How To Ask Questions The Smart Way

Eric Steven Raymond

Thyrsus Enterprises

<esr@thyrsus.com>

Rick Moen

<respond-auto@linuxmafia.com>

Revision History

Revision 3.10	21 May 2014	esr

New section on Stack Overflow.

Revision 3.9	23 Apr 2013	esr

URL fixes.

Revision 3.8	19 Jun 2012	esr

URL fix.

Revision 3.7	06 Dec 2010	esr

Helpful hints for ESL speakers.

Revision 3.7	02 Nov 2010	esr

Several translations have disappeared.

Revision 3.6	19 Mar 2008	esr

Minor update and new links.

Revision 3.5	2 Jan 2008	esr

Typo fix and some translation links.

Revision 3.4	24 Mar 2007	esr

New section, "When asking about code".

Revision 3.3	29 Sep 2006	esr

Folded in a good suggestion from Kai Niggemann.

Revision 3.2	10 Jan 2006	esr

Folded in edits from Rick Moen.

Revision 3.1	28 Oct 2004	esr

Document 'Google is your friend!'

Revision 3.0	2 Feb 2004	esr

Major addition of stuff about proper etiquette on Web forums.

Table of Contents

Translations

Translations: yArabic Bahasa Indonesian Belorussian Brazilo-Portuguese Chinese Czech Dutch French Georgian German Greek Hebrew Japanese Polish Portuguese Romanian Russian Serbian Spanish Swedish Thai

Disclaimer

Many project websites link to this document in their sections on how to get help. That's fine, it's the use we intended — but if you are a webmaster creating such a link for your project page, please display prominently near the link notice that *we are not a help desk for your project!*

We have learned the hard way that without such a notice, we will repeatedly be pestered by idiots who think having published this document makes it our job to solve all the world's technical problems.

If you're reading this document because you need help, and you walk away with the impression you can get it directly from the authors of this document, *you* are one of the idiots we are talking about. Don't ask *us* questions. We'll just ignore you. We are here to show you how to get help from people who actually know about the software or hardware you're dealing with, but 99.9% of the time that will not be us. Unless you know for *certain* that one of the authors is an expert on what you're dealing with, leave us alone and everybody will be happier.

Introduction

In the world of hackers, the kind of answers you get to your technical questions depends as much on the way you ask the questions as on the difficulty of developing the answer. This guide will teach you how to ask questions in a way more likely to get you a satisfactory answer.

Now that use of open source has become widespread, you can often get as good answers from other, more experienced users as from hackers. This is a Good Thing; users tend to be just a little bit more tolerant of the kind of failures newbies often have. Still, treating experienced users like hackers in the ways we recommend here will generally be the most effective way to get useful answers out of them, too.

The first thing to understand is that hackers actually like hard problems and good, thought-provoking questions about them. If we didn't, we wouldn't be here. If you give us an interesting question to chew on we'll be grateful to you; good questions are a stimulus and a gift. Good questions help us develop our understanding, and often reveal problems we might not have noticed or thought about otherwise. Among hackers, “Good question!” is a strong and sincere compliment.

Despite this, hackers have a reputation for meeting simple questions with what looks like hostility or arrogance. It sometimes looks like we're reflexively rude to newbies and the ignorant. But this isn't really true.

What we are, unapologetically, is hostile to people who seem to be unwilling to think or to do their own homework before asking questions. People like that are time sinks — they take without giving back, and they waste time we could have spent on another question more interesting and another person more worthy of an answer. We call people like this “losers” (and for historical reasons we sometimes spell it “lusers”).

We realize that there are many people who just want to use the software we write, and who have no interest in learning technical details. For most people, a computer is merely a tool, a means to an end; they have more important things to do and lives to live. We acknowledge that, and don't expect everyone to take an interest in the technical matters that fascinate us. Nevertheless, our style of answering questions is tuned for people who *do* take such an interest and are willing to be active participants in problem-solving. That's not going to change. Nor should it; if it did, we would become less effective at the things we do best.

We're (largely) volunteers. We take time out of busy lives to answer questions, and at times we're overwhelmed with them. So we filter ruthlessly. In particular, we throw away questions from people who appear to be losers in order to spend our question-answering time more efficiently, on winners.

If you find this attitude obnoxious, condescending, or arrogant, check your assumptions. We're not asking you to genuflect to us — in fact, most of us would love nothing more than to deal with you as an equal and welcome you into our culture, if you put in the effort required to make that possible. But it's simply not efficient for us to try to help people who are not willing to help themselves. It's OK to be ignorant; it's not OK to play stupid.

So, while it isn't necessary to already be technically competent to get attention from us, it *is* necessary to demonstrate the kind of attitude that leads to competence — alert, thoughtful, observant, willing to be an active partner in developing a solution. If you can't live with this sort of discrimination, we suggest you pay somebody for a commercial support contract instead of asking hackers to personally donate help to you.

If you decide to come to us for help, you don't want to be one of the losers. You don't want to seem like one, either. The best way to get a rapid and responsive answer is to ask it like a person with smarts, confidence, and clues who just happens to need help on one particular problem.

(Improvements to this guide are welcome. You can mail suggestions to esr@thyrsus.com or respond-auto@linuxmafia.com. Note however that this document is not intended to be a general guide to netiquette, and we will generally reject suggestions that are not specifically related to eliciting useful answers in a technical forum.)

Before You Ask

Before asking a technical question by e-mail, or in a newsgroup, or on a website chat board, do the following:

1. Try to find an answer by searching the archives of the forum you plan to post to.

2. Try to find an answer by searching the Web.
3. Try to find an answer by reading the manual.
4. Try to find an answer by reading a FAQ.
5. Try to find an answer by inspection or experimentation.
6. Try to find an answer by asking a skilled friend.
7. If you're a programmer, try to find an answer by reading the source code.

When you ask your question, display the fact that you have done these things first; this will help establish that you're not being a lazy sponge and wasting people's time. Better yet, display what you have *learned* from doing these things. We like answering questions for people who have demonstrated they can learn from the answers.

Use tactics like doing a Google search on the text of whatever error message you get (searching Google groups as well as Web pages). This might well take you straight to fix documentation or a mailing list thread answering your question. Even if it doesn't, saying “I googled on the following phrase but didn't get anything that looked promising” is a good thing to do in e-mail or news postings requesting help, if only because it records what searches won't help. It will also help to direct other people with similar problems to your thread by linking the search terms to what will hopefully be your problem and resolution thread.

Take your time. Do not expect to be able to solve a complicated problem with a few seconds of Googling. Read and understand the FAQs, sit back, relax and give the problem some thought before approaching experts. Trust us, they will be able to tell from your questions how much reading and thinking you did, and will be more willing to help if you come prepared. Don't instantly fire your whole arsenal of questions just because your first search turned up no answers (or too many).

Prepare your question. Think it through. Hasty-sounding questions get hasty answers, or none at all. The more you do to demonstrate that having put thought and effort into solving your problem before seeking help, the more likely you are to actually get help.

Beware of asking the wrong question. If you ask one that is based on faulty assumptions, J. Random Hacker is quite likely to reply with a uselessly literal answer while thinking “Stupid question...”, and hoping the experience of getting what you asked for rather than what you needed will teach you a lesson.

Never assume you are *entitled* to an answer. You are not; you aren't, after all, paying for the service. You will earn an answer, if you earn it, by asking a substantial, interesting, and thought-provoking question — one that implicitly contributes to the experience of the community rather than merely passively demanding knowledge from others.

On the other hand, making it clear that you are able and willing to help in the process of developing the solution is a very good start. “Would someone provide a pointer?”, “What is my example missing?”, and “What site should I have checked?” are more likely to get answered than “Please post the exact procedure I should use.” because you're making it clear that you're truly willing to complete the process if someone can just point you in the right direction.

When You Ask

Choose your forum carefully

Be sensitive in choosing where you ask your question. You are likely to be ignored, or written off as a loser, if you:

- post your question to a forum where it's off topic
- post a very elementary question to a forum where advanced technical questions are expected, or vice-versa
- cross-post to too many different newsgroups
- post a personal e-mail to somebody who is neither an acquaintance of yours nor personally responsible for solving your problem

Hackers blow off questions that are inappropriately targeted in order to try to protect their communications channels from being drowned in irrelevance. You don't want this to happen to you.

The first step, therefore, is to find the right forum. Again, Google and other Web-searching methods are your friend. Use them to find the project webpage most closely associated with the hardware or software giving you difficulties. Usually it will have links to a FAQ (Frequently Asked Questions) list, and to project mailing lists and their archives. These mailing lists are the final places to go for help, if your own efforts (including *reading* those FAQs you found) do not find you a solution. The project page may also describe a bug-reporting procedure, or have a link to one; if so, follow it.

Shooting off an e-mail to a person or forum which you are not familiar with is risky at best. For example, do not assume that the author of an informative webpage wants to be your free consultant. Do not make optimistic guesses about whether your question will be welcome — if you're unsure, send it elsewhere, or refrain from sending it at all.

When selecting a Web forum, newsgroup or mailing list, don't trust the name by itself too far; look for a FAQ or charter to verify your question is on-topic. Read some of the back traffic before posting so you'll get a feel for how things are done there. In fact, it's a very good idea to do a keyword search for words relating to your problem on the newsgroup or mailing list archives before you post. It may find you an answer, and if not it will help you formulate a better question.

Don't shotgun-blast all the available help channels at once, that's like yelling and irritates people. Step through them softly.

Know what your topic is! One of the classic mistakes is asking questions about the Unix or Windows programming interface in a forum devoted to a language or library or tool portable across both. If you don't understand why this is a blunder, you'd be best off not asking any questions at all until you get it.

In general, questions to a well-selected public forum are more likely to get useful answers than equivalent questions to a private one. There are multiple reasons for this. One is simply the size of the pool of potential respondents. Another is the size of the audience; hackers would rather answer questions that educate many people than questions serving only a few.

Understandably, skilled hackers and authors of popular software are already receiving more than their fair share of mis-targeted messages. By adding to the flood, you could in extreme cases even be the straw that

breaks the camel's back — quite a few times, contributors to popular projects have withdrawn their support because collateral damage in the form of useless e-mail traffic to their personal accounts became unbearable.

Stack Overflow

Search, *then* ask on Stack Exchange

In recent years, the Stack Exchange community of sites has emerged as a major resource for answering technical and other questions and is even the preferred forum for many open-source projects.

Start with a Google search before looking at Stack Exchange; Google indexes it in real time. There's a very good chance someone has already asked a similar question, and the Stack Exchange sites are often near the top of the search results. If you didn't find anything through Google, search again on the specific site most relevant to your question (see below). Searching with tags can help narrow down the results.

If you still didn't find anything, post your question on the *one* site where it's most on-topic. Use the formatting tools, especially for code, and add tags that are related to the substance of your question (particularly the name of the programming language, operating system, or library you're having trouble with). If a commenter asks you for more information, edit your main post to include it. If any answer is helpful, click the up arrow to upvote it; if an answer gives a solution to your problem, click the check under the voting arrows to accept it as correct.

Stack Exchange has grown to over 100 sites, but here are the most likely candidates:

- Super User is for questions about general-purpose computing. If your question isn't about code or programs that you talk to only over a network connection, it probably goes here.
- Stack Overflow is for questions about programming.
- Server Fault is for questions about server and network administration.

Several projects have their own specific sites, including Android, Ubuntu, TeX/LaTeX, and SharePoint. Check the Stack Exchange site for an up-to-date list.

Web and IRC forums

Your local user group, or your Linux distribution, may advertise a Web forum or IRC channel where newbies can get help. (In non-English-speaking countries newbie forums are still more likely to be mailing lists.) These are good first places to ask, especially if you think you may have tripped over a relatively simple or common problem. An advertised IRC channel is an open invitation to ask questions there and often get answers in real time.

In fact, if you got the program that is giving you problems from a Linux distribution (as is common today), it may be better to ask in the distro's forum/list before trying the program's project forum/list. The project's hackers may just say, “use *our* build”.

Before posting to any Web forum, check if it has a Search feature. If it does, try a couple of keyword searches for something like your problem; it just might help. If you did a general Web search before (as you should have), search the forum anyway; your Web-wide search engine might not have all of this forum indexed recently.

There is an increasing tendency for projects to do user support over a Web forum or IRC channel, with e-mail

reserved more for development traffic. So look for those channels first when seeking project-specific help.

As a second step, use project mailing lists

When a project has a development mailing list, write to the mailing list, not to individual developers, even if you believe you know who can best answer your question. Check the documentation of the project and its homepage for the address of a project mailing list, and use it. There are several good reasons for this policy:

- Any question good enough to be asked of one developer will also be of value to the whole group. Contrariwise, if you suspect your question is too dumb for a mailing list, it's not an excuse to harass individual developers.

- Asking questions on the list distributes load among developers. The individual developer (especially if he's the project leader) may be too busy to answer your questions.

- Most mailing lists are archived and the archives are indexed by search engines. If you ask your question on-list and it is answered, a future querent could find your question and the answer on the Web instead of asking it again.

- If certain questions are seen to be asked often, developers can use that information to improve the documentation or the software itself to be less confusing. But if those questions are asked in private, nobody has the complete picture of what questions are asked most often.

If a project has both a “user” and a “developer” (or “hacker”) mailing list or Web forum, and you are not hacking on the code, ask in the “user” list/forum. Do not assume that you will be welcome on the developer list, where they're likely to experience your question as noise disrupting their developer traffic.

However, if you are *sure* your question is non-trivial, and you get no answer in the “user” list/forum for several days, try the “developer” one. You would be well advised to lurk there for a few daysor at least review the last few days of archived messages, to learn the local folkways before posting (actually this is good advice on any private or semi-private list).

If you cannot find a project's mailing list address, but only see the address of the maintainer of the project, go ahead and write to the maintainer. But even in that case, don't assume that the mailing list doesn't exist. Mention in your e-mail that you tried and could not find the appropriate mailing list. Also mention that you don't object to having your message forwarded to other people. (Many people believe that private e-mail should remain private, even if there is nothing secret in it. By allowing your message to be forwarded you give your correspondent a choice about how to handle your e-mail.)

Use meaningful, specific subject headers

On mailing lists, newsgroups or Web forums, the subject header is your golden opportunity to attract qualified experts' attention in around 50 characters or fewer. Don't waste it on babble like “Please help me” (let alone “PLEASE HELP ME!!!!”; messages with subjects like that get discarded by reflex). Don't try to impress us with the depth of your anguish; use the space for a super-concise problem description instead.

One good convention for subject headers, used by many tech support organizations, is “object - deviation”. The “object” part specifies what thing or group of things is having a problem, and the “deviation” part describes the deviation from expected behavior.

Stupid:

> HELP! Video doesn't work properly on my laptop!

Smart:

> X.org 6.8.1 misshapen mouse cursor, Fooware MV1005 vid. chipset

Smarter:

> X.org 6.8.1 mouse cursor on Fooware MV1005 vid. chipset - is misshapen

The process of writing an “object-deviation” description will help you organize your thinking about the problem in more detail. What is affected? Just the mouse cursor or other graphics too? Is this specific to the X.org version of X? To version 6.8.1? Is this specific to Fooware video chipsets? To model MV1005? A hacker who sees the result can immediately understand what it is that you are having a problem with *and* the problem you are having, at a glance.

More generally, imagine looking at the index of an archive of questions, with just the subject lines showing. Make your subject line reflect your question well enough that the next guy searching the archive with a question similar to yours will be able to follow the thread to an answer rather than posting the question again.

If you ask a question in a reply, be sure to change the subject line to indicate that you're asking a question. A Subject line that looks like “Re: test” or “Re: new bug” is less likely to attract useful amounts of attention. Also, pare quotation of previous messages to the minimum consistent with cluing in new readers.

Do not simply hit reply to a list message in order to start an entirely new thread. This will limit your audience. Some mail readers, like mutt, allow the user to sort by thread and then hide messages in a thread by folding the thread. Folks who do that will never see your message.

Changing the subject is not sufficient. Mutt, and probably other mail readers, looks at other information in the e-mail's headers to assign it to a thread, not the subject line. Instead start an entirely new e-mail.

On Web forums the rules of good practice are slightly different, because messages are usually much more tightly bound to specific discussion threads and often invisible outside those threads. Changing the subject when asking a question in reply is not essential. Not all forums even allow separate subject lines on replies, and nearly nobody reads them when they do. However, asking a question in a reply is a dubious practice in itself, because it will only be seen by those who are watching this thread. So, unless you are sure you *want* to ask only the people currently active in the thread, start a new one.

Make it easy to reply

Finishing your query with “Please send your reply to... ” makes it quite unlikely you will get an answer. If you can't be bothered to take even the few seconds required to set up a correct Reply-To header in your mail agent, we can't be bothered to take even a few seconds to think about your problem. If your mail program doesn't permit this, get a better mail program. If your operating system doesn't support any e-mail programs that permit this, get a better operating system.

In Web forums, asking for a reply by e-mail is outright rude, unless you believe the information may be sensitive (and somebody will, for some unknown reason, let you but not the whole forum know it). If you want an e-mail copy when somebody replies in the thread, request that the Web forum send it; this feature is supported almost everywhere under options like “watch this thread”, “send e-mail on answers”, etc.

Write in clear, grammatical, correctly-spelled language

We've found by experience that people who are careless and sloppy writers are usually also careless and sloppy at thinking and coding (often enough to bet on, anyway). Answering questions for careless and sloppy thinkers is not rewarding; we'd rather spend our time elsewhere.

So expressing your question clearly and well is important. If you can't be bothered to do that, we can't be bothered to pay attention. Spend the extra effort to polish your language. It doesn't have to be stiff or formal — in fact, hacker culture values informal, slangy and humorous language used with precision. But it has to *be* precise; there has to be some indication that you're thinking and paying attention.

Spell, punctuate, and capitalize correctly. Don't confuse “its” with “it's”, “loose” with “lose”, or “discrete” with “discreet”. Don't TYPE IN ALL CAPS; this is read as shouting and considered rude. (All-smalls is only slightly less annoying, as it's difficult to read. Alan Cox can get away with it, but you can't.)

More generally, if you write like a semi-literate boob you will very likely be ignored. So don't use instant-messaging shortcuts. Spelling "you" as "u" makes you look like a semi-literate boob to save two entire keystrokes. Worse: writing like a l33t script kiddie hax0r is the absolute kiss of death and guarantees you will receive nothing but stony silence (or, at best, a heaping helping of scorn and sarcasm) in return.

If you are asking questions in a forum that does not use your native language, you will get a limited amount of slack for spelling and grammar errors — but no extra slack at all for laziness (and yes, we can usually spot that difference). Also, unless you know what your respondent's languages are, write in English. Busy hackers tend to simply flush questions in languages they don't understand, and English is the working language of the Internet. By writing in English you minimize your chances that your question will be discarded unread.

If you are writing in English but it is a second language for you, it is good form to alert potential respondents to potential language difficulties and options for getting around them. Examples:

- English is not my native language; please excuse typing errors.
- If you speak $LANGUAGE, please email/PM me; I may need assistance translating my question.
- I am familiar with the technical terms, but some slang expressions and idioms are difficult for me.
- I've posted my question in $LANGUAGE and English. I'll be glad to translate responses, if you only use one or the other.

Send questions in accessible, standard formats

If you make your question artificially hard to read, it is more likely to be passed over in favor of one that isn't. So:

- Send plain text mail, not HTML. (It's not hard to turn off HTML.)
- MIME attachments are usually OK, but only if they are real content (such as an attached source file or patch), and not merely boilerplate generated by your mail client (such as another copy of your message).
- Don't send e-mail in which entire paragraphs are single multiply-wrapped lines. (This makes it too difficult to reply to just part of the message.) Assume that your respondents will be reading mail on 80-character-wide text displays and set your line wrap accordingly, to something less than 80.

- However, do *not* wrap data (such as log file dumps or session transcripts) at any fixed column width. Data should be included as-is, so respondents can have confidence that they are seeing what you saw.

- Don't send MIME Quoted-Printable encoding to an English-language forum. This encoding can be necessary when you're posting in a language ASCII doesn't cover, but many e-mail agents don't support it. When they break, all those =20 glyphs scattered through the text are ugly and distracting — or may actively sabotage the semantics of your text.

- Never, *ever* expect hackers to be able to read closed proprietary document formats like Microsoft Word or Excel. Most hackers react to these about as well as you would to having a pile of steaming pig manure dumped on your doorstep. Even when they can cope, they resent having to do so.

- If you're sending e-mail from a Windows machine, turn off Microsoft's problematic “Smart Quotes” feature (From Tools > AutoCorrect Options, clear the smart quotes checkbox under AutoFormat As You Type.). This is so you'll avoid sprinkling garbage characters through your mail.

- In Web forums, do not abuse “smiley” and “HTML” features (when they are present). A smiley or two is usually OK, but colored fancy text tends to make people think you are lame. Seriously overusing smileys and color and fonts will make you come off like a giggly teenage girl, which is not generally a good idea unless you are more interested in sex than answers.

If you're using a graphical-user-interface mail client such as Netscape Messenger, MS Outlook, or their ilk, beware that it may violate these rules when used with its default settings. Most such clients have a menu-based “View Source” command. Use this on something in your sent-mail folder, verifying sending of plain text without unnecessary attached crud.

Be precise and informative about your problem

- Describe the symptoms of your problem or bug carefully and clearly.

- Describe the environment in which it occurs (machine, OS, application, whatever). Provide your vendor's distribution and release level (e.g.: “Fedora Core 7”, “Slackware 9.1”, etc.).

- Describe the research you did to try and understand the problem before you asked the question.

- Describe the diagnostic steps you took to try and pin down the problem yourself before you asked the question.

- Describe any possibly relevant recent changes in your computer or software configuration.

- If at all possible, provide a way to *reproduce the problem in a controlled environment*.

Do the best you can to anticipate the questions a hacker will ask, and answer them in advance in your request for help.

Giving hackers the ability to reproduce the problem in a controlled environment is especially important if you are reporting something you think is a bug in code. When you do this, your odds of getting a useful answer and the speed with which you are likely to get that answer both improve tremendously.

Simon Tatham has written an excellent essay entitled How to Report Bugs Effectively. I strongly recommend that you read it.

Volume is not precision

You need to be precise and informative. This end is not served by simply dumping huge volumes of code or data into a help request. If you have a large, complicated test case that is breaking a program, try to trim it and make it as small as possible.

This is useful for at least three reasons. One: being seen to invest effort in simplifying the question makes it more likely you'll get an answer, Two: simplifying the question makes it more likely you'll get a *useful* answer. Three: In the process of refining your bug report, you may develop a fix or workaround yourself.

Don't rush to claim that you have found a bug

When you are having problems with a piece of software, don't claim you have found a bug unless you are very, *very* sure of your ground. Hint: unless you can provide a source-code patch that fixes the problem, or a regression test against a previous version that demonstrates incorrect behavior, you are probably not sure enough. This applies to webpages and documentation, too; if you have found a documentation “bug”, you should supply replacement text and which pages it should go on.

Remember, there are many other users that are not experiencing your problem. Otherwise you would have learned about it while reading the documentation and searching the Web (you did do that before complaining, didn't you?). This means that very probably it is you who are doing something wrong, not the software.

The people who wrote the software work very hard to make it work as well as possible. If you claim you have found a bug, you'll be impugning their competence, which may offend some of them even if you are correct. It's especially undiplomatic to yell “bug” in the Subject line.

When asking your question, it is best to write as though you assume *you* are doing something wrong, even if you are privately pretty sure you have found an actual bug. If there really is a bug, you will hear about it in the answer. Play it so the maintainers will want to apologize to you if the bug is real, rather than so that you will owe them an apology if you have messed up.

Grovelling is not a substitute for doing your homework

Some people who get that they shouldn't behave rudely or arrogantly, demanding an answer, retreat to the opposite extreme of grovelling. “I know I'm just a pathetic newbie loser, but...”. This is distracting and unhelpful. It's especially annoying when it's coupled with vagueness about the actual problem.

Don't waste your time, or ours, on crude primate politics. Instead, present the background facts and your question as clearly as you can. That is a better way to position yourself than by grovelling.

Sometimes Web forums have separate places for newbie questions. If you feel you do have a newbie question, just go there. But don't grovel there either.

Describe the problem's symptoms, not your guesses

It's not useful to tell hackers what you think is causing your problem. (If your diagnostic theories were such hot stuff, would you be consulting others for help?) So, make sure you're telling them the raw symptoms of what goes wrong, rather than your interpretations and theories. Let them do the interpretation and diagnosis. If you feel it's important to state your guess, clearly label it as such and describe why that answer isn't working for you.

Stupid:

> I'm getting back-to-back SIG11 errors on kernel compiles, and suspect a hairline crack on one of the motherboard traces. What's the best way to check for those?

Smart:

> My home-built K6/233 on an FIC-PA2007 motherboard (VIA Apollo VP2 chipset) with 256MB Corsair PC133 SDRAM starts getting frequent SIG11 errors about 20 minutes after power-on during the course of kernel compiles, but never in the first 20 minutes. Rebooting doesn't restart the clock, but powering down overnight does. Swapping out all RAM didn't help. The relevant part of a typical compile session log follows.

Since the preceding point seems to be a tough one for many people to grasp, here's a phrase to remind you: "All diagnosticians are from Missouri." That US state's official motto is "Show me" (earned in 1899, when Congressman Willard D. Vandiver said "I come from a country that raises corn and cotton and cockleburs and Democrats, and frothy eloquence neither convinces nor satisfies me. I'm from Missouri. You've got to show me.") In diagnosticians' case, it's not a matter of skepticism, but rather a literal, functional need to see whatever is as close as possible to the same raw evidence that you see, rather than your surmises and summaries. Show us.

Describe your problem's symptoms in chronological order

The clues most useful in figuring out something that went wrong often lie in the events immediately prior. So, your account should describe precisely what you did, and what the machine and software did, leading up to the blowup. In the case of command-line processes, having a session log (e.g., using the script utility) and quoting the relevant twenty or so lines is very useful.

If the program that blew up on you has diagnostic options (such as -v for verbose), try to select options that will add useful debugging information to the transcript. Remember that more is not necessarily better; try to choose a debug level that will inform rather than drowning the reader in junk.

If your account ends up being long (more than about four paragraphs), it might be useful to succinctly state the problem up top, then follow with the chronological tale. That way, hackers will know what to watch for in reading your account.

Describe the goal, not the step

If you are trying to find out how to do something (as opposed to reporting a bug), begin by describing the goal. Only then describe the particular step towards it that you are blocked on.

Often, people who need technical help have a high-level goal in mind and get stuck on what they think is one particular path towards the goal. They come for help with the step, but don't realize that the path is wrong. It can take substantial effort to get past this.

Stupid:

> How do I get the color-picker on the FooDraw program to take a hexadecimal RGB value?

Smart:

> I'm trying to replace the color table on an image with values of my choosing. Right now the only way I can see to do this is by editing each table slot, but I can't get FooDraw's color picker to take a hexadecimal RGB value.

The second version of the question is smart. It allows an answer that suggests a tool better suited to the task.

Don't ask people to reply by private e-mail

Hackers believe solving problems should be a public, transparent process during which a first try at an answer can and should be corrected if someone more knowledgeable notices that it is incomplete or incorrect. Also, helpers get some of their reward for being respondents from being seen to be competent and knowledgeable by their peers.

When you ask for a private reply, you are disrupting both the process and the reward. Don't do this. It's the *respondent's* choice whether to reply privately — and if he does, it's usually because he thinks the question is too ill-formed or obvious to be interesting to others.

There is one limited exception to this rule. If you think the question is such that you are likely to get many answers that are all closely similar, then the magic words are “e-mail me and I'll summarize the answers for the group”. It is courteous to try and save the mailing list or newsgroup a flood of substantially identical postings — but you have to keep the promise to summarize.

Be explicit about your question

Open-ended questions tend to be perceived as open-ended time sinks. Those people most likely to be able to give you a useful answer are also the busiest people (if only because they take on the most work themselves). People like that are allergic to open-ended time sinks, thus they tend to be allergic to open-ended questions.

You are more likely to get a useful response if you are explicit about what you want respondents to do (provide pointers, send code, check your patch, whatever). This will focus their effort and implicitly put an upper bound on the time and energy a respondent must allocate to helping you. This is good.

To understand the world the experts live in, think of expertise as an abundant resource and time to respond as a scarce one. The less of a time commitment you implicitly ask for, the more likely you are to get an answer from someone really good and really busy.

So it is useful to frame your question to minimize the time commitment required for an expert to field it — but this is often not the same thing as simplifying the question. Thus, for example, “Would you give me a pointer to a good explanation of X?” is usually a smarter question than “Would you explain X, please?”. If you have some malfunctioning code, it is usually smarter to ask for someone to explain what's wrong with it than it is to ask someone to fix it.

When asking about code

Don't ask others to debug your broken code without giving a hint what sort of problem they should be searching for. Posting a few hundred lines of code, saying "it doesn't work", will get you ignored. Posting a dozen lines of code, saying "after line 7 I was expecting to see <x>, but <y> occurred instead" is much more likely to get you a response.

The most effective way to be precise about a code problem is to provide a minimal bug-demonstrating test case. What's a minimal test case? It's an illustration of the problem; just enough code to exhibit the undesirable behavior and no more. How do you make a minimal test case? If you know what line or section of code is producing the problematic behavior, make a copy of it and add just enough supporting code to produce a complete example (i.e. enough that the source is acceptable to the compiler/interpreter/whatever application processes it). If you can't narrow it down to a particular section, make a copy of the source and start removing chunks that don't affect the problematic behavior. The smaller your minimal test case is, the

better (see the section called “Volume is not precision”).

Generating a really small minimal test case will not always be possible, but trying to is good discipline. It may help you learn what you need to solve the problem on your own — and even when it doesn't, hackers like to see that you have tried. It will make them more cooperative.

If you simply want a code review, say as much up front, and be sure to mention what areas you think might particularly need review and why.

Don't post homework questions

Hackers are good at spotting homework questions; most of us have done them ourselves. Those questions are for *you* to work out, so that you will learn from the experience. It is OK to ask for hints, but not for entire solutions.

If you suspect you have been passed a homework question, but can't solve it anyway, try asking in a user group forum or (as a last resort) in a “user” list/forum of a project. While the hackers *will* spot it, some of the advanced users may at least give you a hint.

Prune pointless queries

Resist the temptation to close your request for help with semantically-null questions like “Can anyone help me?” or “Is there an answer?” First: if you've written your problem description halfway competently, such tacked-on questions are at best superfluous. Second: because they are superfluous, hackers find them annoying — and are likely to return logically impeccable but dismissive answers like “Yes, you can be helped” and “No, there is no help for you.”

In general, asking yes-or-no questions is a good thing to avoid unless you want a yes-or-no answer.

Don't flag your question as “Urgent”, even if it is for you

That's your problem, not ours. Claiming urgency is very likely to be counter-productive: most hackers will simply delete such messages as rude and selfish attempts to elicit immediate and special attention. Furthermore, the word 'Urgent' (and other similar attempts to grab attention in the subject line) often triggers spam filters - your intended recipients might never see it at all!

There is one semi-exception. It can be worth mentioning if you're using the program in some high-profile place, one that the hackers will get excited about; in such a case, if you're under time pressure, and you say so politely, people may get interested enough to answer faster.

This is a very risky thing to do, however, because the hackers' metric for what is exciting probably differs from yours. Posting from the International Space Station would qualify, for example, but posting on behalf of a feel-good charitable or political cause would almost certainly not. In fact, posting “Urgent: Help me save the fuzzy baby seals!” will reliably get you shunned or flamed even by hackers who think fuzzy baby seals are important.

If you find this mysterious, re-read the rest of this how-to repeatedly until you understand it before posting anything at all.

Courtesy never hurts, and sometimes helps

Be courteous. Use “Please” and “Thanks for your attention” or “Thanks for your consideration”. Make it

clear you appreciate the time people spend helping you for free.

To be honest, this isn't as important as (and cannot substitute for) being grammatical, clear, precise and descriptive, avoiding proprietary formats etc.; hackers in general would rather get somewhat brusque but technically sharp bug reports than polite vagueness. (If this puzzles you, remember that we value a question by what it teaches us.)

However, if you've got your technical ducks in a row, politeness does increase your chances of getting a useful answer.

(We must note that the only serious objection we've received from veteran hackers to this HOWTO is with respect to our previous recommendation to use “Thanks in advance”. Some hackers feel this connotes an intention not to thank anybody afterwards. Our recommendation is to either say “Thanks in advance” first *and* thank respondents afterwards, or express courtesy in a different way, such as by saying “Thanks for your attention” or “Thanks for your consideration”.)

Follow up with a brief note on the solution

Send a note after the problem has been solved to all who helped you; let them know how it came out and thank them again for their help. If the problem attracted general interest in a mailing list or newsgroup, it's appropriate to post the followup there.

Optimally, the reply should be to the thread started by the original question posting, and should have ‘FIXED’, ‘RESOLVED’ or an equally obvious tag in the subject line. On mailing lists with fast turnaround, a potential respondent who sees a thread about “Problem X” ending with “Problem X - FIXED” knows not to waste his/her time even reading the thread (unless (s)he personally finds Problem X interesting) and can therefore use that time solving a different problem.

Your followup doesn't have to be long and involved; a simple “Howdy — it was a failed network cable! Thanks, everyone. - Bill” would be better than nothing. In fact, a short and sweet summary is better than a long dissertation unless the solution has real technical depth. Say what action solved the problem, but you need not replay the whole troubleshooting sequence.

For problems with some depth, it is appropriate to post a summary of the troubleshooting history. Describe your final problem statement. Describe what worked as a solution, and indicate avoidable blind alleys *after that*. The blind alleys should come after the correct solution and other summary material, rather than turning the follow-up into a detective story. Name the names of people who helped you; you'll make friends that way.

Besides being courteous and informative, this sort of followup will help others searching the archive of the mailing-list/newsgroup/forum to know exactly which solution helped you and thus may also help them.

Last, and not least, this sort of followup helps everybody who assisted feel a satisfying sense of closure about the problem. If you are not a techie or hacker yourself, trust us that this feeling is very important to the gurus and experts you tapped for help. Problem narratives that trail off into unresolved nothingness are frustrating things; hackers itch to see them resolved. The goodwill that scratching that itch earns you will be very, very helpful to you next time you need to pose a question.

Consider how you might be able to prevent others from having the same problem in the future. Ask yourself if a documentation or FAQ patch would help, and if the answer is yes send that patch to the maintainer.

Among hackers, this sort of good followup behavior is actually more important than conventional politeness. It's how you get a reputation for playing well with others, which can be a very valuable asset.

How To Interpret Answers

RTFM and STFW: How To Tell You've Seriously Screwed Up

There is an ancient and hallowed tradition: if you get a reply that reads “RTFM”, the person who sent it thinks you should have Read The Fucking Manual. He or she is almost certainly right. Go read it.

RTFM has a younger relative. If you get a reply that reads “STFW”, the person who sent it thinks you should have Searched The Fucking Web. He or she is almost certainly right. Go search it. (The milder version of this is when you are told “Google is your friend!”)

In Web forums, you may also be told to search the forum archives. In fact, someone may even be so kind as to provide a pointer to the previous thread where this problem was solved. But do not rely on this consideration; do your archive-searching before asking.

Often, the person telling you to do a search has the manual or the web page with the information you need open, and is looking at it as he or she types. These replies mean that he thinks (a) the information you need is easy to find, and (b) you will learn more if you seek out the information than if you have it spoon-fed to you.

You shouldn't be offended by this; by hacker standards, your respondent is showing you a rough kind of respect simply by not ignoring you. You should instead be thankful for this grandmotherly kindness.

If you don't understand...

If you don't understand the answer, do not immediately bounce back a demand for clarification. Use the same tools that you used to try and answer your original question (manuals, FAQs, the Web, skilled friends) to understand the answer. Then, if you still need to ask for clarification, exhibit what you have learned.

For example, suppose I tell you: “It sounds like you've got a stuck zentry; you'll need to clear it.” Then: here's a *bad* followup question: “What's a zentry?” Here's a *good* followup question: “OK, I read the man page and zentries are only mentioned under the -z and -p switches. Neither of them says anything about clearing zentries. Is it one of these or am I missing something here?”

Dealing with rudeness

Much of what looks like rudeness in hacker circles is not intended to give offense. Rather, it's the product of the direct, cut-through-the-bullshit communications style that is natural to people who are more concerned about solving problems than making others feel warm and fuzzy.

When you perceive rudeness, try to react calmly. If someone is really acting out, it is very likely a senior person on the list or newsgroup or forum will call him or her on it. If that *doesn't* happen and you lose your temper, it is likely that the person you lose it at was behaving within the hacker community's norms and *you* will be considered at fault. This will hurt your chances of getting the information or help you want.

On the other hand, you will occasionally run across rudeness and posturing that is quite gratuitous. The flip-side of the above is that it is acceptable form to slam real offenders quite hard, dissecting their misbehavior with a sharp verbal scalpel. Be very, very sure of your ground before you try this, however. The line between correcting an incivility and starting a pointless flamewar is thin enough that hackers themselves not infrequently blunder across it; if you are a newbie or an outsider, your chances of avoiding such a blunder are low. If you're after information rather than entertainment, it's better to keep your fingers off the keyboard than to risk this.

(Some people assert that many hackers have a mild form of autism or Asperger's Syndrome, and are actually missing some of the brain circuitry that lubricates “normal” human social interaction. This may or may not be true. If you are not a hacker yourself, it may help you cope with our eccentricities if you think of us as being brain-damaged. Go right ahead. We won't care; we *like* being whatever it is we are, and generally have a healthy skepticism about clinical labels.)

In the next section, we'll talk about a different issue; the kind of “rudeness” you'll see when *you* misbehave.

On Not Reacting Like A Loser

Odds are you'll screw up a few times on hacker community forums — in ways detailed in this article, or similar. And you'll be told exactly how you screwed up, possibly with colourful asides. In public.

When this happens, the worst thing you can do is whine about the experience, claim to have been verbally assaulted, demand apologies, scream, hold your breath, threaten lawsuits, complain to people's employers, leave the toilet seat up, etc. Instead, here's what you do:

Get over it. It's normal. In fact, it's healthy and appropriate.

Community standards do not maintain themselves: They're maintained by people actively applying them, visibly, *in public*. Don't whine that all criticism should have been conveyed via private e-mail: That's not how it works. Nor is it useful to insist you've been personally insulted when someone comments that one of your claims was wrong, or that his views differ. Those are loser attitudes.

There have been hacker forums where, out of some misguided sense of hyper-courtesy, participants are banned from posting any fault-finding with another's posts, and told “Don't say anything if you're unwilling to help the user.” The resulting departure of clueful participants to elsewhere causes them to descend into meaningless babble and become useless as technical forums.

Exaggeratedly “friendly” (in that fashion) or useful: Pick one.

Remember: When that hacker tells you that you've screwed up, and (no matter how gruffly) tells you not to do it again, he's acting out of concern for (1) you and (2) his community. It would be much easier for him to ignore you and filter you out of his life. If you can't manage to be grateful, at least have a little dignity, don't whine, and don't expect to be treated like a fragile doll just because you're a newcomer with a theatrically hypersensitive soul and delusions of entitlement.

Sometimes people will attack you personally, flame without an apparent reason, etc., even if you don't screw up (or have only screwed up in their imagination). In this case, complaining is the way to *really* screw up.

These flamers are either lamers who don't have a clue but believe themselves to be experts, or would-be psychologists testing whether you'll screw up. The other readers either ignore them, or find ways to deal with them on their own. The flamers' behavior creates problems for themselves, which don't have to concern you.

Don't let yourself be drawn into a flamewar, either. Most flames are best ignored — after you've checked whether they are really flames, not pointers to the ways in which you have screwed up, and not cleverly ciphered answers to your real question (this happens as well).

Questions Not To Ask

Here are some classic stupid questions, and what hackers are thinking when they don't answer them.

Q: Where can I find program or resource X?

A: The same place I'd find it, fool — at the other end of a web search. Ghod, doesn't everybody know how to use Google yet?

Q: How can I use X to do Y?

A: If what you want is to do Y, you should ask that question without pre-supposing the use of a method that may not be appropriate. Questions of this form often indicate a person who is not merely ignorant about X, but confused about what problem Y they are solving and too fixated on the details of their particular situation. It is generally best to ignore such people until they define their problem better.

Q: How can I configure my shell prompt?

A: If you're smart enough to ask this question, you're smart enough to RTFM and find out yourself.

Q: Can I convert an AcmeCorp document into a TeX file using the Bass-o-matic file converter?

A: Try it and see. If you did that, you'd (a) learn the answer, and (b) stop wasting my time.

Q: My {program, configuration, SQL statement} doesn't work

A: This is not a question, and I'm not interested in playing Twenty Questions to pry your actual question out of you — I have better things to do. On seeing something like this, my reaction is normally of one of the following:

- do you have anything else to add to that?
- oh, that's too bad, I hope you get it fixed.
- and this has exactly what to do with me?

Q: I'm having problems with my Windows machine. Can you help?

:

A: Yes. Throw out that Microsoft trash and install an open-source operating system like Linux or BSD.

Note: you *can* ask questions related to Windows machines if they are about a program that does have an official Windows build, or interacts with Windows machines (i.e., Samba). Just don't be surprised by the reply that the problem is with Windows and not the program, because Windows is so broken in general that this is very often the case.

Q: My program doesn't work. I think system facility X is broken.

A: While it is possible that you are the first person to notice an obvious deficiency in system calls and libraries heavily used by hundreds or thousands of people, it is rather more likely that you are utterly clueless. Extraordinary claims require extraordinary evidence; when you make a claim like this one, you must back it up with clear and exhaustive documentation of the failure case.

Q: I'm having problems installing Linux or X. Can you help?

A: No. I'd need hands-on access to your machine to troubleshoot this. Go ask your local Linux user group for hands-on help. (You can find a list of user groups here.)

Note: questions about installing Linux may be appropriate if you're on a forum or mailing list about a particular distribution, and the problem is with *that* distro; or on local user groups forums. In this case, be sure to describe the exact details of the failure. But do careful searching first, with "linux" and *all* suspicious pieces of hardware.

Q: How can I crack root/steal channel-ops privileges/read someone's e-mail?

A: You're a lowlife for wanting to do such things and a moron for asking a hacker to help you.

Good and Bad Questions

Finally, I'm going to illustrate how to ask questions in a smart way by example; pairs of questions about the same problem, one asked in a stupid way and one in a smart way.

Stupid: Where can I find out stuff about the Foonly Flurbamatic?

> This question just begs for "STFW" as a reply.

Smart: I used Google to try to find “Foonly Flurbamatic 2600” on the Web, but I got no useful hits. Can I get a pointer to programming information on this device?

> This one has already STFWed, and sounds like he might have a real problem.

Stupid: I can't get the code from project foo to compile. Why is it broken?

> The querent assumes that somebody else screwed up. Arrogant git...

Smart: The code from project foo doesn't compile under Nulix version 6.2. I've read the FAQ, but it doesn't have anything in it about Nulix-related problems. Here's a transcript of my compilation attempt; is it something I did?

> The querent has specified the environment, read the FAQ, is showing the error, and is not assuming his problems are someone else's fault. This one might be worth some attention.

Stupid: I'm having problems with my motherboard. Can anybody help?

> J. Random Hacker's response to this is likely to be “Right. Do you need burping and diapering, too?” followed by a punch of the delete key.

Smart: I tried X, Y, and Z on the S2464 motherboard. When that didn't work, I tried A, B, and C. Note the curious symptom when I tried C. Obviously the florbish is grommicking, but the results aren't what one might expect. What are the usual causes of grommicking on Athlon MP motherboards? Anybody got ideas for more tests I can run to pin down the problem?

> This person, on the other hand, seems worthy of an answer. He/she has exhibited problem-solving intelligence rather than passively waiting for an answer to drop from on high.

In the last question, notice the subtle but important difference between demanding “Give me an answer” and “Please help me figure out what additional diagnostics I can run to achieve enlightenment.”

In fact, the form of that last question is closely based on a real incident that happened in August 2001 on the linux-kernel mailing list (lkml). I (Eric) was the one asking the question that time. I was seeing mysterious lockups on a Tyan S2462 motherboard. The list members supplied the critical information I needed to solve them.

By asking the question in the way I did, I gave people something to chew on; I made it easy and attractive for them to get involved. I demonstrated respect for my peers' ability and invited them to consult with me as a peer. I also demonstrated respect for the value of their time by telling them the blind alleys I had already run down.

Afterwards, when I thanked everyone and remarked how well the process had worked, an lkml member observed that he thought it had worked not because I'm a “name” on that list, but because I asked the question in the proper form.

Hackers are in some ways a very ruthless meritocracy; I'm certain he was right, and that if I *had* behaved like a sponge I would have been flamed or ignored no matter who I was. His suggestion that I write up the whole incident as instruction to others led directly to the composition of this guide.

If You Can't Get An Answer

If you can't get an answer, please don't take it personally that we don't feel we can help you. Sometimes the members of the asked group may simply not know the answer. No response is not the same as being ignored, though admittedly it's hard to spot the difference from outside.

In general, simply re-posting your question is a bad idea. This will be seen as pointlessly annoying. Have patience: the person with your answer may be in a different time-zone and asleep. Or it may be that your question wasn't well-formed to begin with.

There are other sources of help you can go to, often sources better adapted to a novice's needs.

There are many online and local user groups who are enthusiasts about the software, even though they may never have written any software themselves. These groups often form so that people can help each other and help new users.

There are also plenty of commercial companies you can contract with for help, both large and small. Don't be dismayed at the idea of having to pay for a bit of help! After all, if your car engine blows a head gasket, chances are you would take it to a repair shop and pay to get it fixed. Even if the software didn't cost you anything, you can't expect that support to always come for free.

For popular software like Linux, there are at least 10,000 users per developer. It's just not possible for one person to handle the support calls from over 10,000 users. Remember that even if you have to pay for support, you are still paying much less than if you had to buy the software as well (and support for closed-source software is usually more expensive and less competent than support for open-source software).

How To Answer Questions in a Helpful Way

Be gentle. Problem-related stress can make people seem rude or stupid even when they're not.

Reply to a first offender off-line. There is no need of public humiliation for someone who may have made an honest mistake. A real newbie may not know how to search archives or where the FAQ is stored or posted.

If you don't know for sure, say so! A wrong but authoritative-sounding answer is worse than none at all. Don't point anyone down a wrong path simply because it's fun to sound like an expert. Be humble and honest; set a good example for both the querent and your peers.

If you can't help, don't hinder. Don't make jokes about procedures that could trash the user's setup — the poor sap might interpret these as instructions.

Ask probing questions to elicit more details. If you're good at this, the querent will learn something — and so might you. Try to turn the bad question into a good one; remember we were all newbies once.

While muttering RTFM is sometimes justified when replying to someone who is just a lazy slob, a pointer to documentation (even if it's just a suggestion to google for a key phrase) is better.

If you're going to answer the question at all, give good value. Don't suggest kludgy workarounds when somebody is using the wrong tool or approach. Suggest good tools. Reframe the question.

Answer the actual question! If the querent has been so thorough as to do his or her research and has included in the query that X, Y, Z, A, B, and C have already been tried without good result, it is supremely unhelpful to respond with “Try A or B,” or with a link to something that only says, “Try X, Y, Z, A, B, or C.”.

Help your community learn from the question. When you field a good question, ask yourself “How would the relevant documentation or FAQ have to change so that nobody has to answer this again?” Then send a patch to the document maintainer.

If you did research to answer the question, *demonstrate your skills rather than writing as though you pulled the answer out of your butt.* Answering one good question is like feeding a hungry person one meal, but teaching them research skills by example is showing them how to grow food for a lifetime.

Related Resources

If you need instruction in the basics of how personal computers, Unix, and the Internet work, see The Unix and Internet Fundamentals HOWTO.

When you release software or write patches for software, try to follow the guidelines in the Software Release Practice HOWTO.

Acknowledgements

Evelyn Mitchell contributed some example stupid questions and inspired the “How To Give A Good Answer” section. Mikhail Ramendik contributed some particularly valuable suggestions for improvements.

Before You Ask. Before asking a question a, do the following: Try to find an answer by searching the archives of the forum you plan to post to. Try to find an answer by searching the Web.Â Or it may be that your question wasn't well-formed to begin with. How To Answer Questions in a Helpful Way. Be gentle. Problem-related stress can make people seem rude or stupid even when they're not. Reply to a first offender off-line. There is no need of public humiliation for someone who may have made an honest mistake. A real newbie may not know how to search archives or where the FAQ is stored or posted. If you don't know for sure, say so! A wrong but authoritative-sounding answer is worse than none at all. Asking questions is the key to understanding and gaining the knowledge you need to advance. There is a way to ask questions that can help you gain knowledge and, at the same time, help you look smart and gain respect. Here are the four rules for asking questions the smart way: 1. Follow the Google rule. If you can discreetly Google the concept while the person is speaking and understand it, you may not want to ask it. If you do not understand the information you found on the Internet, donâ€™t be afraid to ask a question. Most likely, others will want to know the same information. The other audience members and even the speaker will respect you for raising your hand and taking the opportunity to clarify something from which everyone can benefit. Here's how asking dumb and naive questions could actually be a more valuable skill. This video is taught by Tim Ferriss, Hope Jahren, Warren Berger, and Jonathon Keats.Â "Very often the dumb question that is sitting right there that no one seems to be asking is the smartest question you can ask," Ferriss says, adding that "not only is it the smartest, most incisive, but if you want to ask it and you're reasonably smart, I guarantee you there are other people who want to ask it but are just. Weâ€™ve asked a stupid question. It happens in both individual and group meetings at work. Hopefully, the employee that we ask is nice and patient, so they answer your question. But, if not, the employee will clearly show you how much disdain your question causes them. I have to be honest. Iâ€™ve been at my current company for a few years now. I went from the newbie engineer who asks a ton of questions to the one who is the receiver of the questions. So Iâ€™ve seen all sides of the spectrum.Â A rule of thumb is if it takes more than 30 seconds to answer the question, then itâ€™s too big and vague. Focus your question to something more specific. â€œWhat is this spreadsheet trying to show?