The linux-kernel mailing list FAQ

These frequently asked questions are divided in various categories. Please contribute any category and Q/A that you may find relevant. You can also add your answer to any question that has already been answered, if you have additional **information** to contribute.

The base (U.S.) site is now available at http://www.tux.org/lkml/. Many thanks to Sam Chessman and David Niemi for hosting the FAQ on a high-bandwidth, professionally managed Linux server.

The present FAQ builds on the previous linux-kernel mailing list FAQ maintained by **Frohwalt Egerer**. Frohwalt has gone on to other computing activities and consequently we will slowly merge his excellent but no longer maintained FAQ in the present one.

Hot off the Presses

vger.rutgers.edu is dead!

The machine which hosted the linux-kernel mailing list is believed to have suffered a catastrophic disk failure. The list has been moved to vger.kernel.org instead. Edit your procmail recipes.
Also note that digest services are not available on vger.kernel.org and are unlikely to be added due to the load these services place on the server. Use the list archives, or use procmail. If you set up a digest service, please inform the FAQ maintainer.

Go to http://www.atnf.csiro.au/~rgooch/linux/docs/kernel-newsflash.html for newflashes about official kernel releases.

Read this before complaining to linux-kernel about compile problems. Chances are a thousand other people have noticed and the fix is already published.

Index

Basic Linux kernel documentation

The following are **Linux kernel** related documents, which you should take a look at **before** you post to the linux-kernel mailing list:

- **The Linux Kernel Hackers' Guide**, compiled by Michael K. Johnson of RedHat fame. Includes among other documents selected Q/A's from the linux-kernel mailing list.
- **The Linux Kernel** book, by David A. Rusling, available in various formats from the Linux Documentation Project and mirrors. Still being worked on, but explains clearly the main structure of the Linux kernel.
- **The Linux FAQ** by Robert Kiesling, also available from the LDP. Has many, many high quality Q/A's.
- **The Linux Kernel HOWTO** by Brian Ward, also available from the LDP. Fundamental reading for anybody wanting to post to the linux-kernel mailing list.
- Various Linux **HOWTOs** on specific questions, such as the **BogoMips mini-HOWTO** by Wim van Dorst. These are all by definition LDP documents.
- The Linux **kernel source code** for any particular kernel version that you may be using. Note that there is a /Documentation directory which holds some very useful text files about drivers, etc. Also check the MAINTAINERS file in the kernel source root directory.
- Some drivers even have **Web pages**, with additional up to date information e.g. the network drivers by Donald Becker, etc. Check the Hardware section in the LDP site.
- Similarly, Linux implementations for some CPU architectures have dedicated **Web pages, mailing lists**, and sometimes even a HOWTO e.g. the **Linux Alpha HOWTO** by Neal Crook. Check the LDP site and its mirrors for Web links to the various architecture specific sites.
- ***Linux device drivers***, a book written by Alessandro Rubini. C. Scott Ananian reviewed it for Amazon.com.
- ***Linux kernel internals***, a book by Michael Beck (Editor) et al. Also reviewed for Amazon.com.

Contributors and some special expressions

This is the list of contributors to this FAQ. They are listed in alphabetic order of their abbreviations, used in the Answers sections below to identify the author(s) of each answer.

- `AC :` Alan Cox
- ADB: Andrew D. Balsa
- `CP :` Colin Plumb
- `DBE:` Daniel Bergstrom
- `DSM:` David S. Miller (co-postmaster)
- `DW :` David Woodhouse
- `KGB:` Krzysztof G. Baranowski

- `MEA:` Matti E. Aarnio (co-postmaster)
- `MRW:` Matthew Wilcox
- `PG :` Paul Gortmaker
- `REG:` Richard E. Gooch <rgooch@atnf.csiro.au> (FAQ maintainer)
- `REW:` Roger E. Wolff
- `RRR:` Rafael R. Reilova
- `TJ :` Trevor Johnson
- `TYT:` Theodore Y. Ts'o

Some English expressions for non-native English readers. Many of these (and far more) may be obtained from the Jargon File:

- AFAIK = As Far As I Know
- AKA = Also Known As
- ASAP = As Soon As Possible
- COLA = comp.os.linux.announce (newsgroup)
- ETA = Estimated Time of Arrival
- FAQ = Frequently Asked Question
- FUD = Fear, Uncertainty and Doubt
- FWIW = For What It's Worth
- FYI = For Your Information
- IANAL = I Am Not A Lawyer
- IIRC = If I Remember Correctly
- IMHO = In My Humble Opinion
- IMNSHO = In My Not-So-Humble Opinion
- IOW = In Other Words
- LART = Luser Attitude Readjustment Tool (quoting Al Viro: "Anything you use to forcibly implant the clue into the place where luser's head is")
- OTOH = On The Other Hand
- PEBKAC = Problem Exists Between Keyboard And Chair
- RSN = Real Soon Now
- RTFM = Read the Fucking Manual (original definition) or Read The Fine Manual (if you want to pretend to be polite)
- TANSTAAFL = There Ain't No Such Thing As A Free Lunch (contributed by David Niemi, quoting Robert Heinlein in his Sci-Fi novel 'The Moon is a Harsh Mistress')
- THX = Thanks (thank you)
- TIA = Thanks In Advance
- WIP = Work In Progress
- WRT = With Respect To

Question Index

whole. It also allows frequent independent revisions, which would be impossible on the FAQ itself.

Note that writing the longer text/Web page on some relevant Linux kernel topic and providing a Q/A in this section confers you instant **Guru status**. Some people would *kill* for this. Now go and write your stuff. ;)

Section 12- Kernel Programming Questions

Answers to common questions about kernel programming details. See also Tigran Aivazian's page on kernel programming.

Section 13- Mysterious kernel messages

We sometimes get these messages in our system logs and wonder what they mean...

Answers

Section 1 - General questions

1. **Why do you use "GNU/Linux" sometimes and just "Linux" in other parts of the FAQ?**
   - (ADB) In this FAQ, we have tried to use the word "Linux" or the expression "Linux kernel" to designate the kernel, and GNU/Linux to designate the entire body of GNU/GPL'ed OS software, as found in the various distributions. We prefer to call a cat, a cat, and a GNU, a GNU. ;-)
     The purpose of the FAQ is to provide information on the Linux kernel and avoid debates on e.g. semantics issues. Further discussion of the relationship between GNU software and Linux can be found at http://www.gnu.org/gnu/linux-and-gnu.html.
     BTW, it seems many people forget that the linux kernel mailing list is a forum for discussion of kernel-related matters, not GNU/Linux in general; please do not bring up this subject on the list.
2. **Why is the Linux kernel written in C/assembly?**
   - (ADB) For many reasons, some practical, others theoretical. The practical reasons first: when Linus began writing Linux, what he had available was a 386, Minix (a minimal OS designed by Andrew Tanenbaum for OS design teaching purposes) and gcc. The theoretical reasons: some small parts of

any OS kernel will always be written in assembly language, because they are too dependent on the hardware to be coded in C; for example, CPU and virtual memory setup. Or because we are dealing with very short routines that must be implemented in the fastest possible code e.g. the stubs for the "top half" interrupt handlers. WRT C, OS designers (since Thompson and Ritchie first wrote UNIX) have traditionally used C to implement as many OS kernel routines as possible. In this sense C can be considered the "canonical" language for OS kernel implementation, and particularly for UNIX variants.

3. **Why don't we rewrite it all in assembly language for processor Mega666?**
   - (ADB) Basically because we wouldn't gain much in terms of efficiency, but would lose a lot in terms of ease of maintenance and readability of the source code. Gcc is actually quite efficient, when we look at the assembler code generated. You are referred to Andrew Tanenbaum's book "Structured Computer Organization", 3rd ed., pages 401-404, for a more detailed comparison of the use of high level languages vs. assembly language in the implementation of OS's. There are a number of references on the subject at the end of the book, too.
4. **Why don't we rewrite the Linux kernel in C++?**
   - (ADB) Again, this has to do with practical and theoretical reasons. On the practical side, when Linux got started gcc didn't have an efficient C++ implementation, and some people would argue that even today it doesn't. Also there are many more C programmers than C++ programmers around. On theoretical grounds, examples of OS's implemented in Object Oriented languages are rare (Java-OS and Oberon System 3 come to mind), and the advantages of this approach are not quite clear cut (for OS design, that is; for GUI implementation KDE is a good example that C++ beats plain C any day).
   - (REW) In the dark old days, in the time that most of you hadn't even heard of the word "Linux", the kernel was once modified to be compiled under g++. That lasted for a few revisions. People complained about the performance drop. It turned out that compiling a piece of C code with g++ would give you worse code. It shouldn't have made a difference, but it did. Been there, done that.
5. **Why is the Linux kernel monolithic? Why don't we rewrite it as a microkernel?**
   - (ADB) No opinions here, just a few pointers. Linux has been implemented as a "personality" on top of a modified version of the Mach3 microkernel. This is a fully functional piece of code, known as MkLinux. The project was in part funded by Apple, and as such it was running at first on PowerPC Macs. But an x86 version is available, with fully open source code. Similarly, the Hurd (the GNU kernel) is being implemented on top of Mach3. The Debian Project is working on a full distribution for the Hurd.
     There is a historical Usenet thread related to this subject, dating back from 1992, with posts from Linus, Andrew Tanenbaum, Roger Wolff, Theodore Y T'so, David Miller and others. Nice reading on a rainy afternoon. It's fascinating to see how some predictions (which seemed rather reasonable at the time) have proved wrong over the years (for example, that we would all be using RISC chips by 1998).
6. **What is an experimental kernel version?**
   - (ADB) Linux kernel versions are divided in two series: experimental (odd series e.g. 1.3.xx or 2.1.x) and production (even series e.g. 1.2.xx, 2.0.xx or the forthcoming 2.2.x). The experimental series are fast moving versions which are used to test new features, algorithms, device drivers, etc. By their own nature the experimental kernels may behave in unpredictable ways, so one may experience data losses, random machine lockups, etc.
7. **What is a production kernel?**
   - (ADB) Production or stable kernels have a well defined feature set, a low number of known bugs, and tried and proven drivers. They are released less frequently than the experimental kernels, but even so some "vintages" are considered better than others. GNU/Linux distributions are usually based on chosen stable kernel versions, not necessarily the latest production version.

8. **What is a feature freeze?**
   - (ADB) A feature freeze is when Linus announces on the linux-kernel list that he will not consider any more features until the release of a new stable kernel version. Usually the net effect of such an announcement is that on the following days people on the list propose a flurry of new features before Linus really enforces the feature freeze. ;-)
9. **What is a code freeze?**
   - (ADB) A code freeze is more restrictive than a feature freeze; it means only severe bug fixes are accepted. This is a short phase that usually precedes the creation of a new stable kernel tree.
10. **What is a f.g.hh*pre*i kernel?**
    - (ADB) These are intermediate pre-release versions of version f.g.hh. Note that usually i < 5, but e.g. 2.0.34*pre*i was available with i = 1 to 16. Sometimes "***pre***" is replaced by the initials of the developer putting together the kernel revision, e.g. 2.1.105*ac*4 means the 4th intermediate release of kernel version 2.1.105 by Alan Cox.
11. **Where do I get the latest kernel source?**
    - (ADB) The primary site for the Linux kernel (experimental and production) sources is hosted by Transmeta (the company Linus Torvalds works for) on a dedicated Web server at http://www.kernel.org/. This site is mirrored across the world, and has pointers to mirrors for each country. You can go directly to a mirror for your country by going to http://www.CODE.kernel.org/ where "CODE" is the appropriate country code. For example, "au" is the country code for Australia, so the principle mirror site for Australia is http://www.au.kernel.org/
12. **Where do I get extra kernel patches?**
    - (REG) There are many places which provide various extra patches to the kernel for new features. One fairly good archive is available here.
13. **What is a patch?**
    - (RRR) A patch file (as it refers to the Linux kernel) is an ASCII text file that contains the differences between the original code and the new code, plus some additional information such as filenames and line numbers. The patch program (man patch) can then apply the patch to an existing kernel source tree.
14. **How do I make a patch suitable for the linux kernel list?**
    - (RRR) To make a patch you use the diff program (read the info file for diff). The easiest way to do this is to set up two source trees under /usr/src, set a symlink "/usr/src/linux" to point to the modified tree, and diff one tree against the other. The file /usr/src/Documentation/CodingStyle has more specific information, read it. Things to remember:
      - Always specify unified (-u) diff format.
      - Avoid making formatting changes to the source that make the diff needlessly larger. Watch out for editors that convert tabs to spaces or vice versa.
      - Unless you have specific reasons, diff against the latest official source tree. Otherwise, your patch is likely to be ignored. Either way, specify in your post against what you've diff'ed.
      - Make sure your diff includes only the intended changes in your patch, not every other patch you have made to your source tree. Usually patches are limited to a few files, or directories. It is best to only diff the relevant files i.e. if I only made changes to the file driver_xyz.c under drivers/net, then I would use the following commands (assuming you have the original source tree named "linux-2.1.105", and the modified tree pointed at by the symlink "linux"):

        ```
        cd /usr/src
        diff -u linux-2.1.105/drivers/net/driver_xyz.c           \
                        linux/drivers/net/driver_xyz.c  > my_patch
        ```

      - The following two should go without saying: the arguments to diff are first source (the original, unmodified file(s)), and then destination (your modified version of the file(s)), otherwise you get a reversed patch (and lots of people wondering what you're smoking). Also, make sure your patch applies and compiles cleanly.

- Of course you need to set up two identical source directories to be able to diff the tree later. A nice trick -- requiring a little bit of consideration, though -- is to create the modified source tree from hard links to the original source tree:

  ```
  tar xzvf linux-2.1.anything.tar.gz
  mv linux linux-2.1.anything.orig
  cp -al linux-2.1.anything.orig linux-2.1.anything
  ```

  This will hardlink every source file from the original tree to a new location; it is very fast, since it does not need to create some 80+ megabytes of files. You can now apply patches to the linux-2.1.anything source tree, since patch does not change the original files but move them to *filename*`.orig`, so the contents of the hard-linked file will not be changed.

  Assuming that your editor does the same thing, too (moving original files to backup files before writing out changed ones) you can also freely edit within the hardlinked tree. If your editor does not handle files this way, you need to make a copy of each file before editing it, like this:

  ```
  cp driver_xyz.c temporary; mv temporary driver_xyz.c
  ```

  You can use file permissions to remind you to do this. Just remove write permissions from all the files in the directory you are working in:

  ```
  chmod -w *.c
  ```

  The changed tree can be diffed at high speed, since most files don't just have indentical contents, they are identical files in both trees. Naturally removing that tree is quite fast, too. Thanks to Janos Farkas <chexum@shadow.banki.hu> for this trick.
- Finally, review the patch file (the format is not that complicated) before posting, and include all relevant information as to the nature of the patch. In particular, specify: why is this patch needed/useful, and what exactly does it fix/improve.
- Depending on the size of your patch, you can either send it as plain ASCII text, or if it is longer than a page, gzip it (man gzip) and send it as a MIME encoded attachment.

- (ADB) Do not post it as a base64 MIME attachment. People using pine won't be able to read it (e.g. Linus?). Use uuencoding for large patches. Thanks to Andrew E. Mileski for the tip.

15. **How do I apply a patch?**
    - (RRR) To apply kernel patches please take a look at the kernel README file (/usr/src/linux/README) under "Installing the kernel". There is also a good explanation on the Linux HQ Project site.
16. **Everybody is talking about a "CVS tree at vger". What's vger? And what's this CVS tree?**
    - (ADB) Apart from the Star Trek reference, vger is a Web server that has been faithfully serving the GNU/Linux community for quite some time now. Andrew Tridgell has setup a CVS server for the Linux kernel source code. Under CVS, changes to files are logged in a tree structure. Hence the "CVS tree at vger" represents the Linux kernel source and assorted patches, in a structured fashion, as logged by the CVS server.
17. **Where can I find more information about CVS?**
    - (ADB) Most GNU/Linux distributions include CVS. You can also check the CVS Bubbles page.

18. **Is there a CVS tutorial somewhere?**
    - (ADB) Here is a CVS tutorial which you can find online:
        - An interactive CVS tutorial.

      Getting a general idea of how CVS works takes about 15 minutes (highly recommended). Note that there are various graphical front ends to CVS, so you don't have to learn the usual assortment of cryptic commands.
19. **How do I get my patch into the kernel?**
    - (RRR) Depending on your patch there are several ways to get it into the kernel. The first thing is to determine under which maintainer does your code fall into (look in the MAINTANERS file). If your patch is only a small bugfix and you're sure that it is 'obviously correct', then by all means send it to the appropriate maintainer and post it to the list. If there is urgency to the bugfix (i.e. a major security hole) you can also send it to Linus directly, but remember he's likely to ignore random patches unless they are "obviously correct" to him, have the maintainer's approval, or have been well tested and meet the first condition. I case you're wondering what constitutes well tested, here's another important bit: one purpose of the list is to get patches peer-reviewed and well-tested. Now, if your patch is relatively big, i.e. a rewrite of a large code section or a new device driver, then to conserve bandwidth and disk-space just post an announcement to the list with a link to the patch. Lastly, if you're not too sure about your patch yet, want some feedback from the maintainer, or wish to avoid open-season flaming on work-in-progress, then use private email.
      There's also a web-based bug/patch-tracking system for the linux kernel called jitterbug. Take a look at http://samba.anu.edu.au/linux-patches/ for more info.
20. **Why does the kernel tarball contain a directory called linux/ instead of linux-x.y.z/ ?**
    - (DW) Because that's the way Linus wants it. It makes applying many consecutive patches simpler, because the directory doesn't need to be renamed each time, and it also makes life easier for Linus.

Section 2 - Driver specific questions

1. **Driver such and such is broken!**
    - (RRR) Try to be more specific. Please, provide information on your particular setup (see Q's How do I make a bug report?) Also see the Q: "kernel x.y.z broken!" below.
    - (ADB) That's the worst possible way to start a thread. Please try to reach the author of the driver first and report the "broken" driver to him. Constructive criticism is welcome, usually.
2. **Here is a new driver for hardware XYZ.**
    - (REW) Good work! Please try to find a few people that also have the XYZ hardware and have them test it on their configuration (e.g. by posting a message on a newsgroup). No it won't go in the standard kernel before some people have tested it.
      Testing will take a while. In the mean time, kernel development will continue, and you will have to rewrite your patch for the most recent version before Linus might consider it.
      As a whole new driver is most likely more than a few pages long, we'd prefer it if you would put the actual driver up for ftp instead of posting it to the list. Post the URL and the description that tells us what your driver does for which hardware.
3. **Is there support for my card TW-345 model C in kernel version f.g.hh?**
    - (REW) First check if your card is detected at boot time. It usually is. Second see if you might need to configure something like modules.conf for your card. Third see if there is a file with the card name in the kernel sources. (e.g. you have a Buslogic card, and there is a buslogic.c file in the kernel sources, you're in luck.). Next, grep for the manufacturer name through ALL the kernel sources. And try the model number of your card. Also try to find the largest chip on your card and grep for the chip number on that thing. Realize that 53C80 chips might be named 5380 in the kernel. Other chips

don't have their middle name removed.
Nothing yet? Now check DejaNews, using the same arguments you used to grep the kernel source. There are 99.99% chances that somebody has exactly the same card TW-345 model C.
Ok. That's what you can do without bothering anyone. If all this doesn't lead somewhere, you should really ask this question on a newsgroup like comp.os.linux.hardware.

4. **Who maintains driver such and such?**
    - (RRR) Have a look at the /usr/src/linux/MAINTAINERS file, this is the most authoritative source. Also check the source code for the driver itself; in both cases, check the latest version of the kernel that you have available. Some drivers have specific Web pages and sometimes even a dedicated mailing list. Check those first. If you cannot contact the maintainer then *as a last resort* post a short message to the list. In any case, keep in mind that maintainers are usually very busy people and most of them work on Linux for free and in their spare time, so don't expect an immediate response. Some maintainers get just too many mails in too small periods of time to be able to answer them all, so please be kind to them.
5. **I want to write a driver for card TW-345 model C, how do I get started?**
    - (REW) Good initiative! First a piece of advise: are you up to this? Ten times as many projects like this get started as get finished. Also, make sure that you're not doing double work. Make sure that such a driver is not already available: read Q/A 2.3 above...
    First prepare yourself. Get the docs, read them (OK, you're allowed to start skipping stuff if you've gotten to the part "detailed register descriptions"). Next, get the Linux kernel source, find a driver that drives similar hardware to the one you're going to work on, and read THAT. (I usually use the smallest one I can find: wc -l *.c | sort -n | head -4).
    Ok. You've thought about it. Now the question is, do you have technical documentation for your card? You can reverse engineer the driver for MS operating systems, but having the documentation is MUCH easier.
    In the dark old ages (70's to middle of the 80's), you got a complete technical description with every card you could get. This is no longer the case. Anyway, contact your vendor and politely ask them for the "device driver kit" or the "technical manual" for the card.
    Try the head office and your local office at the same time. Local offices occasionally have bad photo copies that they give out before you get an official rejection from the head office. In that case whom you got the documentation from becomes confidential information. Don't put the guy's name in the source.
    If you can't get the technical documentation, consider giving up and investing in a competitors product (and tell the manufacturer about this). Not given up yet? Ok. Next step is to find out what the DOS driver does. Try to get the card to work while you run it in a microsoft emulator (dosemu or WINE). This will allow you to program these tools to log the I/O accesses of the driver. This will give you a large list of I/O accesses that the driver did. If you're good, you might be able to see patterns, and deduce how the driver works. From there you might be able to write a working driver. Good luck! You'll need it.
6. **I want to get the docs, but they want me to sign an NDA (Non-Disclosure Agreement).**
    - (REW) Some people find this a tremendous problem. Some companies just want to know who has the docs to their hardware, and don't mind if you write a GPL-ed driver. In that case, there is really no problem: just tell them what you intend to do and ask them to acknowledge in writing that they've understood what you're saying. In that case, you can get your driver into the standard kernel, but you cannot send out the docs to anybody who wants to work on the driver. They will have to rely on the comments in the source.
    Other companies (just like Netscape) themselves signed NDAs that forbids them to disclose information to you.
    Some really think that they have trade secrets in the interface towards the software, and intend to

keep them secret. Those won't allow you to write a driver and then put the source on the net. Be careful with these.
- (ADB) The first and only NDA I ever received instantly found its way to the wastebasket. I would advise anybody who gets an NDA to refuse to sign it, if it refers to anything that may/will be put under GNU/GPL. Of course, for contract work this doesn't apply.

7. **I want/need/must have a driver for card TW-345 model C! Won't anybody write one for me?**
   - (REW) Some Linux developers will settle for a beer, and develop the driver for you. Others want a "free sample" of the hardware and will then go ahead and write the driver.
     If you need more than a few of the cards or you manufacture the cards yourself, you can consider paying one of the commercial Linux device driver companies to get a commercially backed, officially maintained device driver.
8. **What's this major/minor device number thing?**
   - (REG) Device numbers are the traditional Unix way to provide a mapping between the filesystem and device drivers. A device number is a combination of a major number and a minor number. Currently Linux has 8 bit majors and minors. When you open a device file (character or block device) the kernel takes the major number from the inode and indexes into a table of driver structure pointers. The specific driver structure is then used to call the driver open() method, which in turn may interpret the minor number. There are two tables: one for character devices and one for block devices, each are 256 entries maximum. Obviously, there must be agreement between device numbers used in a driver and files in /dev. The kernel source has the file Documentation/devices.tex which lists all the official major and minor numbers. H. Peter Anvin (HPA) maintains this list. If you write a new driver (for public consumption), you will need to get a major number allocated by HPA. See the Q/A on devfs for an improved (IMHO) mechanism for handling device drivers.
9. **Why aren't WinModems supported?**
   - (REG, quoting Edward S. Marshall) The problem is the lack of specifications for this hardware. Most companies producing so-called "WinModems" refuse to provide specifications which would allow non-Microsoft operating systems to use them.
     The basic issue is that they don't work like a traditional modem; they don't have a DSP, and make the CPU do all the work. Hence, you can't talk to them like a traditional modem, and you -need- to run the modem driver as a realtime task, or you'll have serious data loss issues under any kind of load. They're simply a poor design.
   - (REG) Note that some people have been putting effort into reverse engineering some WinModems, so you may be lucky and find that yours is now supported. If not, it's time to get a refund and buy a real modem.
     Note that modems have to be approved by the appropriate statutory or regulatory body for standards compliance (to make sure they don't send crap down the line and blow up the exchange). With WinModems, the driver software needs to be certified as well as the hardware. It's harder to get approval for Open Source drivers, since it usually costs money to obtain approval. Also, in theory, it's easier to modify an Open Source driver, so it would no longer be compliant. In reality, 99.999% of users don't even know there is source code for the driver, so "Standards Compliance" may well be a smoke-screen for manfacturers who don't want to bother with non-WinTel systems. If certification was the only problem, manufacturers could release binary-only drivers.
   - (DW)The good news is that a certain amount of WinModem hardware is now supported. The bad news is that that is just the tip of the iceberg. Although the WinModems can now be used, they have functionality similar to that of a sound card - all the modulation and demodulation has to be performed by the host CPU. Work is progressing on this front too - see www.linmodems.org and Fabrice Bellard's Generic Linux V34 Soft Modem page for more up-to-date information.

Section 3 - Mailing list questions

(REG) The essential point to remember when posting to the linux-kernel mailing list is that there are a lot of very busy people reading the list. No matter how important you think you are, it is most likely that there are many people on the list who are more important than you. "Important" is not measured by the amount of money you have, how much your question is worth to your company or how desperate you are for an answer, rather, it is measured by how much you contribute to the linux kernel.

With that in mind, you should make sure that you are not wasting the time of other people on the list. *Write for maximum efficiency of reading.* It doesn't matter if it takes twice as long for you to compose a more readable message, if it halves the time a hundred key kernel developers spend trying to decode your message. Ignoring good taste and consideration is most likely to result in you being ignored.

1. **How do I subscribe to the linux-kernel mailing list?**
    - (ADB) Think again before you subscribe. Do you really want to get that much traffic in your mailbox? Are you so concerned about Linux kernel development that you will patch your kernel once a week, suffer through the oopses, bugs and the resulting time and energy losses? Are you ready to join the Order of the Great Penguin, and be called a "Linux geek" for the rest of your life? Maybe you're better off reading the weekly "Kernel Traffic" summary at http://kt.linuxcare.com/.
      OK, if you still want to read linux-kernel in it's full glory, send the line "subscribe linux-kernel your_email@your_ISP" in the body of the message to majordomo@vger.kernel.org (don't include the " characters, and of course replace the fake email address with your true address). **You have been warned!**
    - (David L. Parsley,RRR) It may be preferable to subscribe instead to 'linux-kernel-digest', which combines multiple postings into a large (~75k) digest form. In this way you will receive from 1 to maybe 3 or 4 mailings per day, instead of many hundreds. ;-)
      The main drawback of the digest version is that it is hard to reply to individual messages on the digest with standard mail clients. So its most useful to those who just wish to stay current with kernel development, but not active developers.
    - (Antonio Gallo) If you use procmail as your favourite MDA, you can use procmail to parse every mail inside the digest sent you by the linuk kernel digest. You just need to append the following lines to your .procmailrc configuration file. The mail will be expanded into the KERNEL mailbox or in any other mailbox you specify.

      ```
      :0:
      * ^X-Mailing-List: linux-kernel-digest@vger.kernel.org
      | formail +1 -ds >> /var/spool/mail/KERNEL
      ```

    - (MEA) Quite often I see things like what this summary report tells:

      ```
      FAILED:
        <smtp cedar-republic.com edmond@cedar-republic.com 60000>: ...\
              <<- RCPT To:<edmond@cedar-republic.com>
              ->> 550 <edmond@cedar-republic.com>... we do not relay
      ```

      Feeding this address to a page at URL: `http://www.zmailer.org/mxverify.html` yields information that ONE of their backup MX servers refuses to send email thru to them. Thus whenever all other servers fail to be reachable, that one ruins their email connectivity.

      Do make sure YOU don't have this very problem!

2. **How do I unsubscribe from the linux-kernel mailing list?**
   - (ADB) At the bottom of each and every message sent by the linux-kernel mailing list server one can read:
     -
     To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
     the body of a message to majordomo@vger.kernel.org
3. **Do I have to be subscribed to post to the list?**
   - (ADB) No, you don't have to be subscribed to the list to post to it. The address of the list is linux-kernel@vger.kernel.org. And you should indicate on your message that you wish to be personally CC'ed the answers/comments posted to the list in response to your posting.
   - (REG) It is, however, generally considered good netiquette to be subscribed to a list (or a newsgroup for that matter) and lurk for a while before posting. That way you can learn what's considered an appropriate post and what isn't.
     Don't treat the list as your personal helpdesk. Remember that the list is a community.
4. **Is there an archive for the list?**
   - (ADB) There are many, and at least this one has a search by word/subject capability. I keep an eye on the list by checking from time to time the archives on a Web server maintained by David Niemi. Here is another Web server that keeps a collection of Linux related list archives. You can also try here and here.
   - (REG) Here are some more resources:
     - "Kernel Traffic" at http://kt.linuxcare.com/ provides a weekly summary of the discussions on the list, and archives previous summaries.
     - The Kernel Journal is a date-sorted collection of announcements of kernel-related work. These announcements appear on the page almost as soon as they appear on the mailing list, and contain URL pointers to programs related to the kernel. This will keep you abreast of real work being done, rather than just a talkfest.
       NOTE that the Kernel Journal has been defunct for several months. I'm told that this is only temporary while the WWW pages are being re-organised.
5. **How can I search the archive for a specific question?**
   - (ADB) Use simple keywords which refer to the issue that matters to you. For example, if you are investigating an oops that happens whenever you plug in a network adapter NIC-007, use "NIC-007" or "oops NIC-007". As soon as you have found a link to a message that interests you, try to follow the thread. Remember that you will almost always get more information by carefully searching the archive than by posting a question to the list itself.
6. **Are there other ways to search the Web for information on a particular Linux kernel issue?**
   - (ADB) Sure. Before you check the list archives, you can search DejaNews and AltaVista (simultaneously, if your browser allows you to open various windows). You can also follow some links on the Linux Documentation Project site.
7. **How heavy is the traffic on the list?**
   - (REW) Very, very heavy: it's been over a 100 a day since mid 1998, and now closing on 200 a day. My log shows 2800 messages in 21 days [June 1998]. I now remember accidentally saving a bunch of messages in a different folder.
   - (ADB) You really don't want to read each and every posting to the list. If you are concerned with list traffic, I suggest you temporarily try the linux-kernel-digest list, which will be much easier on your mailbox (thanks to A. Wik for this suggestion).
   - (REG) There is a weekly summary called "Kernel Traffic" at http://kt.linuxcare.com/, which can save you a lot of time.
8. **What kind of question can I ask on the list?**
   - (ADB) The basic rule is to avoid asking questions that have been asked before, or that are irrelevant

to other list users, or that are off topic. Please use your good sense.
- (REG) Remember that this is a list for the discussion of **kernel development**. If you have some ideas or bug reports to contribute, this is the place. User space issues are not appropriate for this forum. If you find a bug in the C library or some application, it doesn't belong on linux-kernel.

9. **What posting style should I use for the list?**
   - (REG, contributed by thunder7@xs4all.nl) When following up a post on the kernel mailing list, please think before you quote. Since everybody else on the list also got the original post, don't quote it entirely. Highlight only the points that you really need to understand your arguments. Make sure the quoted part is recognizable as such, by ensuring each quoted line starts with a > (or more >>, in case of multi-level quoting). Don't quote signatures, entire patches, entire config files or entire posts. Don't quote the standard signature. The kernel-list is crowded enough already, let's take care!
   - (REG) Be aware that your message is far more likely to be deleted without being read if you have too much quoted material before your reply.
   - (REG) And please reply **after** the quoted text, not before it (as per RFC 1855). It's very confusing to see a reply before the quoted context. And it's embarrassing: it makes you look like a newbie. Change your mailer if necessary, if the one you have makes it hard to do reply-after-quoting.
     I know some people like to quote the entire message they are replying to, so they put their reply right at the top so people won't give up after the first page of quoted material. **Don't do it**. It's annoying. Just learn to stop quoting everything. No-one wants to see it all anyway (list archives allow people to see everything if they missed it). You're not helping yourself anyway, as you're more likely to be ignored if you reply-before-quoting.
   - (REG) Please don't use tabs or multiple spaces to quote text. Use the "`> `" sequence instead. Using whitespace to quote text makes it difficult to differentiate between what's quoted and the reply. And don't try to be cute or "different" and use some other character like "`}`" or whatever. Again, it's confusing. It wastes people's time. Write for maximum efficiency of reading.
   - (REG) Please try to have halfway reasonable spelling and grammar. When reading text with really bad spelling or grammar, people stall while trying to parse your post. Don't think you're being "artistic" by stripping out all punctuation characters. Linux-kernel is not an online gallery, it's a communications medium. Write for maximum efficiency of reading.
   - (PG) Don't attach huge files to your post. One major culprit is people attaching their kernel `.config` file to their post. These can be in excess of 1000 lines, and will grow more as kernel options are continuously added. If the contents of your .config file are relevant to your post then attach the output of

     ```
     grep ^C .config
     ```

     or

     ```
     grep "=[y|m]" .config
     ```

     .

10. **Is the list moderated?**
    - (ADB) No, the linux-kernel list is not moderated.
11. **Can I be ejected from the list?**
    - (ADB) It is technically possible, but I have never heard of anybody being ejected from the linux-kernel list.
    - (REW) **But** you will if you post questions or answers that are asked and answered on this FAQ. ;-)
12. **Are there any implicit rules on this list that I should be aware of?**
    - (ADB) Here are a few implicit rules which you should be aware of:

- Stick to the subject. This is a Linux kernel list, mainly for developers.
- Use English only!
- Don't post in HTML format! If you are using IE or Netscape, please turn off HTML formatting for your posts to the kernel list.
- If you will be asking a question, before you post to the list, try to find the answer in the available documentation or in the list archives. Just remember that 99% of the questions on this list have already been answered at least once. Usually the first answer is the most detailed, so the archives contain far better information than you will get from somebody who has answered the same question a dozen times or more.
- Be precise, clear and concise, whether asking a question or making a comment or announcing a bug, posting a patch or whatever. Post facts, avoid opinions.
- Be nice, there is no need to be rude. Avoid expressions that may be interpreted as aggressive towards other list participants, even if the subject being treated is particularly relevant to you and/or controversial.
- Don't drag on with controversies. Don't try to have the last word. You will eventually have the last word, but meanwhile you'll have lost all your sympathy credit.
- A line of code is worth a thousand words. If you think of a new feature, implement it first, then post to the list for comments.
- It's very easy to criticize someone else's code, but when you write something for the first time, it's not that simple. If you find a bug, a mistake, or something that could be perfected, don't immediately post a comment such as "This piece of code is crap, how did it get into the kernel?". Contact the author of the code, explain the issue, and try to get the point across in a simple, humble way. Do that a few times and you will get a lot of credit as a good code debugger. Then when you write a piece of code people will pay attention to you.
- Don't flame beginners that ask the wrong questions. This adds noise to the list. Send them a private mail pointing them to a source of information e.g. this FAQ.

13. **Does the list get spammed?**
    - (ADB) The linux-kernel list is no longer spammed, you will rarely if ever find a commercial posting to the list itself. OTOH once you post to the list, expect to get a few undesirable mails in the following days. Unfortunately some people watch the list and think it's a good idea to pick names from it. There are many ways to avoid spam, check the dedicated anti spam sites on the list. I learned many things this way.
    - (REW) Although the list maintainers do their best to keep the list spam free, it is not possible to do this 100%. Some of the good kernel development people cannot keep up with the volume on linux-kernel. But they do occasionally post. Therefore we need to keep the submissions open for "everybody". Some of the other important people have two or three Email addresses. They too need to post from different addresses. Consequently something that looks like a submission from a valid return address tends to go on the list. There is nothing an automated filtering system can do about it. The end result is about one spam a month. It happens. The maintainer will get a flood of mail about it and he will block the domain it came from. Please don't bother the list about it, don't add noise. Don't post "This guy is a jerk if he spams this list". Don't post "I traced him, you can mail bomb him at this address". Don't post "I traced him, bother his postmaster at such and such".
14. **I am not getting any mail anymore from the list! Is it down or what?**
    - (ADB) Majordomo is an intelligent mail list server. If for any reason your email address is unavailable, after some retries you will be automatically unsubscribed.
    - (REW) On the other hand, accidents with the mailing list server have happened. These have wiped out the whole subscription list once or twice. Just resubscribe. Majordomo will get you a nice note saying you're still subscribed if suddenly everybody went dumb. Don't post "Just testing: Is the list working? I didn't get any mail for a few days now".

- (MEA) You may get unsubscribed because MTAs relaying traffic to you get bounces for some reason. One thing to verify is that your email routing data in the DNS is valid, e.g. feed your address to the input box at: http://www.zmailer.org/mxverify.html
- (MEA) VGER and/or one of its fanout boxes may be in overload. Usually system keepers notice the situation, and it becomes fixed within 1-2 days without messages being lost, but we don't track the entire world. Asking help from postmaster@vger.kernel.org could expedite the issue. Asking help on lists WILL NOT help, doing so just puts more load on the system!

15. **Is there an NNTP gateway somewhere for the mailing list?**
    - (RRR) Yes there is the newsgroup fa.linux.kernel, but you can only read the mailinglist there, not post directly. Posting to the list must go by email to linux-kernel@vger.kernel.org.
      Here's the dejanews URL, if your NNTP host don't have the group
      http://www.dejanews.com/bg.xp?level=fa.linux.kernel
16. **I want to post a Great Idea (tm) to the list. What should I do?**
    - (REG) OK, that's great. Now:
      - First make sure that your idea is relevant to kernel development. Perhaps your idea is better implemented in the C library, or perhaps in a new library? Before posting to linux-kernel, be sure it really **is** a kernel issue.
      - OK, so you have this great idea for the kernel. Are you sure someone hasn't thought of it before? Reading all of this document is a good starting point. Also search the mailing list archives to see if that topic has been raised before.
      - Now you have verified that you have an idea none has suggested before. For the best response, code up an implementation/kernel patch and post that to the kernel list when you announce your idea. If you provide code, you can be sure someone will try it out and give you comments. If you don't know anything about kernel hacking, this is a good time to start learning:-) By the time you've implemented your idea, you'll be able to call yourself a Linux Guru.
      - If you really can't code something up, but still would like your idea implemented, post a message to the kernel list. Be as clear and precise as possible, so that people can understand your idea quickly. If you are lucky, someone who likes your idea may find the time to implement it. If nobody steps forward to implement it, you're out of luck: remember, we're all volunteers and we all have too many things to do as it is.
      - If you get a negative response to your idea, don't get offended, after all, we all have different notions on what is a Good Idea (tm) and a Bad Idea (tm). If someone is rude to you, please resist the temptation to carry on a war on the list. Instead, email them privately saying that you don't like their rudeness. If everybody is polite, but just strongly disagrees with your idea, be careful not to push it too hard. If people haven't understood the point you are making, try explaining it a different way. But if people understand your idea but maintain it is flawed, it's time to stop pushing it. Pushing harder will just get you ignored.
      - If you're convinced you're right, despite what everybody else says, stop talking about it and implement it! If you're right, you'll have the last laugh.
    - (ADB) Good code (i.e. documented, elegant, efficient) and some benchmarking data showing your Great Idea performs well will go a long way to show you're right.
17. **There is a long thread going on about something completely offtopic, unrelated to the kernel, and even some people who are in the "Who's who" section of this FAQ are mingling in it. What should I do to fight this "noise"?**
    - (REW, ADB) Ignore it.
    - (REG) Don't send a response to the kernel list under any circumstances. If you feel compelled to respond, do so privately informing the person that the message was offtopic. Or set up a procmail recipe to drop all messages for that thread: that way you'll never see the thread again.

18. **Can we have the Subject: line modified to help mail filters?**
    - (REG) The usual proposition is that a string like **[LINUX-KERNEL]** is prepended to the subject line.
      This question has been raised many times before, and the answer has always been "no" or "there are better ways to filter email". The majority of the developers, and all (?) of the list maintainers take this position. Some of the reasons are:
        - It would increase the size of the Subject: line. This is a problem, as it limits the amount of useful information that can be seen in the Subject: line, making it harder to scan through a list of subject lines looking for interesting subjects.
        - It doesn't work for cross-posted messages, as the subject line for a single email will change depending on which list it was sent via. Not only can this confuse simple-minded filtering recipes, it can also break threaded mail readers (people may end up reading the same message twice).
        - The correct way to filter is to base your recipe on the **X-Mailing-List:** line, which should always have "linux-kernel@vger.kernel.org".

      An example procmail recipe would look like this:

      ```
      # Linux-kernel list
      :0: /var/lib/emacs/lock/!home!fred!mfilter!linux!kernel
      * ^X-Mailing-List: linux-kernel@vger\.kernel\.org
      /home/fred/mfilter/linux/kernel
      ```

      People using mailagent might try this in their .rules file (thanks to Martin Smith <martin@sharrow.demon.co.uk>):

      ```
      To CC: /linux-kernel-digest@vger.kernel.org/
      { SPLIT -adi ~/Kernel }
      ```

      Similarly to procmail you can omit the mail folder from the split command. This causes the split messages to go back into the mailagent queue for further processing.

      Most mailers with filtering capabilities can be similarly configured. If not, then you can simply install procmail. If perchance you're running a damaged OS that can't filter properly, and there is no procmail port for it, then you should either upgrade, or accept that you won't be able to filter linux-kernel. Don't bother asking for a subject line modification.
19. **Can we have a Reply-To: header automatically added to the list traffic?**
    - (DW) Some mailing lists automatically add a **Reply-To:** header to the mails which go through them, forcing people to reply to the list, rather than replying personally to the original poster. This is a bad idea for many reasons which won't be listed here. See Chip Rosenthal's excellent summary **Reply-To:** Munging Considered Harmful for more explanation.
20. **Can I post job offers/requests to the list?**
    - (REG) Of course not! This is a technical development list, not a job exchange. If you want to join a job exchange list, send `subscribe linuxjobs` in the *body* of the message to majordomo@eax.com by executing the following command:
      `echo "subscribe linuxjobs" | mail majordomo@eax.com`
      To send messages, email to linuxjobs@eax.com

Section 4 - "How do I" questions

1. **How do I post a patch?**
    - (ADB) I assume you made the patch following the general instructions found above. Now write a short post describing your patch, the version of the kernel it applies to, your tests, the feedback you would like to get, etc. This should fit in 10 lines. Attach your patch and a one line README file describing it very succinctly, and mentioning your name and email (either as two ASCII files or as a MIME encoded tarball). In the subject of your post, put: [PATCH] <the driver name or piece of code patched>, kernel <kernel version>. Send. Wait.
      The small README file insures that your patch will not start circulating around the net without people noticing your name. If you don't care about copyright and/or your patch is trivial, you can skip tarring the files, just gzip the patch file and attach it to your post.
    - (PG) If your patch is on the large size (say larger than 500 lines) consider posting a URL pointing to the patch along with the patch description, instead of the whole patch. If you don't have a WWW site handy to put the patch on, then at least gzip it prior to attaching it to your post/patch description.
    - (REG) Note that Linus does not read linux-kernel very much. So if you want him to see a patch, you will need to send it to him directly (say by Cc:ing him if you post to the list). Note that Linus likes to be able to read patches in plain ASCII, so anything that is uuencoded or MIMEd is likely to go straight to the bit-bucket. If because your patch is large you only send a URL, send a plain-text copy to Linus privately.
      Also note that Linus drops patches silently when he is too busy (which is always:-), so if you don't see it in the next kernel patch, send it again. Oh, and don't expect him to tell you he's applied the patch, either.
2. **How do I capture an Oops?**
    - (REG, quoting Keith Owens) If an Oops is recoverable then the text appears first in the kernel message buffer (/proc/kmsg). You can use the dmesg command to print the contents but most of the time klogd and syslogd will automatically capture the Oops and write it to your log files.

      Sometimes an Oops is so bad that the kernel is completely hung. When this occurs, almost anything that requires kernel support is also dead. In particular most interrupt driven subsystems are unusable, especially after the dreaded "`Aiee, killing interrupt handler`" message. Since most disk controllers use interrupts, no disk I/O is possible so the Oops does not get written to the log files. The same problem applies to logging over the network, most network cards require interrupt handlers.

      In a complete hang, you have three options.
        - Write the Oops down by hand from the screen and type it in after you have rebooted. This is the only option if you have not planned for a kernel hang.
        - If you plan ahead and install a serial console linked to another machine (read `linux/Documentation/serial-console.txt`) then you can capture the Oops report on the other machine. By far the easiest and most reliable option.
        - Since kernel 2.3.10 it has also been possible to use a parallel port line printer as a console. You can either attach a real printer, or another computer with EPP (Enhanced Parallel Port) support, which pretends to be a printer.
        - There have been patches on linux-kernel to save the log somewhere in hardware. Unfortunately these patches are very hardware specific. Search the l-k archives for "Oops assist", "OOPS output over reboot" and "KMSGDUMP". Most of these patches require that the keyboard still works and even that can be useless when the kernel hangs.

      Other operating systems can save the log even when the machine hangs, why doesn't Linux? Any OS that can save the log after a catastrophic kernel failure must do so without kernel support, that typically means using the underlying hardware. Alas the ix86 hardware does not provide enough support for this, in particular most BIOS will clear memory on reset, destroying any data in storage.

3. **How do I post an Oops?**
   - (ADB) Assuming you have found a genuine Oops (those are rare nowadays, but they happen), you should post the relevant portions of your system log, kernel configuration file and kernel symbol map, plus a description of your hardware and the circumstances under which the Oops occurred. Can the Oops be triggered by any particular method? Did it happen after you changed any part of your hardware configuration?
     Don't post your oops report before you have checked linux/Documentation/oops-tracing.txt, the relevant paragraphs in linux/README, the ksymoops C program in linux/scripts/ksymoops which has another README, and the gdb man and info pages (thanks to Paul Kimoto for this tip). These documents describe the basic procedure for kernel oops tracing. Good trace info makes it much easier to understand and solve apparently weird oopses.
   - (REG) **Don't** even bother posting an Oops if you haven't run it through `ksymoops` to decode the symbol addresses. The report will be ignored because it contains too little useful information. Make sure you copy the correct `System.map` file into `/boot` or into the modules directory, otherwise you will get incorrect results.
   - (REG, quoting "The Doctor What") There are some situations that make a kernel oops useless. The two most obvious are if your are overclocking your CPU or running VMWARE's vmmon. The reason is that overclocking can introduce random bit errors, while VMWARE's vmmon has the ability to (and does) change parts of the kernel. In both cases, data in the kernel, as reported by the oops, won't be useful.
4. **I think I found a bug, how do I report it?**
   - (ADB) A bug differs very slightly from an oops, actually. An oops is when the kernel detects that something has gone wrong. A bug is when something (in the kernel, presumably) doesn't behave the way it should, either with a driver or in some kernel algorithm. If you can detect this misbehaviour, you may or may not be getting an oops.
     Perhaps the most important step is to determine under which conditions this misbehaviour can be triggered, and whether it is reproducible.
5. **What information should go in a bug report?**
   - (ADB) Does it affect system security? Is it related to a specific driver/hardware configuration? Did you manage to identify the piece(s) of kernel code concerned? It really depends on the kind of bug you found.
     In fact Frowahlt Egerer has proposed a standard Linux kernel bug report form (check the previous FAQ, appendix B) that is as good as one can get (but unfortunately it's rarely if ever used).
   - (TYT) Please follow general good bug reporting guidelines: remember, the developers don't have access to your system, and they're not mind readers. Tell us which kernel version, and what your hardware is (if you're not sure, more detail is better than less). At the very least, tell us what processor and motherboard you have, how much memory, how many and what kind of disks (IDE, SCSI, etc.), what kind of disk controllers you have, what other expansion boards (specify whether they're PCI or ISA or some other bus). Also useful: what version of gcc and binutils were used to compile the kernel.
     Try to find a simple, reliable way to trigger the problem. Telling the developer that they have to set up some complicated application environment (especially if it involves some ghastly expensive proprietary software like SAP or Oracle :-) may cause the developer to hit the 'd' key and move on.
     In general, raw data is better than jumping to conclusions. If you want to give your guesses in your bug reports, they're of course welcome, but this is *not* a substitute for raw data. Many problems are not what they first seem. A hardware problem can masquerade as a VM problem. A device driver or VM problem can cause the filesystem code to notice a discrepancy, and flag a warning. Even if you're *sure* that the problem isn't a hardware problem, or by some other theory that the developer

advances, the scientific method demands that you do a test to rule these sorts of things out. Sometimes, you will get surprised.....
If you get a kernel oops message, it's useless unless you give us the proper symbolic information. This used to mean sending relevant pieces out of System.map. Fortunately, with the latest syslogd/klogd, this is much simpler (check the man page of klogd to see if your version has this feature; if it doesn't, you should upgrade to the latest version, and probably to a modern distribution). Make sure that you have the System.map file installed the appropriate place so that klogd can find it (the standard search path is in the /boot, /, and /usr/src/linux directories).
If the system oops and then dies without a chance for klogd to record the information into a syslog file, copy down the oops message exactly, and then use the ksymoops (see the man page) to get the symbolic information out. Remember, the raw numbers by themselves will generally not be useful.
If you can, try to isolate the problem to a specific kernel version. Knowledge that it worked in version 2.2.17, as well as 2.3.0-test6, but it stopped working in 2.3.0-test7-pre1, is extremely helpful, and will save developers a lot of time. (If you're comfortable disecting patches, fell free, taking apart the individual file changes and try to isolate to a particular change.)
    - (REG) You did of course read `REPORTING-BUGS` from the kernel source tree first, didn't you?

6. **I found a bug in an "old" version of the kernel, should I report it?**
    - (CP) Only if it hasn't been fixed yet. The best thing to do is to try to repeat it with a new version of the kernel. If not, you have to figure out if it's been fixed yet. The kernel release announcements and patch descriptions from Jitterbug are also useful. Failing that, look for discussion of the bug in linux-kernel and check the patches between your kernel and the latest ones for relevant changes.
    If you can't find your bug mentioned, and you're not running a truly ancient kernel, posting a bug report is worthwhile. You can probably expect a request of the form "try it with the latest kernel" or "try it with this patch" in response. If there's a reason why you can't run the latest kernel (like it's your main dialin server and you don't want to mess with it), saying it in your original report will save some explaining later.

Section 5 - "Who's who" questions

1. **Who is in charge here?**
    - (ADB) Do you mean "Who takes decisions relative to the mailing list?" or do you mean "Who takes decisions relative to the Linux kernel"? If the former: there is relatively little to decide when it comes to the mailing list. Majordomo, once correctly setup, will manage the list in an autonomous fashion. In any case, you can always reach the Majordomo-owner for the list, if you have a very specific question about the list mechanism itself. When it comes to kernel development management and decision making, see the answer to Question 7.8 below.
2. Who is H. Peter Anvin?
3. Who is Donald Becker?
4. **Who is Alan Cox?**
    - (AC) Alan Cox supervises the 2.0.34/35/36 kernel releases, works on the Mac68K port, the SGI port, 2.0 networking, modular sound, video capture and helps collect up and sort patches to the kernel. He gets to do all this and sleep because the nice guys at Red Hat pay him to hack Linux.
5. **Who is Richard E. Gooch?**
    - (REG himself) "I've written various utilities and kernel patches which you can find here including the MTRR, devfs and fastpoll patches. My PhD in Computer Science was on the topic of Astronomical Visualization , which is my current research interest. This is what I work on when I don't get distracted by kernel hacking. See my home page to find out more about me."
6. **Who is Paul Gortmaker?**

- (ADB, OK'ed by Paul) Paul has contributed various pieces of kernel code over the last few years, among other things the Real Time Clock driver. He is also the maintainer of the 8390 based network drivers (NE-2000, etc.), and wrote the Linux Ethernet HOWTO and the Boot-Prompt HOWTO.

7. Who is Mark Lord?
8. Who is Larry McVoy?
9. **Who is David S. Miller?**
    - (DSM) David Miller is mainly known for the porting work he has done, primarily for the 32-bit and 64-bit Sparc platforms although he has made significant contributions to the MIPS effort as well. He is also the current maintainer of the IP networking layer in the kernel and likes to address general performance and scalability problems all over as his time permits.
10. Who is Linus Torvalds?
11. **Who is Theodore Y. T'so?**
    - (TYTSO) Theodore Ts'o has over the years written, rewritten, or supported Posix Job Control, the high level tty driver, the serial driver, the ramdisk support, e2fsck/e2fsprogs, and other bits and pieces of code in and near the kernel. He is currently a member of the Technical Board of Linux International. His day job at MIT is concerned with Kerberos and other network security and I/T architecture issues. He is also a member of the Internet Engineering Task Force, where he serves as a member of the Security Area Directorate.
12. **Who is Roger Wolff?**
    - (REW himself) "I wrote the kmalloc that still drives linux-2.0.x. I wrote the Specialix and Olicom device drivers. I currently write Linux device drivers for a living. Contact me if you need one."
13. **Why don't we have a Linux Kernel Team page, same as there are for other projects?**
    - (ADB) Perhaps because there is no Linux Kernel Team, per se. Also because so many people contributed to the Linux kernel that it would be a tough task to setup and maintain such a page. Finally, although this is not a rule, most Linux kernel contributors prefer to keep a low profile, for various reasons.
14. **Why doesn't <any of the above> answer my mails? Isn't that rude?**
    - (ADB) Probably because of sheer lack of time to answer each email that gets sent to them. What would you do if you got 1000 mails in your mailbox, from one day to the next? They don't mean to be rude, however.
      One hint: if you attach to your mail a genuinely useful piece of good quality code that you wrote, there are good chances that it will be answered (choose a good subject line, too). If you ask a dozen beginner's questions, the truth is, there are zero chances that you will get even the simplest reply pointing to some source of information.
      Aside from that, you may get "mail rejected" error messages if you try to contact some major contributors of the list. It is due to the spam filtering systems used by them. Please complain about it to your ISP and don't post to the list about spam !! .
    - (REG) Some people also have very aggressive mail filtering which rejects (non-list) messages from people they don't know, asking for a re-send with a password (this stops SPAM dead). If you mail to someone and receive such an automatic response, don't get upset. Remember, a person's mailbox is their personal property.
      Also, some people maintain "guru lists" and only read posts on linux-kernel by someone on their guru list, other people's posts go to `/dev/null`. This is done because there are too many questions asked on linux-kernel which shouldn't be (which is why people should read this FAQ first!), and people can't cope with the load. If you post to the list and want make sure a specific individual will see the message, `Cc:` that person.

Other OS developers

Rogier Wolff (REW) suggested we add a section on OS developers who influenced/preceded the design of Linux.

- **Who is Prof. Douglas Comer?**
  - (Prof. Comer) Dr. Douglas Comer is a full professor of Computer Science at Purdue University, where he teaches courses on operating systems and computer networks. He has written numerous research papers and textbooks, and currently heads several networking research projects.
    He has been involved in TCP/IP and internetworking since the late 1970s, and is an internationally recognized authority. He designed and implemented X25NET and Cypress networks, and the Xinu operating system. He is director of the Internetworking Research Group at Purdue, editor of Software - Practice and Experience, and a former member of the Internet Architecture Board.
    Dr. Comer completed the original version of Xinu (and wrote "The Xinu approach" book) in 1979. Since then, Xinu has been expanded and ported to a wide variety of platforms, including: IBM PC, Macintosh, Digital Equipment Corporation VAX and DECStation 3100, Sun Microsystems Sun 2, Sun 3 and Sparcstations, and Intel Pentium. It has been used as the basis for many research projects. Furthermore, Xinu has been used as an embedded system in products by companies such as Motorola, Mitsubishi, Hewlett-Packard, and Lexmark. There is a full TCP/IP stack, and even the original version of Xinu (for the PDP-11) supported arbitrary processes and network I/O.
- **Who is Richard M. Stallman?**
  - (RMS) Richard Stallman is the founder of the GNU project, launched in 1984 to develop the free operating system GNU (an acronym for "GNU's Not Unix"), and thereby give computer users the freedom that most of them have lost. GNU is free software: everyone is free to copy it and redistribute it, as well as to make changes either large or small.
    Today, Linux-based variants of the GNU system, based on the kernel Linux developed by Linus Torvalds, are in widespread use. There are estimated to be over 10 million users of GNU/Linux systems today.
    Richard Stallman is the principal author of the GNU C Compiler, a portable optimizing compiler which was designed to support diverse architectures and multiple languages. The compiler now supports over 30 different architectures and 7 programming languages.
    Stallman also wrote the GNU symbolic debugger (GDB), GNU Emacs, and various other GNU programs.
    Stallman received the Grace Hopper Award from the Association for Computing Machinery for 1991 for his development of the first Emacs editor in the 1970s. In 1990 he was awarded a MacArthur Foundation fellowship, and in 1996 an honorary doctorate from the Royal Institute of Technology in Sweden. In 1998 he received the Electronic Frontier Foundation's Pioneer award along with Linus Torvalds.
- **Who is Prof. Andrew Tanenbaum?**
  - (Prof. Tanenbaum) Andrew S. Tanenbaum has an S.B. degree from MIT and a Ph.D. from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands, where he heads the Computer Systems Group.
    His current research focuses primarily on the design of wide-area distributed systems that scale to millions of users. These research projects have led to over 70 refereed papers in journals and conference proceedings. He is also the author of five books.
    Prof. Tanenbaum has also produced a considerable volume of software. He was the principal architect of the Amsterdam Compiler Kit, a widely-used toolkit for writing portable compilers, and MINIX, a small UNIX-like operating system for operating systems courses.
    Prof. Tanenbaum is a Fellow of the ACM, a Senior Member of the IEEE, a member of the Royal Netherlands Academy of Arts and Sciences, and winner of the ACM Karl V. Karlstrom Outstanding Educator Award.

Section 6 - CPU questions

1. **What is the "best" CPU for GNU/Linux?**
    - (REW) There is no "best" CPU. The choice of CPU always depends on your price/performance/technical requirements. On the x86 side, we have Intel, AMD, Cyrix and IDT/Centaur, with various models available. All of these work.
      Besides the x86 processors, the Linux kernel runs on 68k processors, MIPS R3000 and R4000, Power PC, ARM, Alpha and Sparc processors. There are lots of different ways to build a computer around a processor. If you have an x86, they built a PC around it. Don't go around buying second hand R4000 computers because the Linux kernel runs on the R4000 processor. Check the latest Linux kernel revision to see if the specific computer you're buying is supported.
    - (ADB) OK, the Linux kernel is a good start. Now, there is a huge difference between kernel support and a ready-to-install distribution. Only three architectures have widely available, reasonably homogeneous distributions: x86 (or i386), Alpha and Sparc. And the Alpha and Sparc distributions that exist still have some rough edges. IOW, if you don't want to spend a lot of time installing and fine-tuning GNU/Linux, and you have a limited budget, your "best" choice is an x86 machine. If you have very specific needs (e.g. a hand-held computer running Linux, where the low power ARM architecture would be the ideal choice, or a workstation dedicated to scientific applications, where an Alpha or a Sparc would provide superior performance), check the various architectures, list your specific requirements, and make a choice. Nowadays Alpha 21164 machines are much more affordable than one or two years ago, but it's certainly harder to put one together than your average PC clone.
2. **What is the fastest CPU for GNU/Linux?**
    - (REW, ADB) The CPU field is very active in terms of technological developments. New CPU models, new architectures, new manufacturing technologies keep pushing the state of the art. WRT GNU/Linux, it is a general consensus that Alpha machines usually provide the best floating point performance, when the actually shipping hardware available at any given point in time is compared (June 1998: the 21164/600).
      However for non floating point applications the issue is not as clear-cut. Very high clock rate x86 machines (e.g. Pentium-II/400) provide impressive integer performance, for use in e.g. databases or Web server applications.
      For 3D rendering applications you may want to consider the GNU/GPL Mesa OpenGL compatible library, which has support for some graphics accelerator chips.
      Also note that some applications are not CPU bound. Check the exact bottleneck in your case.
3. **I want to implement the Linux kernel for CPU Hyper123, how do I get started?**
    - (ADB) Is Hyper123 supported by gcc, or at least is the Hyper architecture supported by gcc? Do you have a target machine with a well defined architecture? If you have answered yes to both questions proceed to REW's answer. If you have answered no to either or both, don't even bother getting started. This is a major project, not exactly the kind of thing you do over the weekend.
      Quoting from a SparcLinux paper by Miguel de Icaza:
      "*Thanks to having an international team of developers and support people, when the first Linux/SPARC distribution on CD went out we had a very strong port: a port that had taken only 22 months to engineer and complete (starting from scratch up to releasing the operating system on a bootable CD-ROM).*"
    - (REW) Auch. Difficult task. Besides having to write support for the processor, you will also have to write the boot sequence to get things going. And a few device drivers.
      You're not running away screaming yet? Good. Make sure you get the programmers manual for

Hyper123, and data sheets for all the peripheral IC's. Make sure you have the docs for the computer that you're working on (addresses, registers for the stuff on the motherboard).
After that, start on learning the processor, by writing the boot program. Try booting a simple program that says "hello world". That will also allow you to write a console device driver.
Next, there is the hard part: get Linux to compile and run on the processor. Make a new arch directory and start putting things in there that implement whatever needs implementing on your processor.

4. **Why is my Cyrix 6x86/L/MX/MII detected by the kernel as a Cx486?**
   - (RRR, ADB) Cyrix 6x86 CPUs are different in many ways from Pentium (tm) and AMD K5/K6 (tm) CPU's, so special code must be included for adequate CPU detection, setup and reporting. Cyrix 6x86 support isn't perfect in kernels 2.0.x up to 2.0.34. From 2.0.35 on things should get much better ('cause we're working on it ;)). Similarly, late 2.1.1xx kernels should fully support the Cyrix CPU's. Please check the Linux Cyrix 6x86 HOWTO site for details and patches.
5. **What about those x86 CPU bugs I read about?**
   - (ADB) There are basically three known bugs that affect x86 processors, and each CPU design got its fair share it seems:
     - ***The Intel Pentium F00F "Death" bug***, affects ALL Pentium and Pentium MMX CPU's. Linus implemented the Intel recommended workaround for this bug a few days after the bug was first reported in the newsgroups. All recent kernels will report and workaround the bug.
     - ***The AMD K6 "sig11" bug***, affects only a few K6 revisions. Was diagnosed by Benoit Poulot-Cazajous. There is no workaround, but you can get your processor exchanged by contacting AMD. 2.2.x kernels will detect buggy K6 processors and report the problem in the kernel boot message. Recently, a new K6 bug has been reported on the linux-kernel list. Benoit is checking into it.
     - ***The Cyrix 6x86(Classic, L, MX) "Coma" bug***, affects ALL Cyrix 6x86 CPU's. I proposed a simple workaround which is implemented as a user space boot option, a few hours after the bug was reported on the linux-kernel mailing list. See the Linux Cyrix 6x86 HOWTO site for details. Cyrix was notified of the bug, and their new MII CPU's are not affected by this problem anymore.
6. **I grabbed the standard kernel tarball from ftp.kernel.org or some mirror of it, and it doesn't compile on the Sparc, what gives?**
   - (DSM) Often the Sparc port diverges due to the sheer high rate of changes which occur to that port. Also changes can happen to major interfaces in the kernel and the Sparc port is not updated at the same time. Eventually the Sparc port maintainers do try to merge all of their work into the standard tree, and at which time it will compile. In any event, trees which will compile just fine are available via two mechanisms, the vger CVS tree (accessible via read-only anonymous CVS) and pre made tarballs of known working stable or test kernel trees. Check:
     - ftp://vger.kernel.org/pub/linux/README.CVS and
     - ftp://vger.kernel.org/pub/linux/Sparc/kernel/v2.{0,1}/
7. **Does the Linux kernel execute the Halt instruction to power down the CPU?**
   - (REG, ADB) Yes. The Linux kernel will execute the Halt instruction when the machine is idle (check the code for the idle_task in sched.c). It has done so since the earliest i386 implementation, even though on the i386 we didn't care about power saving; it's just that halting the CPU is the Right Thing (tm) to do when there is no other task that must be run.
     On the Pentium, K6 and C6 CPUs, power consumption gets automatically reduced from an average 12-24 Watts operating power down to 2-3 Watts when the processor is Halted. On the Cyrix 6x86 CPUs, Halt state power consumption can be further reduced down to 150 mw by enabling the Suspend-on-Halt feature.
     Reduced power consumption means cooler, more reliable machine operation and longer component

life. And it saves trees too.

8. **I have a non-Intel x86 CPU. What is the [best|correct] kernel config option for my CPU?**
   - (ADB) For 386 class machines, compile as a 386. For 486-class machines, compile as a 486.
     For the Cyrix 6x86 family CPUs and the AMD K5 and K6, you should probably compile the kernel as a Pentium or PPro. The only difference between the Pentium (-M586) and PPro (-M686) compile options is in the string operations (AFAIK). The Pentium option uses a header file that breaks down the complex string opcodes into simpler operations (which are faster on the Intel Pentium and Pentium MMX).
     The PPro option uses the complex opcodes, but should be slightly faster than a Pentium because of the PPro has deeper, smarter pipelines.
     The same rules apply to the 6x86 family and the K5/K6, but the difference in speed is minimal between the Pentium and PPro kernel config options on these CPUs (PPro should be slightly better).

     The 486 kernel config option (-M486) should **not** be used for anything above a 486-class CPU. This option sets code alignment options that work well on the 486, but that cause excessive NOP padding on 586 and above class machines. Usually, the 6x86 speculative execution capabilities will just optimize this padding at run time, but the NOP opcodes still take precious L1/L2 cache space (same applies to the K6; I am not 100% sure of what the K5 does).
     The 386 config option (-M386) does not suffer from excessive padding, but does not produce code optimized for recent x86 CPUs either, so it is also deprecated, except for kernels included in GNU/Linux distributions which must run on the widest possible range of machines.

Section 7 - OS questions

1. **OS $toomuch has this Nice feature, so it must be better than GNU/Linux.**
   - (ADB) Sorry, but this simply means that OS $toomuch was designed with a given set of objectives and priorities, and GNU/Linux was designed with a different one. Neither is better than the other and also note that I am not referring to the respective implementations. But please, no OS comparisons on the linux-kernel list. Check the newsgroups instead, particularly comp.os.linux.advocacy which is dedicated to that kind of debate.
2. **Why doesn't the Linux kernel have a graphical boot screen like $toomuch OS?**
   - (ADB) Because it doesn't need one. You can add that feature to the boot loader code, if you want to. The Linux kernel has no graphics primitives, just like any UNIX kernel.
3. **The kernel in OS CTE-variant has this Nice-very-nice feature, can I port it to the Linux kernel?**
   - (ADB) Sure, you can do (almost) anything you want with Free Software. Oh, OS CTE-variant is not Free Software?
4. **How about adding feature Nice-also-very-nice to the Linux kernel?**
   - (ADB) You should probably read the definition of creeping featurism first. Related concepts, in increasing order of obfuscation: the KISS rule-of-thumb, the "Small is Beautiful" concept, Occam's Razor and Complexity Theory. A good book to read on these concepts as they apply to OS design is "The Mythical Man-Month" by Frederick P. Brooks, Jr.
     You may also want to check the Linux Kernel Wish List for 2.1/2.2 kernels.
5. **Are there more bugs in later versions of the Linux kernel, compared to earlier versions?**
   - (ADB) There are no more known bugs in later kernel versions than in earlier kernel versions. However, the Linux kernel source code has been growing at a constant rate. As a general rule, large pieces of code tend to have undetected bugs. OTOH, the core code for the Linux kernel seems to have stabilized at around 16 thousand lines of C code, according to Larry McVoy.
   - (REW) I'd say more than 23 thousand lines in 2.1.x. Add together the totals from kernel, mm,

arch/<somearch>/, subtract fpu-emulation.

6. **Why does the Linux kernel source code keep getting larger and larger?**
    - (ADB) There are four causes for this unbounded growth:
        1. **New architectures are implemented.** This is usually OK, because the code that is specific to each architecture is (in theory, at least) separate from the others. Common code doesn't grow.
        2. **New drivers are implemented.** Again, this is OK, because each driver has different source files, and those are selectively compiled in the kernel executable or built as modules according to the specified kernel configuration.
        3. **Old code gets adequately documented.** Adding comments and documentation increases the size of the source, but it's still a Good Idea (tm).
        4. Creeping featurism. It's generally considered a Bad Idea (tm) to keep adding more features to an already working piece of code.
7. **The kernel source is HUUUUGE and takes too long to download. Couldn't it be split in various tarballs?**
    - (REG) The kernel (as of 2.1.110) has about 1.5 million lines of code in *.c, *.h and *.S files. Of those, about 253 k lines (17%) are in the architecture-specific subdirectories, and about 815 k lines (54%) are in platform-independent drivers. If, like most people, you are only interested in i386, you could save about 230 k lines by removing the other architecture-specific trees. That is a 15% saving, which is not that much, really. The "core" kernel and filesystems take up about 433 k lines, or around 29%.
      If you want to start pruning drivers away, the problem becomes much harder, since most of that code is architecture independent. Or at least, is supposed to be/will be. There is some driver code which probably should be moved to an i386-specific subdirectory, and perhaps over time it will be (it will take a lot of work!), but you need to be careful. PCI cards for example should be architecture independent. Throwing out the non i386-specific drivers will save around 97 k lines, a saving of about 6%.
      But the most important argument for/against splitting the kernel sources is not about how much space/download time you could save. It's about the work involved for Linus or whoever will be putting together the kernel releases. Building tarballs (compressed tarfiles) of the whole kernel already represents a considerable amount of work; splitting it into various architecture-dependent tarballs would dramatically increase this effort and would probably pose serious maintainability problems too.
      If you are really desperate for a reduced kernel, set up some automated procedure yourself, which takes the patches which are made available, applies them to a base tree and then tars up the tree into multiple components. Once you've done all this, make it available to the world as a public service. There will be others who will appreciate your efforts.
      Under no circumstances should you complain to the kernel list. I promise you that Linus and the core developers will completely ignore such messages, so whinging about it is a complete waste of bandwidth. The only message on this subject that should be posted is an announcement of a new service providing split kernel sources.
8. **What are the licensing/copying terms on the Linux kernel?**
    - (RRR) In the root directory of the Linux kernel source tree (e.g. /usr/src/linux/), you will find a file COPYING. The file states that the Linux kernel is placed under the GNU General Public License (version 2), a copy of which is provided. If still in doubt, post to the appropriate forums (such as gnu.misc.discuss) or ask a lawyer, but **don't** ask about it on the linux-kernel list.
9. **What are those references to "bazaar" and "cathedral"?**
    - (ADB) These terms are used to describe two different development models adopted by the Free Software community, and were first coined by Eric S. Raymond. You should check his original

article.
Note that Eric's article describes two among an infinite range of possible different development models. You could for example create new "Versailles", "Great Wall of China" or "Pyramid of Kheops" software development models. As long as the end result is under a GNU/GPL license, it will still be Free Software.

10. **What is this "World Domination" thing?**
    - (ADB) Geek humor? Please don't take this seriously! This is just a way of saying that there are more and more people using GNU/Linux all over the world i.e. that the Free Software movement is gaining momentum. Note that the "Free" in Free Software refers to **freedom**, just about the opposite of what's implied by "World Domination".
    - (REW) This is a reference to an interview of Linus some years ago. After being pretty modest about the success that Linux was enjoying he concluded the interview with the remark: "*Of course, what I really want is total world domination.*"
      I've been browsing the net for the reference for this.
      http://www.ukuug.org/newsletter/news@uk63-5.shtml and
      http://www.mountain-inter.net/~jfiset/lg/issue15/issue15.html are close but not quite close enough.
      Linus has referred back to this remark often enough.
11. **What are the plans for future versions of the Linux kernel?**
    - (ADB) Linus would be the best person to ask, but I don't know if he would have the time and patience to answer this question. There are some development issues that can be mentioned, though:
        1. PnP support in the kernel. Right now one can get PnP support using the isapnptools user space package and manually tuning the I/O, IRQ and DMA channel allocation, but future Linux kernels will do that for you.
        2. Improved SMP support.
        3. Improved 64 bit code support.
        4. Improved POSIX support.
        5. Improved APM support.
12. **Why does it show BogoMips instead of MHz in the kernel boot message?**
    - (ADB) On some processor architectures it is very difficult to find out the clock speed of the processor, and since the kernel does not depend on determining the MHz rating of a processor to operate correctly, MHz simply do not get calculated at boot time. OTOH, BogoMips get calculated because the kernel bases itself on BogoMips data to implement small time delays (busy loops) needed by various drivers in different circumstances. Note that neither BogoMips nor MHz measure processor performance in any way. See the BogoMips HOWTO by Wim van Dorst for an accurate description of BogoMips. Also take a look at the Linux Benchmarking HOWTO (shameless plug) if you want some basic information on Linux performance measurements.
      Sometimes your BogoMips reading will vary by as much as 30%, from one kernel to another. This is due to changes in the alignment of the BogoMips calibration loop, which interacts with cache behavior. Richard B. Johnson has recently proposed a small patch that takes care of this problem.
13. **I installed kernel x.y.z and package foo doesn't work anymore, what should I do?**
    - (RRR) Check out the /usr/src/linux/Documentation/Changes and make sure you have the recommended versions (or newer) of the relevant software. This is very important. A lot of things are evolving on Linux and newer versions of the kernel may break older packages (especially on the development kernels). If you are using development kernels keep an eye for reports on the kernel list. If all else fails post a bug report (see Q/A on bug reports) to the list.
14. **People talk about user space vs. kernel space. What's the advantage of each?**
    - (REG) User space is what all user (including root) programs run in. It is fully virtual memory (i.e. normally swappable). The X server is in user space, for example. So is your shell. Kernel space is the domain of the kernel (wow!), device drivers and hardware interrupt handlers. Kernel memory is

non-swapable (i.e. it's REAL RAM), and hence should be used sparingly. Also, operations performed in kernel space are not pre-emptive: this means other processes are prevented from running until the operation completes.
Some people think that it's better to implement stuff in kernel space ("so that everyone has it"). In general this is a Bad Idea (tm) (see "creeping featurism" above), since kernel space resources are more "heavy" than user space resources. For example, coding a Mandelbrot fractal generator in kernel space is a *really stupid* idea.
The job of the kernel is to provide a safe and simple interface to hardware and give different processes a fair share of the resources, and to arbitrate access to resources/hardware.
Many ideas are best implemented in user space, with perhaps the absolute minimum of kernel support. The only exceptions to this principle are where it is particularly complicated or inefficient to implement the solution in user space only. This is why filesystems are in the kernel (you *could* put them in user space implemented as daemons), because a kernel implementation is *much* faster.
Note that you can make user space memory non-swappable by using the **mlock(2)** system call. This is a privileged operation and should not be used trivially.

15. **What are threads?**
    - (ADB) Very shortly, threads are "lightweight" processes (see the definition of a process in the Linux Kernel book) that can run in parallel. This is a lot different from standard UNIX processes that are scheduled and run in sequential fashion. More threads information can be found here or in the excellent Linux Parallel Processing HOWTO by Professor Hank Dietz.
    - (REG) When people talk about threads, they usually mean **kernel threads** i.e. threads that can be scheduled by the kernel. On SMP hardware, threads allow you to do things truly concurrently (this is particularly useful for large computations). However, even without SMP hardware, using threads can be good. It can allow you to divide your problems into more logical units, and have each of those run separately. For example, you could have one thread read blocking on a socket while another reads something from disk. Neither operation has to delay the other. Read "Threads Primer" by Bill Lewis and Daniel J. Berg (Prentice Hall, ISBN 0-13-443698-9).
16. **Can I use threads with GNU/Linux?**
    - (REG) Yes! The Linux kernel has the clone(2) system call, which provides the underlying mechanism for implementing a threads library. And Xavier Leroy has provided us with LinuxThreads, a POSIX 1003b implementation of threads for the Linux kernel.
      If you have a libc 5 system, you'll need to install LinuxThreads if it is not already installed. You can get the LinuxThreads library here.
      If you have a libc 6 (aka glibc 2) system, you shouldn't need to do anything. Glibc has LinuxThreads merged in.
17. **You mean threads are implemented in kernel space in GNU/Linux? Why not a hybrid kernel/user space implementation? Wouldn't that be more efficient?**
    - (REG) It is not clear that there is any significant benefit for Linux to have a hybrid threading library. If we look at Solaris Threads, they have a hybrid scheme, and claim that is an advantage. Well, yes, I suppose so, given their environment (the Solaris 2 kernel). They have a very heavy kernel, so a pure kernel space implementation would be too slow (remember the time it takes to enter/exit the kernel). Linux, on the other hand, has a very efficient kernel, so the difference between a kernel context switch under Linux and a user space context switch under Solaris 2 is pretty small. Also note that Solaris Threads took a long time to get right, because of problems such as signal delivery to threads. With a pure kernel threads implementation, signal delivery is much simpler. Fixing the signal delivery problems with Solaris Threads increased the complexity of their library, leading to bloat and performance loss. We don't want to make the same mistakes.
      Now, you may argue that a hybrid scheme under Linux would be **even better**. Maybe. Prove it. Code it and benchmark it. In any case, this is a discussion that is not relevant to the kernel, since a

hybrid scheme is built on top of kernel threads (Solaris 2 builds their threads on top of LWPs (Light Weight Processes) too). It's a user space issue, so please, keep it off the kernel list.
BTW: if you do manage to code something up and it is much faster than pure kernel space threads, you may need some kind of extra kernel support (depending on how you implement things). If that happens, **then** come and talk about it on the kernel list.
The Linux philosophy is to optimize the kernel first, so that all possible implementations can share the benefits.

18. **Can GNU/Linux machines be clustered?**
    - (REG) Different people mean different things when they talk about clustering. Some people want transparent fault tolerance and load balancing of general applications, others want parallel processing of a single job. Most people who talk about fault tolerance expect hardware and OS support of this (if a node goes down, the OS will automatically migrate the application to another node). This is not (yet) available for Linux.
      You can write a fault tolerant application for a network of computers without direct OS support: you just need to structure your application appropriately. Note that a fault tolerant distributed application may also be a parallel, multithreaded application.
      The Beowulf project provides an API and system management software to write parallel distributed applications on a network of Linux machines. The main emphasis here is on parallelism to get maximum processing power, although fault tolerance is possible too. An example of a Beowulf clustered Linux system is Avalon, which has just been listed among the world's 500 most powerful supercomputers.
      Beowulf clusters deliver GFLOPS using arrays of commodity computers. It is an incredibly cheap and elegant way to get significant computing power for e.g. scientific applications.
    - (ADB) Also check the Parallel Processing HOWTO by Professor Hank Dietz.
    - (REG) In June 2000, Mission Critical Linux released Kimberlite which they describe as an "open source linux clustering cabability". Tim Burke, their Cluster Architect describes it thus:
      A Kimberlite Cluster provides support for two server nodes connected to a shared SCSI or Fibre Channel storage subsystem, in an active-active failover environment. The software provides the ability to detect when either node leaves the cluster, and will automatically trigger recovery scripts which perform the procedures necessary to restart applications on the remaining node. When the node rejoins the cluster, applications can be moved back to it, manually or automatically, if required. Sample recovery scripts are provided. Kimberlite is designed to deliver the highest levels of data integrity and be extremely robust. It is suitable for deployment in any environment that requires high availability for un-modified Linux applications.

19. **How well does Linux scale for SMP?**
    - (REG) Reasonably well. Kernel version 2.2 has much better scalability than version 2.0. People are running 4 processor Intel Xeon machines and 14 processor UltraSparc machines. Version 2.2 still has a global kernel lock, but this is often released quite quickly (for example, when the process blocks waiting for a resource and/or data), so the net result is that it is quite unlikely for two processors to compete for the global lock. Experiments with 14 processor UltraSparc machines shows that Linux scales well, indicating that the current locking strategy is not hurting us for these machines.
      Also consider that for parallel processing jobs, the kernel is not involved, so even Kernel v2.0 scaled well for these applications. When we talk about SMP scalability, we are referring to how many IO operations the kernel can perform at the same time.
      Unfortunately some hysterical NT supporters continue to spread FUD that Linux does not scale well on SMP. Efforts to insert a bit of truth have generally fallen on deaf ears. If someone tells you that NT scales better than Linux, ignore them. They're operating in a fact-free zone. Tests indicate that

NT has trouble scaling to 4 processors. There really is no competition.
Note that kernel version 2.3 has replaced the remaining global kernel lock with finer grained locking. This allows Linux to scale well to 64 processor machines and beyond.

20. **Can I lock a process/thread to a CPU?**
    - (REG) Not yet. When scheduling, the kernel takes account of which CPU a thread was on last, and gives a small advantage to the same CPU. This will mean that usually a thread is run on the same CPU for long periods of time, but there is no guarantee.
      Locking threads to a CPU is work under consideration.

Section 8 - Compiler/binutils questions

1. **I downloaded the newest kernel and it doesn't even compile! What's wrong?**
    - (RRR) Make sure you are compiling with gcc 2.7.2.x with default optimizations flags (IOW, leave the Makefiles alone) and a recent binutils. The binutils package is the one that contains the assembler (gas) and linker (ld). See Documentation/Changes for more info. If that works then, experiment with different compiler/optimizations.
    - (REW) Linus cannot test every permutation of drivers and options. He's a selfish little guy. He just compiles the version that runs on his computers, and then releases it. Actually, he sometimes even doesn't compile it before releasing. He's a busy man. Give him a break. Wait for half a day. Someone will post a patch that will fix it within that time. If that doesn't happen for more than a day, fix it yourself, and post the patch to linux-kernel. If you don't have the expertise to do this yourself, please wait for another day, before reporting the problem.
      Please check if it hasn't been reported before. Most companies have a help desk that keeps the end users from bothering the developers. Linux is different: You get to talk to the developers. But don't waste everybody's time by posting stuff that has been reported already.
    - (DBE) Not all ports of the Linux kernel to different hardware platforms are fully merged into the official tree at kernel.org. If you have problems compiling a kernel for a non-i386 architecture please check the related Web pages and mailing-lists of that specific port. A good bet is probably the vger CVS.
2. **What are the recommended compiler/binutils for building 2.0.x kernels and 2.2.x kernels?**
    - (RRR) The recommended compiler is gcc 2.7.2.x. The recommended binutils is 2.8.1.0.23, or 2.9.1.0.3 and newer. Avoid binutils versions from 2.8.1.0.25 to 2.9.1.0.2, these were beta releases and known to be buggy. See the Changes file for details.
3. **Why gcc 2.7.2? I like xyz-compiler better.**
    - (RRR) Quick Answer: it's what Linus uses. Real Answer: gcc 2.7.2 has been extensively tested and proven to be a very stable compiler. What is at issue is not whether other compilers can optimize better than gcc 2.7.2, but whether they will compile the kernel correctly. Current kernels and compilers are very complex pieces of software. There are just too many ways that the two can interact and cause trouble (a recent example: gcc 2.8.x and kernel 2.0.x). By keeping constant one of the variables - the compiler - kernel developers can concentrate on the kernel. If both the compiler and kernel are changing then it's anyone's guess what went wrong.
4. **Can I compile the kernel with gcc 2.8.x, egcs, (add your xyz compiler here)? What about optimizations? How do I get to use -O99, etc.?**
    - (RRR) Sure, it's your kernel. But if it doesn't work, you get to fix it. Seriously now, there is really no point in compiling a production kernel with an experimental compiler. Production kernels should only be compiled with gcc 2.7.2.x, preferably 2.7.2.3. Newer compilers are known to break the 2.0 series kernels, known symptoms of this breakage are hwclock and the X server seg.faulting. Compiling a 2.0 kernel with egcs or gcc 2.8, even after applying the workaround of copying the

ioport.c file from a late 2.1 kernel to 2.0, is not recommended and will inevitably lead to unpredictable kernel behaviour.
Regarding 2.1 kernels, they usually compile fine with other compiler versions, but do NOT complain to the list if your are not using gcc 2.7.2. Linux developers have enough work tracking kernel bugs, to also be swamped with compiler related bugs.
If you want to play with the optimization options, you need to hack the Makefile in arch/i386/Makefile (assuming you have an x86 processor), but if it breaks... well, you should know the answer by now. Also keep in mind that some demented optimizations (such as -O99) may even produce slower and bigger kernels, due to gratuitous loop unrolling and function inlining.
    - (ADB) I think the standard Paul Gortmaker disclaimer (?) is: "If it breaks, you get to keep the pieces." ;-)
5. **I compiled the kernel with xyz compiler and get the following warnings/errors/strange-behavior, should I post a bug report to the list? Should I post a patch?**
    - (RRR) In general, no, unless you get these with gcc 2.7.2. Few exceptions:
      Everyone welcomes code cleanup patches, for instance, newer compilers tend to complain a lot more than old 2.7.2. Some of these warnings may even be warranted (i.e. ambiguous use of else statements), fixing these is a good thing.
      There could be some aging code around that makes too many assumptions about the compiler (especially true about inline assembly), some of the newer compilers break these statements. Fixing these is also a good thing, but be very sure you're are really fixing a bug in the kernel. Workarounds for other compilers will be ignored (if the compiler is buggy, fix the compiler!).
6. **Why does my kernel compilation stops at random locations with: "Internal compiler error: program cc1 caught fatal signal 11."?**
    - (REW) Sometimes bad hardware causes this. Read the Web page at http://www.BitWizard.nl/sig11/ about this. The important word here is random. If it stops at the same place every time, the kernel source might have a glitch or your compiler might be bad. The Web page is mostly about the random error source: hardware. There is a bunch of different error messages that you can get if you have bad or marginal hardware.
    - (ADB) Overclocked processors very often fail long compilations with a sig11, because a long gcc compilation puts more strain on the processor. As the processor heats up, it may attain a point where internal timings get out of spec. At this point, something gives and you get a sig11.
      Also, some old K6 revisions would sig11 when compiling large programs if > 32 Mb of RAM were installed on the Linux box. AMD will exchange these faulty processors for free. Benoit Poulot-Cazajous correctly diagnosed the problem and devised an ingenious test for this bug that is run at boot time in 2.2.x kernels.
7. **What compiler flags should I use to compile modules?**
    - (REG) At the very least, you need these: `-O2 -DMODULE -D__KERNEL__ -DLINUX -Dlinux`

Section 9 - Feature specific questions

1. **GNU/Linux Y2K compliance?**
    - (ADB) Y2K compliance under GNU/Linux is a multi-level problem.
        1. ***Applications***. Check your application sources for routines that only operate on/test the last two digits of the year field/variable(s). Obviously the problem here is that 2000 > 1999, but 00 < 99. Unfortunately, poor programming practices are just as common and unavoidable as death and taxes...
        2. ***Libraries***. Libc5 and glibc are known to be Y2K compliant. Alan Cox mentioned that libc4 had some problems.

3. ***Kernel***. The Linux kernel is y2k compliant. BTW the code snippet in the /arch/i386/kernel/time.c will force those non-y2k compliant RTC implementations to the correct date on 00:00:00 Jan 1, 2000. It's been there for quite some time, now, nice and quiet; added by Alan Modra circa 1994!
4. ***BIOS***. On x86 PC machines, upon boot some BIOS's will wrap back to 1900, later versions will correctly wrap the RTC clock to 2000. This is a rather critical problem in embedded systems if they are not running Linux; if they are running Linux this is solved by Alan Modra's code snippet mentioned above. :-)
5. ***Hardware***. The standard PC RTC chip will not wrap the century. Wrapping must be done in software/BIOS. The chip will store the century data, but it just won't increment it on 00:00:00 Jan 1, 2000. Same issue as BIOS WRT embedded systems.

Testing the kernel, the BIOS and the RTC hardware is relatively easy if you are allowed to reboot the machine; just enter the CMOS setup routine and set the time to Dec 31 1999, 23:58:00. Boot and check what happens.
Checking applications and libraries takes a lot more work... Specially checking applications when you don't have access to the source code :(The only way is simulation. But this is getting off topic: if you don't have access to the source code, then it's not relevant to GNU/Linux. ;)

2. **What is the maximum file size supported under ext2fs? 2 GB?**
   - (REW, AC) In the 2.0.x kernels, maximum **file** size (not to be confused with partition sizes, which can be much larger) under ext2fs is 2GB. Larger files are only supported on 64-bit architectures (Alpha and UltraSPARC) in late 2.1.1xx kernels.
     Files larger than 2GB are difficult to support on 32-bit architectures. This will probably be implemented in the 2.3 kernel series.
3. **GGI/KGI or the Graphics Interface in Kernel Space debate?**
   - (REG, ADB) GGI/KDI information can be found here. The GGI/KGI developers warn against useless debates on the kernel list.
4. **How do I get more than 16 SCSI disks?**
   - (REG) Get kernel version 2.2.0 or later.
5. **What's devfs and why is it a Good Idea (tm)?**
   - (REG) OK, pushing my own barrow here. Devfs allows device drivers to have a direct link with device special files (what you see in /dev). The current dependence on major/minor numbers to provide this link poses scalability and performance problems. Devfs also only has device nodes for devices that you have available. Read the devfs FAQ for more details. Note that devfs went into the official 2.3.46 kernel.
6. **Linux memory management? Zone allocation?**
   - (ADB) Rik van Riel has setup a nice page on Linux memory management. It has a link to an excellent tutorial on virtual memory.
7. **How many open files can I have?**
   - (REG) With kernels 2.0.x you can have 256 open FDs (file descriptors). With 2.2.x you can have 1024. Various patches exist which allow you to increase these limits. Note that this can break select(2).
8. **When will the Linux accept(2) bug be fixed?**
   - (REG) Firstly, this is not a bug in the Linux kernel, despite the fact that the "Sendmail 8.9.0 Known Bugs List" states there is a bug with Linux accept(2). The Linux accept(2) call can return the ETIMEDOUT error when there are system resource problems. This is not wrong, just different from what Sendmail expects. Since accept(2) is not a standard, it cannot be claimed that Linux is violating a standard (like POSIX).
     Unfortunately, the current manual page for accept(2) does not mention that the call can return the ETIMEDOUT error. This should be fixed.

9. **What about STREAMS? I noticed Caldera has a STREAMS package, when will that go in the kernel source proper?**
   - (REG) STREAMS allow you to "push" filters onto a network stack. The idea is that you can have a very primitive network stream of data, and then "push" a filter ("module") that implements TCP/IP or whatever on top of that. Conceptually, this is very nice, as it allows clean separation of your protocol layers. Unfortunately, implementing STREAMS poses many performance problems. Some Unix STREAMS based server telnet implementations even ran the data up to user space and back down again to a pseudo-tty driver, which is very inefficient.
     STREAMS will **never** be available in the standard Linux kernel, it will remain a separate implementation with some add-on kernel support (that comes with the STREAMS package). Linus and his networking gurus are unanimous in their decision to keep STREAMS out of the kernel. They have stated several times on the kernel list when this topic comes up that even optional support will not be included.
   - (REW, quoting Larry McVoy) "It's too bad, I can see why some people think they are cool, but the performance cost - both on uniprocessors and even more so on SMP boxes - is way too high for STREAMS to ever get added to the Linux kernel."
     Please stop asking for them, we have agreement amongst the head guy, the networking guys, and the fringe folks like myself that they aren't going in.
   - (DMG, the STREAMS guy) STREAMS is a good framework for implementing complex and/or deep protocol stacks having nothing to do with TCP/IP, such as SNA. It trades some efficiency for flexibility. You may find the Linux STREAMS package (LiS) to be quite useful if you need to port protocol drivers from Solaris or UnixWare, as Caldera did.
     The Linux STREAMS (LiS) package is available for download if you want to use STREAMS for Linux. The following site also contains a dissenting view, which supports STREAMS.
10. **I need encryption and steganography*. Why isn't it in the kernel?**
    - (TJ) In France and Russia, strong encryption is essentially illegal: according to http://cwis.kub.nl/~frw/people/koops/jenc8bjk.htm , using it there requires a license which is seldom granted. The United States has cumbersome restrictions on exporting such software (it's considered a "munition"--see http://www.epic.org/crypto/export_controls/). Having these features in the standard kernel would therefore cause great inconvenience to people in those countries. However, separate programs and patches to the kernel are available at:
      - ftp://ftp.csua.berkeley.edu/pub/cypherpunks/filesystems/linux/
      - ftp://netwinder.org/users/a/andrewm/loop/
      - http://www.freeswan.org/
      - http://www.inka.de/~bigred/devel/cipe.html
      - http://www.quick.com.au/ftp/pub/sjg/
      - http://www.ssh.org/
      - http://web.mit.edu/kerberos/www/
      - http://tcfs.dia.unisa.it/
      - ftp://ftp.tik.ee.ethz.ch/pub/packages/skip

      (*) Steganography is disguising sensitive data as noise in a digitized image, sound file, or the like.
11. **How about an undelete facility in the kernel?**
    - (REG) This idea keeps being raised again and again. There is no need for kernel support to do this. You can easily do it in user space. There are replacement versions of the **rm** utility which will move/copy files to a wastebasket area instead of actually deleting. If you're really keen, you could implement a wrapper for the **unlink** system call, and use LD_PRELOAD to override the function for all applications.
12. **How about tmpfs for Linux?**
    - (REG) Another idea that keeps being raised again and again. The kernel developers feel that there is

no real need for this. The Linux ext2 filesystem is so good that it outperforms tmpfs (memory-based filesystems) in other operating systems. Jim Nance (jlnance@avanticorp.com) has posted a comparison to linux-kernel. Here is an extract of his message:

```
The original question is enough of an FAQ that I thought it would be
good to have real numbers rather than just my assurances that Linux
has a fast FS layer.  Therefore I wrote a benchmarking program that
creates/writes/destroys files and ran it under several operating
systems and on several types of file systems.  I have included that
program as an attachment to this mail.  Here are the results:

OS                        Hardware        FS Type         Loops/Second
--------------------------------------------------------------------
Linux 2.2.5-ac6           1               nfs             16.33
Linux 2.2.5-ac6           1               arla            73.67
Linux 2.2.5-ac6           1               ext2            15383.32
Solaris 2.6               2               afs             71.33
Solaris 2.6               2               nfs             10.00
Solaris 2.6               2               ufs             23.67
Solaris 2.6               2               tmpfs           9162.32
Digital Unix 4.0D         3               afs             49.33
Digital Unix 4.0D         3               nfs             14.67
Digital Unix 4.0D         3               ufs             28.67
Digital Unix 4.0D         3               memfs           3062.66
Linux 2.0.33              4               afs             69.33
Linux 2.0.33              4               nfs             15.00
Linux 2.0.33              4               ext2            2218.33

Hardware:
1 -> 333 MHz PII, 512M ram, Compaq WDE4360W disk
2 -> Ultra450 class Sun server (300MHz?)
3 -> Personal Workstation 600 AU. 600 MHz alpha.  1.5G ram
4 -> 75 MHz Pentium, 32M ram, Segate ST31200N disk

Notice how Linux writting to an ext2 file system is significantly
faster than any other OS/FS combination.  The next closest is Solaris
writting to tmpfs, and its still far behind ext2.  Its also good to
notice how slow both Solaris and Digital Unix are on their local file
systems.  This is probably why both have a ram base file system.

Please note that this benchmark is intended to measure the time it
takes to create and delete files, which is expensive on most non-linux
systems.  It does not indicate anything about the data I/O rate to an
existing file.
```

So if you're still convinced a tmpfs for Linux is needed, go ahead and write one. It will be interesting to see the results.

- (REG, by Adam Sulmicki) If after reading all the above you still feel you need tmpfs read on. Also if you are in a special situation like having the whole system mounted readonly and some program (like X) screams at you that it needs to write /tmp, you will find it useful. However, keep in mind it is more of a hack than true tmpfs.
  The magic way to do it is:
  - compile ramdisk support into kernel, the option is: `CONFIG_BLK_DEV_RAM=y`
  - Run the following command to create 2mb ext2 ram disk: `/sbin/mke2fs -vm0 /dev/ram 2048`

- mount it: `/sbin/mount /dev/ram /tmp` And you are done.

13. **What is the maximum file size/filesystem size?**
    - (REG) Maximum file size depends on the block size on your filesystem. For ext2 (and UFS, SysVFS and similar filesystems), the limits are:

      ```
      Block size      Maximum file size (GiBytes)
      512 B           2
      1   kiB         16
      2   kiB         128
      4   kiB         1024
      8   kiB         8192   (PAGE_SIZE must be >= 8 kiB)
      ```

      plus a small amount. The limitation is due to the classic triply-indirect addressing scheme. In the future, ext2 will have extent-based addressing, which will overcome this problem.

      The limit for a single filesystem (partition) on a 32 bit CPU is 4 Gi blocks. Each block is 512 Bytes, so that works out to 2 TiB. For 64 bit CPUs, the limits are bigger than you can imagine.
14. **Linux uses lots of swap while I still have stuff in cache. Isn't this wrong?**
    - (MRW) Not really. Linux will swap out binaries which haven't been used for a long time (e.g. lprd on many systems) in favour of retaining data from files which have been used recently (e.g. header files while compiling a big program). This is more efficient. Trust us, we're engineers.
15. **Why don't we add resource forks/streams to Linux filesystems like NT has?**
    - (REG) Resource forks (aka "named streams") are a way of storing multiple "streams" of data in a file. Each stream may be read, written and seeked in just like in files with only one stream of data. Resource forks are used to store ancillary data with files (such as which icon to display for the file when using a graphical filemanager). These extra streams of data may be manipulated by any user who has write access to the file, just as the "primary" stream can be manipulated.
      Unix only supports one "stream" of data per file. Adding support for multiple streams to the Linux kernel is not considered to be especially difficult. However, files with multiple data streams would break a large number of user-space programmes (which currently only manipulate the "primary" stream) and **protocols** (such as ftp, http, email, NFS and many more). A number of new utilities would need to be written, and a large number of shell scripts would have to be audited for correctness in a multiple-stream world. Because of this massive breakage, many kernel developers consider resource forks to be a bad idea.
      Rather than add kernel support, a user-space library could be written which provided easy management of multiple steams of data for applications, while still storing the data in a single Unix file. If someone wants to write such a library, please do so. Once it's completed, send an email to the FAQ maintainer.
      Note that a separate problem is the storage of "extended attributes". These are attributes like file permissions (such as ACLs and POSIX capability sets), which have limited size, and tend to be read and written atomically (i.e. you can't read or write part of the attribute nor seek in it). These usually require special privileges to modify. Also, you normally don't want to copy these attributes when copying files around, thus these extended attributes don't present the problems of massive breakage that resource forks would.

Section 10 - "What's changed between kernels 2.0.x and 2.2.x" questions

1. **Size (source and executable)?**

- (REW) I use the following to quickly estimate the size of a project:
  `cat `find . -name *.c -o -name *.h -o -name *.S `| wc -l`
  I get 811985 (lines of code, including comments and blank lines) when I run this on the 2.0.33 kernel source, and 1460508 when I run this on a 1.0.106 kernel.
  This means that the Linux kernel qualifies as an "extremely large" software product, requiring the effort of 200 to 500 programmers for 5 to 10 years. [Richard Farley: Software engineering concepts, Mc Graw-Hill, 1985, page 11].
  Actually, the Linux kernel is now 7 years old, and has seriously involved 100 to 1000 programmers. (i.e. not counting those that have contributed a "one line fix"). This is my personal guess, so feel free to disagree or tell me otherwise.
- (ADB) I can't compare actual kernel footprints of 2.0.x vs. 2.1.x, but I think it's worth mentionning that 2.1.x kernels have the ability to "jettison" kernel initialization code, freeing the corresponding physical memory. So, even though the **executable** is certainly larger for 2.1.x kernels, you may actually get a smaller memory **footprint**.

2. **Can I use a 2.2.x kernel with a distribution based on a 2.0.x kernel?**
   - (REW) Yes. However some applications may need upgrading. Read /usr/src/linux/Documentation/Changes before you complain about something not working. Also note that the 2.1.(x+1) kernel may need a different set of upgrades than 2.1.x, so you should check the Changes file **every single time** you upgrade your Linux kernel.
3. **New filesystems supported?**
   - ***NTFS*** (read-only). Allows read-only access to Windows NT (tm) partitions.
   - ***Coda***. Coda is an advanced experimental distributed file system with features such as server replication and disconnected operation for laptops. Note that Coda is also available for 2.0.x kernels as an add-on package. Check the Coda Web site for more information.
4. **Performance?**
   - (REG) Here are some performance optimizations which are only available on 2.2.x kernels:
     - ***MTRRs***. MTRRs are registers in PPro and Pentium II CPUs which define memory regions with distinct properties. The default mode for PCI memory accesses is "uncacheable" which means memory and I/O addresses on a PCI peripheral are not cached. For linear frame buffers, a better mode is "write-combining" which allows the CPU to re-order and slightly delay writes to memory so that they can be done in blocks. If you are writing to the PCI bus, you then use PCI burst mode transfers, which are a few times faster.
     - ***Finer grained locking***. Most instances of the global SMP spinlock have been replaced with finer grained locking. This gives much better concurrency.
     - ***User buffer checks***. Replaced the old, painful way of checking if user buffers passed to syscalls were legal by a kernel exception handler. The kernel now assumes a buffer is OK. If not, an exception handler catches the fault and returns -EFAULT to user space. The advantage is that legal buffers no longer need to be carefully checked, which is much faster. The old scheme was also suffering from race conditions under SMP.
     - ***New directory entry cache (dcache)***. This makes file lookups much faster.
       Example: `time find /usr -name gcc -print`
       2.1.104: cold cache: `0.180u 0.460s 0:15.02 4.2% 0+0k 0+0io 85pf+0w`
       2.1.104: warm cache: `0.100u 0.150s 0:00.25 100.0% 0+0k 0+0io 72pf+0w`
       2.0.33: cold cache: `0.100u 0.660s 0:14.87 5.1% 0+0k 0+0io 85pf+0w`
       2.0.33: warm cache: `0.090u 0.600s 0:00.69 100.0% 0+0k 0+0io 72pf+0w`
       Note /usr had 17750 files/directories. We see how with a cold cache (no disc blocks cached) there is very little difference. However, once the cache is warm, we see a fourfold reduction in system time. This is because inode lookups are not needed when a dcache entry is available. Tests performed on a Pentium/MMX 200.

5. **New drivers not available under 2.0.x?**
   - (XXX) Please add your answer here...
6. **What are those __initxxx macros?**
   - (KGB) __initfunc() for example is a macro used to put its first argument (a function) into a special ELF section that is dropped from memory once drivers's initialization is over.
     So if you write an initialization function, whose code will never be used again after your driver is initialized, you can use __initfunc() around its declaration in order to reduce your kernel memory footprint by a few KB of memory. Similarly, __initdata() is used for variables, arrays, strings, etc.
     For implementation details and examples please consult the file include/linux/init.h from a 2.2.x source tree.
     The main idea here is that the kernel memory is not swappable. Jettisoning useless code represents a nice way to save RAM.
7. **I have seen many posts on a "Memory Rusting Effect". Under what circumstances/why does it occur?**
   - (ADB) AFAIK the expression was coined by Bill Metzenthen, who also provided many data points measuring this phenomenon. It describes the not-so-sane behaviour of the MM layer in kernels 2.1.x (where x is approx. > 50) in small memory systems (e.g. 8 MB machines). Alan Cox recommends the following procedure to detect the Memory Rusting Effect:
     (if you have > 8 MB of RAM installed, reboot with **`mem=8M`** as a kernel boot parameter before)
     ```
     time make zImage
     find / -print
     time make zImage
     ```
     Comparing the results of the `time` measurements will show whether the memory has been rusted or not by the `find`.
     Note that just comparing the time it takes to compile a sample kernel under 2.0.x versus 2.1.x is the **wrong** way to measure the Memory Rusting Effect. So don't post this kind of data to the list (we already know it takes longer under 2.1.x in small memory machines).
     The Memory Rusting Effect is being investigated by a number of kernel hackers, and it seems practically solved as of kernel 2.1.111.
8. **Why does ifconfig show incorrect statistics with 2.2.x kernels?**
   - (TJ) This is in linux/Documentation/Changes that comes with the kernel sources:
     ```
     "For support for new features like IPv6, upgrade to the latest
     net-tools.  This will also fix other problems.  For example, the
     format  of /proc/net/dev changed; as a result, an older ifconfig will
     incorrectly report errors."
     ```
9. **My pseudo-tty devices don't work any more. What happened?**
   - (TJ) Support for ptys using a major number of 4 was dropped in Linux 2.1.115. Replace your device files with ones using the new major numbers, 2 and 3. They will work with later 1.3 versions of Linux, and any 2.x version.
10. **Can I use Unix 98 ptys?**
    - (TJ, with much information provided by H. Peter Anvin) Yes, but only if you have a kernel and libc which support them, and if your applications are written and compiled to use them. They will be supported by Linux 2.2 and glibc 2.1.This is in Documentation/Changes that comes with the kernel sources.
      There is also the new standalone libpt by Duncan Simpson which implements the Unix98 PTY API independently of libc (check the Incoming directory on sunsite.unc.edu/Linux and mirrors). You still need to have your apps compiled to use this API, of course.
11. **Capabilities?**
    - (TJ) There's a FAQ on capabilities under Linux at

ftp://ftp.guardian.no/pub/free/linux/capabilities/capfaq.txt.

12. **Kernel API changes**
    - (REG) Some parts of the kernel API (programming interface) have changed from v2.0 to v2.2. This is relevant to the authors of 3rd party device drivers, filesystems and other code. So called "3rd party" code is any kernel code which is not distributed with the official kernel tarball that Linus distributes. A quick reference for programmers wishing to port their code to v2.2 is available here. Note that this document is not relevant for programmes running in user space.
      If you want to port your drivers to the upcoming 2.4 series kernel (in development), then read this, which tells you how to port code from 2.2 to 2.4.

Section 11 - Primer documents

1. **What's a primer document and why should I read it first?**
    - (REG) From time to time various technical debates start on the linux kernel list. Some of these are about quite important topics, however often these debates are repeated every few months or so and much of the same ground is covered each time around. Other times, questions about how some part of the Linux kernel works are posted. Often we see the same old questions time and time again. Don't get me wrong: these are often reasonable questions, it's just that seeing them over and over is something we'd rather avoid.
      This section has some primer document links on various topics that should be read before starting a debate or posing a question (which itself can lead to a debate). This is not an attempt to censor debate, rather, it's an attempt to get you familiar with the current arguments so that you can contribute something new without going over old ground. If it's just a question you have, hopefully we can explain it clearly once, in a single document, and then point everybody to it.
2. **How about having I/O completion ports?**
    - (REG) The existing UNIX semantics - select(2) and poll(2) - for polling for activity on FDs do not scale very well: the overhead is too high with large numbers of FDs. Here is a primer document which explains some of the problems and explores some solutions.
3. **What is the VFS and how does it work?**
    - (REG) The VFS (Virtual FileSystem or Virtual Filesystem Switch, depending on who you talk to) is basically the Linux filesystem layer. It incorporates the dentry cache and standard UNIX file semantics. It also contains a "switch" to specific filesystem types (ext2, vfat, iso9660 and so on), which is why Linux supports so many different filesystems. Read this VFS primer document if you want to know more.
4. **What's the Linux kernel's notion of time?**
    - (ADB) I have tried to put together some information on this topic, which you can find here. Colin Plumb is working on new code for the Linux kernel software clock.
5. **Is there any magic in /proc/scsi that I can use to rescan the SCSI bus?**
    - (TJ) The text below is from drivers/scsi/scsi.c.

```
/*
 * Usage: echo "scsi add-single-device 0 1 2 3" >/proc/scsi/scsi
 * with  "0 1 2 3" replaced by your "Host Channel Id Lun".
 * Consider this feature BETA.
 *     CAUTION: This is not for hotplugging your peripherals. As
 *     SCSI was not designed for this you could damage your
 *     hardware !
 * However perhaps it is legal to switch on an
 * already connected device. It is perhaps not
 * guaranteed this device doesn't corrupt an ongoing data transfer.
 */
```

For a typical discussion of this topic, see http://jpj.net/~trevor/linux/rescan_scsi.txt.gz.

Section 12 - Kernel Programming Questions

1. **When is cli() needed?**
   - (ADB) cli() is a kernel wide function that disables maskable interrupts, whereas sti() is the equivalent function that enables maskable interrupts. Some routines must be run with interrupts disabled, because some peripherals need a guaranteed access sequence, or because the routine is not reentrant and could be reentered from an interrupt, etc. You should never use cli() in a user space program/daemon.
   - (REW) The use of cli() is no longer encouraged. On a single processor, this simply clears an internal CPU flag, which is ANDed with the Maskable Interrupt Request pin. On SMP systems it is quite troublesome to keep ALL processors from servicing interrupts if one processor wants to do something uninterrupted. Currently we try to do locking on a much finer scale. For example, you should put a spinlock on the record that describes THIS INSTANCE of the device that needs the handling without accesses to other registers (e.g. from the interrupt routine). Besides preventing the overhead of trying to keep the other CPUs from handling interrupts, this allows the other CPUs to service interrupts from a second card of the same type in the same machine.
2. **Why do I see sometimes a cli()-sti() pair, and sometimes a save_flags-cli()-restore_flags sequence?**
   - (RRR) The cli()-sti() pair assumes that interrupts were enabled when execution of the code began, and thus proceeds to reenable them at the end. The save_flags-cli-restore_flags sequence doesn't make this assumption. Since the interrupt flag is one of the flags saved by save_flags(), it will be correctly restored to its previous state by restore_flags(). This is critical for code that may be called with interrupts either on or off.
     Using save_flags-cli-restore_flags does incur in a very slight overhead as compared to the cli()-sti) pair, which may be significant for speed critical code (apart from being superfluous if it's known a priori that the code will never be called with interrupts off).
   - (REG) Note that on UP systems cli(), sti() and restore_flags() operate immediately. However, on SMP systems, these functions may have to wait for the global IRQ lock (when another CPU has disabled interrupts). Other than this difference, these functions are SMP safe. It is also safe to call cli() multiple times on one CPU: the global IRQ lock is only grabbed the first time.
3. **Can I call printk() when interrupts are disabled?**
   - (REG) Yes, you can, although you should be careful. Older kernels had the infamous cli()-sti() pair in printk(), so you would get enabled interrupts when returning from printk(), whether printk() was called with interrupts disabled or enabled; whereas recent kernels (e.g. 2.1.107) restore the flags when printk() is finished. You have to know which version of the kernel you are coding for. Read the Source, Luke. Also note that in 2.2.x kernels, printk() grabs a spinlock for SMP machines to avoid any possible deadlocks.
4. **What is the exact purpose of start_bh_atomic() and end_bh_atomic()?**
   - (REG, quoting Krzysztof G. Baranowski) To protect your code from being interrupted by a bottom half handler. Is it mostly used in syscalls and functions called from userspace and is better than cli/sti pair, because most of the time there is no need to mask interrupts on hardware level..
5. **Is it safe to grab the global kernel lock multiple times?**
   - (REG) Yes. The global kernel lock is recursive per process. That means a process can grab the lock multiple times and not deadlock. The lock is released when **unlock_kernel()** is called as many times

as **lock_kernel()** was called.

Section 13 - Mysterious kernel messages

1. **What exactly does a "Socket destroy delayed" mean?**
   - (TJ, from a post by Henner Eisen) Sometimes you may get:
     ```
      Jul 25 22:14:02 zero kernel: Socket destroy delayed (r=212 w=0)
     ```
     in /var/log/messages.
     It means that the kernel cannot free the internal data structures associated with a released socket because there are still socket data buffers (in the above case 212 bytes read memory) accounted to the socket. For this reason, destroying is delayed and tried again later. At some point, after the the remaining sk_buffs accounted to the socket are freed, destroying should succeed. Also:
     ```
     > It keeps spitting that out about every 5 seconds or so. the only way to
     > fix it is to reboot. It doesn't happen very often, but I'd like to find
     > out what's causing it.
     ```
     This might indicate a problem that some kernel entity (i.e protocol module or network device driver), which is responsible for freeing an sk_buff, fails to do so. To help tracking down the problem, try to find out under which circumstances the messages start to appear (in particular, which program closed a socket right before the messages appears, which network protocol does it use, which network device drivers are involved).

2. **What do I do about "inconsistent MTRRs"?**
   - (REG) Sometimes you may get:
     ```
     mtrr: your CPUs had inconsistent ... MTRR settings
     mtrr: probably your BIOS does not setup all CPUs
     ```
     You need to attend an English comprehension course. Note the use of the word "had". Past tense. The MTRRs *were* inconsistent, but they aren't anymore. The kernel fixed them up. Everything is fine now.

Contributing

Contributions are welcome on this FAQ. These can be submitted by Email to Richard (see the Contributors section above).

Sometimes, we may feel your contribution is controversial and/or incomplete and/or could be improved somehow. Also, the turnaround cycle may take up to 2-3 days. In any case, we will write back to you with comments/acknowledgments. Q/A's that are in principle accepted but still in the "process queue" are marked as [WIP].

Last updated on 27 Sep 2000 by Richard Gooch. This document is GPL'ed by its various contributors.