisk.

    -   fixed compress to work with long filenames.
        Previous versions of compress would refuse to compress files
        with names longer than 12 characters - this was hardcoded in
        the source the FSF makes available.

    -   brought device names up to standards.
        Fixed some device names according to decisions made on
        the Linux Standards discussion list, particularly
        renaming /dev/lp* to /dev/par*, 'hard' /dev/fd*
        devices, /dev/bm (bus mouse), and fixing the numbering
        of /dev/ttys*.

    -   revised /etc/group.
        /etc/group now contains only the 'standard' group names
        discussed in the Linux Standards list.  Of special note is the
        renaming of the 'bin' group to 'obsolete'.

        Using the 'bin' group as a means of identifying executables is
        not recommended.  That is what the executable bits are designed
        to do.

    -   revised /etc/passed.
        /etc/passwd was changed as a result of the new /etc/group, and
        to eliminate unnecessary usernames - many groups were removed
        because using uid != 0 for important files is a security hole
        on NFS-mountable drives

    -   changes file permissions and ownerships.
        This was done to reflect changes in /etc/group and /etc/passwd.

    -   fixed up the install script where it was broken.
        All known major bugs were fixed.  Particularly where /usr was
        concerned.

    -   fixed the install documentation to refer to pax.

    -   minor gaffes from 0.96 fixed (/etc/getty linked with
        shared libs, correct file ownerships, etc)

If you have questions, problems, or complaints about the root
```

```
diskette, either post to comp.os.linux, or send mail to me at
jwinstea@jarthur.Claremont.EDU.

If you have questions, problems, or complaints about the boot diskette
or the kernel itself, post to comp.os.linux or send mail to Linus
Torvalds at torvalds@cc.helsinki.fi.

Remember, the only stupid questions are the ones you don't ask.

SUMMARY

This section very briefly summarizes previous changes.

0.95a -> 0.96
     - reintroduced GNU bash as /bin/sh
     - replaced GNU tar (/bin/tar) with POSIX pax utility
     - all-new and improved installation script
     - split /etc/rc into /etc/rc and /etc/rc.local
     - new mount/umount/swapon with auto-mounting - uses /etc/fstab
     - recompiled all utilities with shared libraries (gcc 2.2.2)
     - added /dev/MAKEDEV - device creation script
     - all new /etc/termcap using "tc=con-unk" entries

FUTURE CHANGES

I'm already anticipating some changes for future releases, so here's
a sneak preview:

-    the install script will be improved.  The current one was
     written rather rapidly, so there are parts of it I'm not
     entirely happy with.

-    I'd like to write an update script that will allow people who
     have already installed Linux to update their binaries from the
     latest root disk.  The install script could serve as a base
     for this, but is a little destructive at present.  (It would
     simply copy over old binaries, etc.)

-    program to allow the 'intro' login to be more interactive, and
     will also serve more general purposes (it will be a generic
     curses-based file-selector with support for file descriptions).

-    the documentation on disk will be cleaned up, and possibly
     added to.

-    fill in the gaps in the MAKEDEV script.  (SCSI tapes, more pty
     devices.)

-    the release after the extended filesystem is added to the
```

```
    Linux kernel, the root disk will use it.  That means v0.98, if
    things go according to current plans.  This is to allow time
    for bugs in the extended filesystem to filter out, and for the
    new mkfs and fsck to stabilize.  (For those that don't know,
    the extended filesystem supports 4 terabyte partitions and long
    filenames, and is currently in testing.)

Again, mail your questions, comments and suggestions about the root
diskette to me at jwinstea@jarthur.Claremont.EDU.
```

14. CHANGES-0.97.1

```
CHANGES IN THE LINUX v0.97.1 ROOT DISKETTE
Jim Winstead Jr. - 16 August 1992

This file mostly contains info about the changes in the root diskette
from Linux v0.97 to Linux v0.97.1.

CHANGES

With the release of Linux v0.95a, the maintenance of the root diskette
has been assumed by Jim Winstead Jr. (jwinstea@jarthur.Claremont.EDU).

This continues with the release of the Linux 0.97.1 release diskette.
The changes between the Linux 0.97.1 and Linux 0.97 root diskettes are
detailed below, and the changes in earlier releases are summarized
after that:

    -   fixed mount.
        A few hours after releasing 0.97, I figured out why
        mount wasn't working correctly with the MS-DOS
        filesystem.  'mount' now does things correctly, and
        will even pass 'odd' mount options correctly (i.e. the
        conv=something option for the MS-DOS fs).  Thanks to
        Werner Almesberger for providing smount, from which
        most of my changes to Doug Quale's mount were taken.

    -   made passwd sgid system.
        I forgot to last time, which made /etc/passwd belong to
        whatever group the person who last changed their password
        belonged to. Thanks to Scott Mace (emace@tenet.edu) for
        spotting this one.

    -   fixed bug in /etc/termcap.
        The 'is' and 'rs' strings had an extra colon in them,
        and the k? strings were wrong. Special thanks to
        Jaakko.Hyvatti@Helsinki.FI for pointing this out.

    -   fixed pax (some).
        Pax was broken in a few spots, and I've been trying to
        clean it up.  In particular, it would give some false
        errors because it would try to create some directories
        twice.  Duh.  I'm also trying to bring it up to POSIX
        compliance, since it's quite out of date.
    -   fixed problems with GNU fileutilities.
        The GNU fileutilities (cp, du and ls in particular)
        were making some bad assumptions about the blocksize
        on filesystems.  I think I've tracked that all down.
        Also, fixed ls so it recognizes the dir and vdir
        counterparts using argv[0] instead of seperate filenames.
        I was also able to trim some size off a few utilities
        due to functions available in libc.

    -   compiled GNU text utilities to use getopt/regex from libc.
        I also fixed cat so you can use it with the various
        options (like -v, etc).  This saved over 30k. (Wow!)

    -   compiled GNU shell utilities to use getopt/regex from libc.
        This saved another 30k.  Wow again!

    -   compiled GNU tput to use termcap from shared libs.
        A lot of the changes to be like this, don't they?  :)
        Saved about 4k here.

    -   compiled sed with -N.
        Saved 6k.  :)

    -   added creation of user account to /INSTALL/install.
        The install script now asks for a username to create an
        account for and sets it up.  This should encourage not
        using 'root' all the time.

If you have questions, problems, or complaints about the root
diskette, either post to comp.os.linux, or send mail to me at
jwinstea@jarthur.Claremont.EDU.

If you have questions, problems, or complaints about the boot diskette
or the kernel itself, post to comp.os.linux or send mail to Linus
Torvalds at torvalds@cc.helsinki.fi.

Remember, the only stupid questions are the ones you don't ask.

SUMMARY

This section very briefly summarizes previous changes.

0.96 -> 0.97
    - many small binaries were added.
    - ps, w, uptime, and related utilities were removed.
    - migrated mount/umount/swapon from /bin to /etc.
    - moved 'rootdev' to /usr/bin and renamed to 'setroot'.
    - removed /lib/libhard.2.2.2 and moved /lib/libsoft.2.2.2 to
      /lib/libm.2.2.2, instead of using a symlink.
    - upgraded efdisk and renamed to fdisk.
    - fixed compress to work with long filenames.
    - brought device names up to standards.
    - revised /etc/group.
    - revised /etc/passed.
    - changes file permissions and ownerships.
    - fixed up the install script where it was broken.
    - fixed the install documentation to refer to pax.
    - minor gaffes from 0.96 fixed (/etc/getty linked with shared
      libs, correct file ownerships, etc)

0.95a -> 0.96
    - reintroduced GNU bash as /bin/sh
    - replaced GNU tar (/bin/tar) with POSIX pax utility
    - all-new and improved installation script
    - split /etc/rc into /etc/rc and /etc/rc.local
    - new mount/umount/swapon with auto-mounting - uses /etc/fstab
    - recompiled all utilities with shared libraries (gcc 2.2.2)
    - added /dev/MAKEDEV - device creation script
    - all new /etc/termcap using "tc=con-unk" entries

FUTURE CHANGES

I'm already anticipating some changes for future releases, so here's
a sneak preview:

-   the install script will be improved.  The current one was
    written rather rapidly, so there are parts of it I'm not
    entirely happy with.  Michael K. Johnson (johnsonm@stolaf.edu)
    has said he is working on this and the update script (below).

-   I'd like to write an update script that will allow people who
    have already installed Linux to update their binaries from the
    latest root disk.  The install script could serve as a base
    for this, but is a little destructive at present.  (It would
    simply copy over old binaries, etc.)

-   program to allow the 'intro' login to be more interactive, and
    will also serve more general purposes (it will be a generic
    curses-based file-selector with support for file descriptions).

-   the documentation on disk will be cleaned up, and possibly
    added to.

-   fill in the gaps in the MAKEDEV script.  (SCSI tapes, more pty
    devices.)

-   the release after the extended filesystem is added to the
    Linux kernel, the root disk will use it.  That means v0.98, if
    things go according to current plans.  This is to allow time
    for bugs in the extended filesystem to filter out, and for the
    new mkfs and fsck to stabilize.  (For those that don't know,
    the extended filesystem supports 4 terabyte partitions and long
    filenames, and is currently in testing.)

Again, mail your questions, comments and suggestions about the root
diskette to me at jwinstea@jarthur.Claremont.EDU.
```

15. INFO-SHEET-1.13.1992

```
LINUX INFORMATION SHEET
(last updated 13 Jan 1992)

1. WHAT IS LINUX 0.12
    LINUX 0.12 is a freely distributable UNIX clone.  It implements a
subset of System V and POSIX functionality.  LINUX has been written
from scratch, and therefore does not contain any AT&T or MINIX
code--not in the kernel, the compiler, the utilities, or the libraries.
For this reason it can be made available with the complete source code
via anonymous FTP.  LINUX runs only on 386/486 AT-bus machines; porting
to non-Intel architectures is likely to be difficult, as the kernel
makes extensive use of 386 memory management and task primitives.

     Version 0.12 is still a beta release, but it already provides much
of the functionality of a System V.3 kernel.  For example, various
users have been able to port programs such as bison/flex without having
to modify code at all.  Another indication of its maturity is that
it is now possible to do LINUX kernel development using LINUX itself
and freely-available programming tools.

2. LINUX features
  - System call compatible with a subset of System V and POSIX
  - Full multiprogramming (multiple programs can run at once)
  - Memory paging with copy-on-write
  - Demand loading of executables
  - Page sharing of executables
  - Virtual memory: swapping to disk when out of RAM
  - POSIX job control
  - virtual consoles on EGA/VGA screens
  - pty's
  - some 387-emulation
  - ANSI compliant C compiler (gcc)
  - A complete set of compiler writing tools
    (bison as yacc-replacement, flex as lex replacement)
  - The GNU 'Bourne again' shell (bash)
  - Micro emacs
  - most utilities you need for development
    (cat, cp, kermit, ls, make, etc.)
  - Over 200 library procedures (atoi, fork, malloc, read, stdio, etc.)
  - Currently 4 national keyboards: Finnish/US/German/French
  - Full source code (in C) for the OS is freely distributable
  - Full source code of the tools can be gotten from many anonymous ftp sites
    (Almost the entire suite of GNU programs has been ported to Linux.)
  - Runs in protected mode on 386 and above
  - Support for extended memory up to 16M on 386 and above
  - RS-232 serial line support with terminal emulation, kermit, zmodem, etc.
  - Supports the real time clock


3. HARDWARE REQUIRED
   - A 386 or 486 machine with an AT-bus.  (EISA will probably work, also,
     but you will need an AT-bus hard disk controller.)
     Both DX and SX processors will work.
   - A hard disk implementing the standard AT hard disk interface--
     for example, an IDE drive.  SCSI drives are not supported yet.
   - A high-density disk drive--either 5.25" (1.2MB) or 3.5" (1.44MB).
   - At least 2 megabytes of RAM.  (LINUX will boot in 2 Mb.  To use
     gcc 4 MB is a good idea.)
   - Any video card of the following: Hercules,CGA,EGA,VGA

In addition, LINUX supports
   - Up to two serial lines
   - A real time clock

4. PARTIAL LIST OF UTILITIES INCLUDED IN OR AVAILABLE FOR LINUX 0.12
   - The MTOOLS package (reading/writing to DOS filesystems)
   - The complete GNU filetools (ls, cat, cp, mv, ...)
   - The GNU C compiler with GNU assembler, linker, ar, ...
   - bison
   - flex
   - rcs
   - pmake (BSD 4.3 Reno/BSD 4.4  make)
   - kermit
   - Micro emacs
   - less
   - mkfs
   - fsck
   - mount/umount


5. LINUX BINARIES
    The LINUX binaries and sources are available at three
    anonymous FTP sites. These are:

    nic.funet.fi:/pub/OS/Linux
    tsx-11.mit.edu:/pub/linux
    tupac-amaru.informatik.rwth-aachen.de:/pub/msdos/replace
```

6. LEGAL STATUS OF LINUX

Although LINUX is supplied with the complete source code, it is copyrighted software. Unlike MINIX, however, it is available for free, provided you obey to the rules specified in the LINUX copyright.

7. NEWS ABOUT LINUX

Since LINUX's introduction to the public there has been a rapidly growing mailing list, "linux-activists@niksula.hut.fi". To subscribe to this list, mail to "linux-activists-request@niksula.hut.fi". If the traffic in this lists increases further, there are plans to swap (at least partially) over to comp.os.misc, so watch out for any LINUX articles in this group. For the current status of LINUX, do "finger torvalds@kruuna.helsinki.fi".

8. FUTURE PLANS

Work is underway on LINUX version 1.0, which will close some of the gaps in the present implementation. Various people are currently working on:

- A virtual filesystem layer
- STREAMS
- init/getty/login
- Interprocess communication
- IEEE POSIX P1003.1 / P1003.2 compatibility
- SCSI support

If you want to help, join the mailing list.