

# How To Become A Hacker

## Table of Contents

[Why This Document?](#)

[What Is a Hacker?](#)

[The Hacker Attitude](#)

[Basic Hacking Skills](#)

[Status in the Hacker Culture](#)

[The Hacker/Nerd Connection](#)

[Points For Style](#)

[Other Resources](#)

[Frequently Asked Questions](#)

---

## Why This Document?

As editor of the [Jargon File](#) and author of a few other well-known documents of similar nature, I often get email requests from enthusiastic network newbies asking (in effect) "how can I learn to be a wizard hacker?". Oddly enough there don't seem to be any other FAQs or web documents that address this vital question, so here's mine.

If you are reading a snapshot of this document offline, the current version lives at

<http://www.tuxedo.org/~esr/faqs/hacker-howto.html>.

Note: there is a list of [Frequently Asked Questions](#) at the end of this document. Please read these—twice—before mailing me any questions about this document.

Numerous translations of this document are available: [Bulgarian](#), [Chinese \(Simplified\)](#), [Chinese \(Traditional\)](#), [Danish](#), [Dutch](#), [French](#), [German](#), [Hungarian](#), [Indonesian](#), [Japanese](#), [Korean](#), [Portuguese](#), [Russian](#), and [Swedish](#). Note that since this document changes occasionally, they may be out of date to varying degrees.

---

## What Is a Hacker?

The [Jargon File](#) contains a bunch of definitions of the term `hacker', most having to do with technical adeptness and a delight in solving problems and overcoming limits. If you want to know how *to become* a hacker, though, only two are really relevant.

There is a community, a shared culture, of expert programmers and networking wizards that traces its history back through decades to the first time-sharing minicomputers and the earliest ARPAnet experiments. The members of this culture originated the term `hacker'. Hackers built the Internet. Hackers made the Unix operating system what it is today. Hackers run Usenet. Hackers make the World Wide Web work. If you are part of this culture, if you have contributed to it and other people in it know who you are and call you a hacker, you're a hacker.

The hacker mind-set is not confined to this software-hacker culture. There are people who apply the hacker attitude to other things, like electronics or music – actually, you can find it at the highest levels of any science or art. Software hackers recognize these kindred spirits elsewhere and may call them "hackers" too– and some claim that the hacker nature is really independent of the particular medium the hacker works in. But in the rest of this document we will focus on the skills and attitudes of software hackers, and the traditions of the shared culture that originated the term `hacker'.

There is another group of people who loudly call themselves hackers, but aren't. These are people (mainly adolescent males) who get a kick out of breaking into computers and phreaking the phone system. Real hackers call these people 'crackers' and want nothing to do with them. Real hackers mostly think crackers are lazy, irresponsible, and not very bright, and object that being able to break security doesn't make you a hacker any more than being able to hotwire cars makes you an automotive engineer. Unfortunately, many journalists and writers have been fooled into using the word 'hacker' to describe crackers; this irritates real hackers no end.

The basic difference is this: hackers build things, crackers break them.

If you want to be a hacker, keep reading. If you want to be a cracker, go read the [alt.2600](#) newsgroup and get ready to do five to ten in the slammer after finding out you aren't as smart as you think you are. And that's all I'm going to say about crackers.

---

## The Hacker Attitude

Hackers solve problems and build things, and they believe in freedom and voluntary mutual help. To be accepted as a hacker, you have to behave as though you have this kind of attitude yourself. And to behave as though you have the attitude, you have to really believe the attitude.

But if you think of cultivating hacker attitudes as just a way to gain acceptance in the culture, you'll miss the point. Becoming the kind of person who believes these things is important for *you* -- for helping you learn and keeping you motivated. As with all creative arts, the most effective way to become a master is to imitate the mind-set of masters -- not just intellectually but emotionally as well.

Or, as the following modern Zen poem has it:

To follow the path:  
look to the master,  
follow the master,  
walk with the master,  
see through the master,  
become the master.

So, if you want to be a hacker, repeat the following things until you believe them:

---

### 1. The world is full of fascinating problems waiting to be solved.

Being a hacker is lots of fun, but it's a kind of fun that takes lots of effort. The effort takes motivation. Successful athletes get their motivation from a kind of physical delight in making their bodies perform, in pushing themselves past their own physical limits. Similarly, to be a hacker you have to get a basic thrill from solving problems, sharpening your skills, and exercising your intelligence.

If you aren't the kind of person that feels this way naturally, you'll need to become one in order to make it as a hacker. Otherwise you'll find your hacking energy is sapped by distractions like sex, money, and social approval.

(You also have to develop a kind of faith in your own learning capacity -- a belief that even though you may not know all of what you need to solve a problem, if you tackle just a piece of it and learn from that, you'll learn enough to solve the next piece -- and so on, until you're done.)

## 2. No problem should ever have to be solved twice.

Creative brains are a valuable, limited resource. They shouldn't be wasted on re-inventing the wheel when there are so many fascinating new problems waiting out there.

To behave like a hacker, you have to believe that the thinking time of other hackers is precious -- so much so that it's almost a moral duty for you to share information, solve problems and then give the solutions away just so other hackers can solve *new* problems instead of having to perpetually re-address old ones.

(You don't have to believe that you're obligated to give *all* your creative product away, though the hackers that do are the ones that get most respect from other hackers. It's consistent with hacker values to sell enough of it to keep you in food and rent and computers. It's fine to use your hacking skills to support a family or even get rich, as long as you don't forget your loyalty to your art and your fellow hackers while doing it.)

---

## 3. Boredom and drudgery are evil.

Hackers (and creative people in general) should never be bored or have to drudge at stupid repetitive work, because when this happens it means they aren't doing what only they can do-- solve new problems. This wastefulness hurts everybody. Therefore boredom and drudgery are not just unpleasant but actually evil.

To behave like a hacker, you have to believe this enough to want to automate away the boring bits as much as possible, not just for yourself but for everybody else (especially other hackers).

(There is one apparent exception to this. Hackers will sometimes do things that may seem repetitive or boring to an observer as a mind-clearing exercise, or in order to acquire a skill or have some particular kind of experience you can't have otherwise. But this is by choice -- nobody who can think should ever be forced into a situation that bores them.)

---

## 4. Freedom is good.

Hackers are naturally anti-authoritarian. Anyone who can give you orders can stop you from solving whatever problem you're being fascinated by -- and, given the way authoritarian minds work, will generally find some appallingly stupid reason to do so. So the authoritarian attitude has to be fought wherever you find it, lest it smother you and other hackers.

(This isn't the same as fighting all authority. Children need to be guided and criminals restrained. A hacker may agree to accept some kinds of authority in order to get something he wants more than the time he spends following orders. But that's a limited, conscious bargain; the kind of personal surrender authoritarians want is not on offer.)

Authoritarians thrive on censorship and secrecy. And they distrust voluntary cooperation and information-sharing -- they only like 'cooperation' that they control. So to behave like a hacker, you have to develop an instinctive hostility to censorship, secrecy, and the use of force or deception to compel responsible adults. And you have to be willing to act on that belief.

---

## 5. Attitude is no substitute for competence.

To be a hacker, you have to develop some of these attitudes. But copping an attitude alone won't make you a hacker, any more than it will make you a champion athlete or a rock star. Becoming a hacker will take intelligence, practice, dedication, and hard work.

Therefore, you have to learn to distrust attitude and respect competence of every kind. Hackers won't let posers waste their time, but they worship competence -- especially competence at hacking, but competence at anything is good. Competence at demanding skills that few can master is especially good, and competence at demanding skills that involve mental acuteness, craft, and concentration is best.

If you revere competence, you'll enjoy developing it in yourself -- the hard work and dedication will become a kind of intense play rather than drudgery. And that's vital to becoming a hacker.

---

## Basic Hacking Skills

The hacker attitude is vital, but skills are even more vital. Attitude is no substitute for competence, and there's a certain basic toolkit of skills which you have to have before any hacker will dream of calling you one.

This toolkit changes slowly over time as technology creates new skills and makes old ones obsolete. For example, it used to include programming in machine language, and didn't until recently involve HTML. But right now it pretty clearly includes the following:

---

### 1. Learn how to program.

This, of course, is the fundamental hacking skill. If you don't know any computer languages, I recommend starting with Python. It is cleanly designed, well documented, and relatively kind to beginners. Despite being a good first language, it is not just a toy; it is very powerful and flexible and well suited for large projects. I have written a more detailed [evaluation of Python](#). Good [tutorials](#) are available at the [Python web site](#).

Java is also a good language for learning to program in. It is more difficult than Python, but produces faster code than Python. I think it makes an excellent second language.

But be aware that you won't reach the skill level of a hacker or even merely a programmer if you only know one or two languages -- you need to learn how to think about programming problems in a general way, independent of any one language. To be a real hacker, you need to get to the point where you can learn a new language in days by relating what's in the manual to what you already know. This means you should learn several very different languages.

If you get into serious programming, you will have to learn C, the core language of Unix. C++ is very closely related to C; if you know one, learning the other will not be difficult. Neither language is a good one to try learning as your first, however.

Other languages of particular importance to hackers include [Perl](#) and [LISP](#). Perl is worth learning for practical reasons; it's very widely used for active web pages and system administration, so that even if you never write Perl you should learn to read it. LISP is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use LISP itself a lot.

It's best, actually, to learn all five of these (Python, Java, C/C++, Perl, and LISP). Besides being the most important hacking languages, they represent very different approaches to programming, and each will educate you in valuable ways.

I can't give complete instructions on how to learn to program here-- it's a complex skill. But I can tell you that books and courses won't do it (many, maybe *most* of the best hackers are self-taught). You can learn language features -- bits of knowledge -- from books, but the mind-set that makes that knowledge into living skill can be learned only by practice and apprenticeship. What will do it is (a) *reading code* and (b) *writing code*.

Learning to program is like learning to write good natural language. The best way to do it is to read some stuff written by masters of the form, write some things yourself, read a lot more, write a little more, read a lot more, write some more ... and repeat until your writing begins to develop the kind of strength and economy you see in your models.

Finding good code to read used to be hard, because there were few large programs available in source for fledgeling hackers to read and tinker with. This has changed dramatically; open-source software, programming tools, and operating systems (all built by hackers) are now widely available. Which brings me neatly to our next topic...

---

## 2. Get one of the open-source Unixes and learn to use and run it.

I'm assuming you have a personal computer or can get access to one (these kids today have it so easy :-)). The single most important step any newbie can take toward acquiring hacker skills is to get a copy of Linux or one of the BSD Unixes, install it on a personal machine, and run it.

Yes, there are other operating systems in the world besides Unix. But they're distributed in binary-- you can't read the code, and you can't modify it. Trying to learn to hack on a DOS or Windows machine or under MacOS is like trying to learn to dance while wearing a body cast.

Besides, Unix is the operating system of the Internet. While you can learn to use the Internet without knowing Unix, you can't be an Internet hacker without understanding Unix. For this reason, the hacker culture today is pretty strongly Unix-centered. (This wasn't always true, and some old-time hackers still aren't happy about it, but the symbiosis between Unix and the Internet has become strong enough that even Microsoft's muscle doesn't seem able to seriously dent it.)

So, bring up a Unix -- I like Linux myself but there are other ways (and yes, you *can* run both Linux and DOS/Windows on the same machine). Learn it. Run it. Tinker with it. Talk to the Internet with it. Read the code. Modify the code. You'll get better programming tools (including C, LISP, Python, and Perl) than any Microsoft operating system can dream of, you'll have fun, and you'll soak up more knowledge than you realize you're learning until you look back on it as a master hacker.

For more about learning Unix, see [The Loginataka](#).

To get your hands on a Linux, see the [Where can I get Linux](#).

You can find BSD Unix help and resources at [www.bsd.org](http://www.bsd.org).

I have written a primer on the [basics of Unix and the Internet](#).

(Note: I don't really recommend installing either Linux or BSD as a solo project if you're a newbie. For Linux, find a local Linux user's group and ask for help; or contact the [Open Projects Network](#). LISC maintains [IRC channels](#) where you can get help.)

---

## 3. Learn how to use the World Wide Web and write HTML.

Most of the things the hacker culture has built do their work out of sight, helping run factories and offices and universities without any obvious impact on how non-hackers live. The Web is the one big exception, the huge shiny hacker toy that even *politicians* admit is changing the world. For this reason alone (and a lot of other good ones as well) you need to learn how to work the Web.

This doesn't just mean learning how to drive a browser (anyone can do that), but learning how to write HTML, the Web's markup language. If you don't know how to program, writing HTML will teach you some mental habits that will help you learn. So build a home page. (There are good beginner tutorials on the Web;[here's one](#).)

But just having a home page isn't anywhere near good enough to make you a hacker. The Web is full of home pages. Most of them are pointless, zero-content sludge -- very snazzy-looking sludge, mind you, but sludge all the same (for more on this see [The HTML Hell Page](#)).

To be worthwhile, your page must have *content* -- it must be interesting and/or useful to other hackers. And that brings us to the next topic...

---

## Status in the Hacker Culture

Like most cultures without a money economy, hackerdom runs on reputation. You're trying to solve interesting problems, but how interesting they are, and whether your solutions are really good, is something that only your technical peers or superiors are normally equipped to judge.

Accordingly, when you play the hacker game, you learn to keep score primarily by what other hackers think of your skill (this is why you aren't really a hacker until other hackers consistently call you one). This fact is obscured by the image of hacking as solitary work; also by a hacker-cultural taboo (now gradually decaying but still potent) against admitting that ego or external validation are involved in one's motivation at all.

Specifically, hackerdom is what anthropologists call a *gift culture*. You gain status and reputation in it not by dominating other people, nor by being beautiful, nor by having things other people want, but rather by giving things away. Specifically, by giving away your time, your creativity, and the results of your skill.

There are basically five kinds of things you can do to be respected by hackers:

---

### 1. Write open-source software

The first (the most central and most traditional) is to write programs that other hackers think are fun or useful, and give the program sources to the whole hacker culture to use.

(We used to call these works "free software", but this confused too many people who weren't sure exactly what "free" was supposed to mean. Many of us now prefer the term "[open-source](#)" software).

Hackerdom's most revered demigods are people who have written large, capable programs that met a widespread need and given them away, so that now everyone uses them.

---

### 2. Help test and debug open-source software

They also serve who stand and debug open-source software. In this imperfect world, we will inevitably spend most of our software development time in the debugging phase. That's why any open-source author who's thinking will tell you that good beta-testers (who know how to describe symptoms clearly, localize problems well, can tolerate bugs in a quickie release, and are willing to apply a few simple diagnostic routines) are worth their weight in rubies. Even one of these can make the difference between a debugging phase that's a protracted, exhausting nightmare and one that's merely a salutary nuisance.

If you're a newbie, try to find a program under development that you're interested in and be a good beta-tester. There's a natural progression from helping test programs to helping debug them to helping modify them. You'll learn a lot this way, and generate good karma with people who will help you later on.

---

### 3. Publish useful information

Another good thing is to collect and filter useful and interesting information into web pages or documents like Frequently Asked Questions (FAQ) lists, and make those generally available.

Maintainers of major technical FAQs get almost as much respect as open-source authors.

---

### 4. Help keep the infrastructure working

The hacker culture (and the engineering development of the Internet, for that matter) is run by volunteers. There's a lot of necessary but unglamorous work that needs done to keep it going -- administering mailing lists, moderating newsgroups, maintaining large software archive sites, developing RFCs and other technical standards.

People who do this sort of thing well get a lot of respect, because everybody knows these jobs are huge time sinks and not as much fun as playing with code. Doing them shows dedication.

---

### 5. Serve the hacker culture itself

Finally, you can serve and propagate the culture itself (by, for example, writing an accurate primer on how to become a hacker :-)). This is not something you'll be positioned to do until you've been around for while and become well-known for one of the first four things.

The hacker culture doesn't have leaders, exactly, but it does have culture heroes and tribal elders and historians and spokespeople. When you've been in the trenches long enough, you may grow into one of these. Beware: hackers distrust blatant ego in their tribal elders, so visibly reaching for this kind of fame is dangerous. Rather than striving for it, you have to sort of position yourself so it drops in your lap, and then be modest and gracious about your status.

---

## The Hacker/Nerd Connection

Contrary to popular myth, you don't have to be a nerd to be a hacker. It does help, however, and many hackers are in fact nerds. Being a social outcast helps you stay concentrated on the really important things, like thinking and hacking.

For this reason, many hackers have adopted the label `nerd' and even use the harsher term `geek' as a badge of pride-- it's a way of declaring their independence from normal social expectations. See [The Geek Page](#) for extensive discussion.

If you can manage to concentrate enough on hacking to be good at it and still have a life, that's fine. This is a lot easier today than it was when I was a newbie in the 1970s; mainstream culture is much friendlier to techno-nerds now. There are even growing numbers of people who realize that hackers are often high-quality lover and spouse material.

If you're attracted to hacking because you don't have a life, that's OK too-- at least you won't have trouble concentrating. Maybe you'll get a life later on.

---

## Points For Style

Again, to be a hacker, you have to enter the hacker mindset. There are some things you can do when you're not at a computer that seem to help. They're not substitutes for hacking (nothing is) but many hackers do them, and feel that they connect in some basic way with the essence of hacking.

- Learn to write your native language well. Though it's a common stereotype that programmers can't write, a surprising number of hackers (including all the best ones I know of) are able writers.
- Read science fiction. Go to science fiction conventions (a good way to meet hackers and proto-hackers).
- Study Zen, and/or take up martial arts. (The mental discipline seems similar in important ways.)
- Develop an analytical ear for music. Learn to appreciate peculiar kinds of music. Learn to play some musical instrument well, or how to sing.
- Develop your appreciation of puns and wordplay.

The more of these things you already do, the more likely it is that you are natural hacker material. Why these things in particular is not completely clear, but they're connected with a mix of left and right-brain skills that seems to be important (hackers need to be able to both reason logically and step outside the apparent logic of a problem at a moment's notice).

Finally, a few things *not* to do.

- Don't use a silly, grandiose user ID or screen name.
- Don't get in flame wars on Usenet (or anywhere else).
- Don't call yourself a `cyberpunk', and don't waste your time on anybody who does.
- Don't post or email writing that's full of spelling errors and bad grammar.

The only reputation you'll make doing any of these things is as a twit. Hackers have long memories -- it could take you years to live your early blunders down enough to be accepted.

The problem with screen names or handles deserves some amplification. Concealing your identity behind a handle is a juvenile and silly behavior characteristic of crackers, warez d00dz, and other lower life forms. Hackers don't do this; they're proud of what they do and want it associated with their *real* names. So if you have a handle, drop it. In the hacker culture it will only mark you as a loser.

---

## Other Resources

Peter Seebach maintains an excellent [Hacker FAQ](#) for managers who don't understand how to deal with hackers. If Peter's site doesn't respond, the following [Excite search](#) should find a copy.

I have also written [A Brief History Of Hackerdom](#).

I have written a paper, [The Cathedral and the Bazaar](#), which explains a lot about how the Linux and open-source cultures work. I have addressed this topic even more directly in its sequel [Homesteading the Noosphere](#).

Rick Moen has written an excellent document on [how to run a Linux user group](#).

Rick Moen and I have collaborated on another document on [How To Ask Smart Questions](#). This will help you seek assistance in a way that makes it more likely that you will actually get it.

---

## Frequently Asked Questions

Q: [Will you teach me how to hack?](#)

Q: [How can I get started, then?](#)

Q: [When do you have to start? Is it too late for me to learn?](#)

Q: [How long will it take me to learn to hack?](#)

Q: [Are Visual Basic or Delphi good languages to start with?](#)

Q: [Would you help me to crack a system, or teach me how to crack?](#)

Q: [How can I get the password for someone else's account?](#)

Q: [How can I break into/read/monitor someone else's email?](#)

Q: [How can I steal channel op privileges on IRC?](#)

Q: [I've been cracked. Will you help me fend off further attacks?](#)

Q: [I'm having problems with my Windows software. Will you help me?](#)

Q: [Where can I find some real hackers to talk with?](#)

Q: [Can you recommend useful books about hacking-related subjects?](#)

Q: [Do I need to be good at math to become a hacker?](#)

Q: [What language should I learn first?](#)

Q: [What kind of hardware do I need?](#)

Q: [Do I need to hate and bash Microsoft?](#)

Q: [But won't open-source software leave programmers unable to make a living?](#)

Q: [How can I get started? Where can I get a free Unix?](#)

### Q: Will you teach me how to hack?

A: Since first publishing this page, I've gotten several requests a week (often several a day) from people to "teach me all about hacking". Unfortunately, I don't have the time or energy to do this; my own hacking projects, and traveling as an open-source advocate, take up 110% of my time.

Even if I did, hacking is an attitude and skill you basically have to teach yourself. You'll find that while real hackers want to help you, they won't respect you if you beg to be spoon-fed everything they know.

Learn a few things first. Show that you're trying, that you're capable of learning on your own. Then go to the hackers you meet with specific questions.

If you do email a hacker asking for advice, here are two things to know up front. First, we've found that people who are lazy or careless in their writing are usually too lazy and careless in their thinking to make good hackers – so take care to spell correctly, and use good grammar and punctuation, otherwise you'll probably be ignored. Secondly, don't *dare* ask for a reply to an ISP account that's different from the account you're sending from; we find people who do that are usually thieves using stolen accounts, and we have no interest in rewarding or assisting thievery.

### Q: How can I get started, then?

A: The best way for you to get started would probably be to go to a LUG (Linux user group) meeting. You can find such groups on the [LDP General Linux Information Page](#); there is probably one near you, possibly associated with a college or university. LUG members will probably give you a Linux if you ask, and will certainly help you install one and get started.

### Q: When do you have to start? Is it too late for me to learn?

A: Any age at which you are motivated to start is a good age. Most people seem to get interested between ages 15 and 20, but I know of exceptions in both directions.

### Q: How long will it take me to learn to hack?

**A:** That depends on how talented you are and how hard you work at it. Most people can acquire a respectable skill set in eighteen months to two years, if they concentrate. Don't think it ends there, though; if you are a real hacker, you will spend the rest of your life learning and perfecting your craft.

**Q: Are Visual Basic or Delphi good languages to start with?**

**A:** No, because they're not portable. There are no open-source implementations of these languages, so you'd be locked into only those platforms the vendor chooses to support. Accepting that kind of monopoly situation is not the hacker way.

Visual Basic is especially awful. The fact that it's a proprietary Microsoft language is enough to disqualify it, and like other Basics it's a poorly-designed language that will teach you bad programming habits.

One of those bad habits is becoming dependent on a single vendor's libraries, widgets, and development tools. In general, any language that isn't supported under at least Linux or one of the BSDs, and/or at least three different vendors' operating systems, is a poor one to learn to hack in.

**Q: Would you help me to crack a system, or teach me how to crack?**

**A:** No. Anyone who can still ask such a question after reading this FAQ is too stupid to be educable even if I had the time for tutoring. Any emailed requests of this kind that I get will be ignored or answered with extreme rudeness.

**Q: How can I get the password for someone else's account?**

**A:** This is cracking. Go away, idiot.

**Q: How can I break into/read/monitor someone else's email?**

**A:** This is cracking. Get lost, moron.

**Q: How can I steal channel op privileges on IRC?**

**A:** This is cracking. Begone, cretin.

**Q: I've been cracked. Will you help me fend off further attacks?**

**A:** No. Every time I've been asked this question so far, it's been from some poor sap running Microsoft Windows. It is not possible to effectively secure Windows systems against crack attacks; the code and architecture simply have too many flaws, which makes securing Windows like trying to bail out a boat with a sieve. The only reliable prevention starts with switching to Linux or some other operating system that is designed to at least be capable of security.

**Q: I'm having problems with my Windows software. Will you help me?**

**A:** Yes. Go to a DOS prompt and type "format c:". Any problems you are experiencing will cease within a few minutes.

**Q: Where can I find some real hackers to talk with?**

**A:** The best way is to find a Unix or Linux user's group local to you and go to their meetings (you can find links to several lists of user groups on the [LDP](#) site at Metalab).

(I used to say here that you wouldn't find any real hackers on IRC, but I'm given to understand this is changing. Apparently some real hacker communities, attached to things like GIMP and Perl, have IRC channels now.)

**Q: Can you recommend useful books about hacking-related subjects?**

**A:** I maintain a [Linux Reading List HOWTO](#) that you may find helpful. The [Loginataka](#) may also be interesting.

For an introduction to Python, see the [introductory materials](#) on the Python site.

### **Q: Do I need to be good at math to become a hacker?**

**A:** No. While you do need to be able to think logically and follow chains of exact reasoning, hacking uses very little formal mathematics or arithmetic.

In particular, you won't need calculus or analysis (we leave that stuff to the electrical engineers :-)). Some grounding in finite mathematics (including Boolean algebra, finite-set theory, combinatorics, and graph theory) can be helpful.

### **Q: What language should I learn first?**

**A:** HTML, if you don't already know it. There are a lot of glossy, hype-intensive *bad* HTML books out there, and distressingly few good ones. The one I like best is [HTML: The Definitive Guide](#).

But HTML is not a full programming language. When you're ready to start programming, I would recommend starting with [Python](#). You will hear a lot of people recommending Perl, and Perl is still more popular than Python, but it's harder to learn and (in my opinion) less well designed.

C is really important, but it's also much more difficult than either Python or Perl. Don't try to learn it first.

Windows users, do *not* settle for Visual Basic. It will teach you bad habits, and it's not portable off Windows. Avoid.

### **Q: What kind of hardware do I need?**

**A:** It used to be that personal computers were rather underpowered and memory-poor, enough so that they placed artificial limits on a hacker's learning process. This stopped being true some time ago; any machine from an Intel 486DX50 up is more than powerful enough for development work, X, and Internet communications, and the smallest disks you can buy today are plenty big enough.

The important thing in choosing a machine on which to learn is whether its hardware is Linux-compatible (or BSD-compatible, should you choose to go that route). Again, this will be true for most modern machines; the only sticky areas are modems and printers; some machines have Windows-specific hardware that won't work with Linux.

There's a FAQ on hardware compatibility; the latest version is [here](#).

### **Q: Do I need to hate and bash Microsoft?**

**A:** No, you don't. Not that Microsoft isn't loathsome, but there was a hacker culture long before Microsoft and there will still be one long after Microsoft is history. Any energy you spend hating Microsoft would be better spent on loving your craft. Write good code -- that will bash Microsoft quite sufficiently without polluting your karma.

### **Q: But won't open-source software leave programmers unable to make a living?**

**A:** This seems unlikely -- so far, the open-source software industry seems to be creating jobs rather than taking them away. If having a program written is a net economic gain over not having it written, a programmer will get paid whether or not the program is going to be open-source after it's done. And, no matter how much "free" software gets written, there always seems to be more demand for new and customized applications. I've written more about this at the [Open Source](#) pages.

### **Q: How can I get started? Where can I get a free Unix?**

**A:** Elsewhere on this page I include pointers to where to get the most commonly used free Unix. To be a hacker you need motivation and initiative and the ability to educate yourself. Start now...