**Survival guide for Unix newbies**

[[article index](http://matt.might.net/articles/)] [[email me](mailto:matt-blog@might.net)] [[@mattmight](http://twitter.com/mattmight)] [[rss](http://matt.might.net/articles/feed.rss)]

As a professor, I worry that the upcoming generation of programmers is [missing out](http://matt.might.net/articles/what-cs-majors-should-know/) on the Unix experience, and with it, [the power it grants](http://www.amazon.com/gp/product/0596003307/ref=as_li_ss_tl?ie=UTF8&tag=ucmbread-20&linkCode=as2&camp=217145&creative=399369&creativeASIN=0596003307).

Modern computing environments tend to favor form over function: the primary objective in their design is ease of use for non-experts.

Unix is a naked celebration of function over form. The premium is on control, efficiency and flexibility. Its audience is the power user.

The origin of Unix's power is an organic design philosophy that emphasizes linguistic abstraction and composition.

In Unix, a word is worth a thousand mouse clicks.

I've written the short guide below as an introduction for those just getting started with Unix or Linux. (I'll write follow-up articles covering intermediate and advanced Unix.)

Please feel free to forward this series to a student, friend, partner or spouse that needs a little help getting started.

**Update**: After this, see the companion post on [settling into Unix](http://matt.might.net/articles/settling-into-unix/).

**What is computing with Unix?**

Unix is a family of operating systems and environments that exploits the power of linguistic abstraction and composition to orchestrate tasks.

Unix users spend a lot of time at the command line.

The command line (often called the console, the shell or the terminal) looks something like this:

$ \_

When you type in a command like ls, it will give you feedback:

$ ls

Documents README.txt

$ \_

In this case, the ls command listed the files in the current directory.

**Getting access to Unix**

It was once a daunting task to get access to or install Unix. Fortunately, it's no longer a hassle.

These days, the two most accessible flavors of Unix are Apple's OS X (based on the BSD family of Unix) and Ubuntu Linux.

**The easy option: OS X**

OS X is a good way for many users to tip-toe into Unix.

What you're buying is a Unix machine that "just works."

To turn it into a complete Unix system, you'll need to install Apple's free Xcode development suite (available from the Mac App Store) and the (free) X11 server.

Xcode includes the compilers and interpreters you'll need to compile and run many open source applications.

You'll need the X11 server to run traditional graphical applications for Unix.

**The DIY option: Linux**

When I was in high school, before CD burning was widely available and when internet connections were slow, the best way to get Linux was to buy a book with an installation CD.

(Given that there wasn't much in the way of online resources for Linux back then, it was good to have a book that walked through the installation process and explained all the features, too.)

A [quick search of amazon](http://www.amazon.com/gp/redirect.html?ie=UTF8&location=http%3A%2F%2Fwww.amazon.com%2Fs%3Fie%3DUTF8%26x%3D0%26ref_%3Dnb_sb_noss%26y%3D0%26field-keywords%3Dlinux%26url%3Dsearch-alias%253Dstripbooks&tag=ucmbread-20&linkCode=ur2&camp=1789&creative=390957) shows no shortages of such books today.

These days, there are plenty of free options:

* If you're a CS major, you should have access to a Unix shell account at your school that you can acces with ssh. (See [details on ssh](http://matt.might.net/articles/basic-unix/#ssh) below.)
* You can set up your computer to dual boot into either your preferred OS or Linux. Macs come with [Boot Camp](http://www.apple.com/support/bootcamp/) to make this easy. If you take the dual boot route with a Windows PC, FIRST BACK UP YOUR DATA. There is a chance you'll wipe out your files by accident. I recommend adding a second internal hard drive if you want to set up a dual boot with Linux for the first time.
* You can burn a CD that lets you boot into or install Linux. I recommend [Ubuntu Linux](http://www.ubuntu.com/) for this.
* You can also create a USB thumb drive that boots into Linux. I recommend [Ubuntu Linux](http://www.ubuntu.com/) for this as well.
* You can run Linux inside your existing OS as a virtual machine. [VirtualBox](http://www.virtualbox.org/) is a freely available virtualization tool. (I personally use [VMware Fusion](http://www.amazon.com/gp/product/B002Q72JB8/ref=as_li_ss_tl?ie=UTF8&tag=ucmbread-20&linkCode=as2&camp=1789&creative=390957&creativeASIN=B002Q72JB8) to run Windows on my Mac for those rare instances where I need to see what a page looks like in an old version of IE.)
* You can run [cygwin](http://www.cygwin.com/) on Windows to get a "Unix-like" environment. (This isn't quite the real deal.)
* You can buy a virtualized Linux server with remote access from a service like [linode.com](http://www.linode.com/?r=bf5d4e7c8a1af61855b5227279a6744c3bde8a8a). (might.net runs on a linode.)

**Getting to the command line**

Once you're into Unix, you'll want to get to the command line.

On Linux, you can press CTRL+ALT+F1 to get to a raw terminal, and CTRL+ALT+F7 to get back to the window system. (In fact, CTRL+ALT+F1-F6 will get you different terminal on most systems, and of course, you can configure this.)

In most popular flavors of Linux, there is a way to open a graphical terminal in a window.

On a Mac, it's Applications > Utilities > Terminal.

**The filesystem: ls and cd**

If you find yourself at a command prompt, run ls to list the files in the current directory.

You might see something like this:

$ ls

Documents README.txt

$ \_

With ls /, you can list the contents of the root directory.

You'll notice several [directories that are common to almost all Unices](http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard#Directory_structure):

* /bin, standard executables.
* /sbin, important system executables.
* /etc, system configuration files.
* /var, frequently updated files like logs.
* /root, home directory of the administrator account.
* /home, a user account home directories. (/Users on a Mac.)
* /opt, package-manager installed files.
* /tmp, temporary files; usually wiped between boots.
* /usr, user applications and utilities.

To change to a different directory, use the command cd.

For example:

* cd Documents to change to the Documents sub-folder.
* cd .. to change to the parent directory.
* cd to change to your home directory.

**Paths**

To print the path of the current working directory, use pwd:

$ pwd

/home/matt

$ \_

On Unix, a path is a description of a file's location.

For example, /home/matt/README is a path to a file called README contained within the home directory for the user matt.

Notice that forward slashes (/) separate components of a path.

For example:

$ pwd

/home/matt

$ cd ..

$ pwd

/home

$ cd matt

$ pwd

/home/matt

$ \_

**Symbolic links**

Unix filesystems make use of aliases for files known as symbolic links (symlinks).

(There are also *hard* links, in which there are two true, indistingushible references to the same file, but these are less commonly used.)

A symbolic link to a file is ordinarily treated identically to that file.

The command ln -s creates sym links.

The command ls -l can reveal where a symlink points.

For example:

$ ls

bar foo

$ ln -s bar baz

$ ls -l

total 8

-rw-r--r-- 1 matt staff 0 Jan 8 09:57 bar

lrwxr-xr-x 1 matt staff 3 Jan 8 09:58 baz -> bar

-rw-r--r-- 1 matt staff 0 Jan 8 09:57 foo

$ \_

**Working with text: cat, less, emacs and vi**

Suppose you want to look at a text file within the current directory.

A common way to take a glance at a text file is with the cat command:

$ cat README.txt

\* A README file for my home directory.

Documents contains my files.

$ \_

(The command cat is actually meant to do more powerful things than just look at files, but that's what it's most commonly used for.)

If the file is too long to fit on a screen, use less instead of cat. (To quit less, press q.)

You're going to spend a lot of time editing text, so you'll want to learn one (or both) of the powerful Unix text editors: emacs and vim.

Both have a tutorial mode for teaching the essentials.

**Help yourself: man up**

If you don't know what a command does, use the command man *command*.

This will bring up the man(ual) page for *command*, which will document how the command operates; it frequently provides examples of usage.

Press q to quit the man page viewer.

The apropos can help you find the command you're looking for if you don't know it's name. It will search a database of command *descriptions*.

Some tools use an alternate documentation system called info

To look up documentation using info, try info *topic*.

For more info on info, run info info.

**Search for it: grep and find**

If you're looking for text within a file or a set of files, grep is the right tool.

The command grep *pattern* *file ...* will search the specified files for the pattern.

The pattern language used throughout Unix is called *regular expressions*, and there are many good introductions to these, including the man page for grep.

To list all files under the working directory (and its subdirectories), try the command find.

**Pipes and redirection**

Nearly every command in Unix makes use of a convention to have a "standard input" (also called stdin or STDIN) and standard output (also called stdout or STDOUT).

There is also a "standard error" (stderr or STDERR) output that is, by convention, reserved for error messages.

Many techniques in Unix rely on redirecting these channels.

If you want to dump the standard output into a file, use *command* > *file*.

For example:

$ pwd > pwd.txt

$ ls

pwd.txt

$ cat pwd.txt

/home/matt

$ \_

One of the most useful capabilities of Unix is the ability to redirect the STDOUT of one command into the STDIN of another.

To do this, you'll want to use a "pipe"; for example:

$ find . | grep READ

./Desktop/READINGLIST.txt

./README.txt

$ \_

This command found a file named READINGLIST.txt in the Desktop subdirectory, and a file named README in the current one.

It's customary for Unix programs to read from STDIN when no input file is specified, which is what grep is doing here.

It is also possible to send the contents of a file into the STDIN of a command using *command* < *file*.

**Permissions: chmod, chown and chgrp**

In Unix, every file and directory has an owner and a group.

Every user on a Unix machine can belong to one or more groups.

Every file also has three sets of permissions: what the owner can do, what the group can do and what anyone can do.

To see the owner, group and permissions associated with a file run ls -l. You'll see something like:

$ ls -l

drwxr-xr-x 4 matt staff 136 Jun 23 2010 Documents

drwxr-xr-x 143 matt staff 4862 Dec 30 2009 Desktop

-rw-r--r-- 1 matt staff 4 Feb 7 11:16 README.txt

$ \_

The first column tells you about the permissions on the file.

The third and fourth tell you the owner and group respectively.

The very first character in the permissions column tells you what kind of file it is. A - means it's a regular file. A d means it's a directory.

The next nine characters come in three classes of three characters each. The three classes are owner permissions, group permissions and world permissions.

Inside a permission class, r means that class can read the file; w means that class can write the file; x means that class can execute the file.

If a file is a directory, x grants the permission to access inside the directory, while r grants permission to list its contents.

**Package managers**

Modern Unixes have package managers to that download install (free) software for you.

On a Mac, [MacPorts](http://www.macports.com/) is a popular package-management system, and [Homebrew](http://mxcl.github.com/homebrew/) is gaining in popularity.

On Ubuntu, apt is the standard package manager, with both a command-line and graphical interface available.

It's a good idea to familiarize yourself with your package manager.

**Remote access: ssh**

If you have more than one machine, or you need remote access to another machine, ssh is a powerful "secure shell" utility.

ssh can grant you console access on a remote machine, while safely encrypting the session.

It is worth setting up passwordless key-based authentication using [ssh-keygen](http://en.wikipedia.org/wiki/Ssh-keygen).

Using ~/.ssh/config, many options, including aliases and per-host private keys, are available. (Try man ssh\_config).

To ssh into another computer, use ssh *user*@*address* where *user* is the user name of your account on the other machine, and *address* is host name or IP address of the other computer.

Windows users can use [PuTTY](http://www.chiark.greenend.org.uk/~sgtatham/putty/) to connect to another computer over ssh.

**What's next?**

Wikipedia has a [list of Unix utilities](http://en.wikipedia.org/wiki/List_of_Unix_utilities) to peruse.

You can also move onto my guide to [settling into Unix](http://matt.might.net/articles/settling-into-unix/).

**Good books**

* [Unix Power Tools](http://www.amazon.com/gp/product/0596003307/ref=as_li_ss_tl?ie=UTF8&tag=ucmbread-20&linkCode=as2&camp=217145&creative=399369&creativeASIN=0596003307):
* The [UNIX and Linux System Administration Handbook](http://www.amazon.com/gp/product/0131480057/ref=as_li_ss_tl?ie=UTF8&tag=ucmbread-20&linkCode=as2&camp=217145&creative=399369&creativeASIN=0131480057):
* [Classic Shell Scripting](http://www.amazon.com/gp/product/0596005954/ref=as_li_ss_tl?ie=UTF8&tag=ucmbread-20&linkCode=as2&camp=217145&creative=399369&creativeASIN=0596005954):

**Related posts**

* [Learn Perl by experiment](http://matt.might.net/articles/perl-by-example/)
* [A quick overview of programming with bash](http://matt.might.net/articles/bash-by-example/)
* [A short introduction to make](http://matt.might.net/articles/intro-to-make/)
* [SSH hacks](http://matt.might.net/articles/ssh-hacks/)
* [Standalone lexers with lex: synopsis, examples, and pitfalls](http://matt.might.net/articles/standalone-lexers-with-lex/)
* [Sculpting text with regex, grep, sed and awk](http://matt.might.net/articles/sculpting-text/)
* [Relational shell programming](http://matt.might.net/articles/sql-in-the-shell/)
* [Settling into Unix](http://matt.might.net/articles/settling-into-unix/)
* [Console productivity hack: Exploiting task frequency](http://matt.might.net/articles/console-hacks-exploiting-frequency/)
* [HOWTO: Word, Excel and PowerPoint without MS Office](http://matt.might.net/articles/how-to-read-and-create-microsoft-word-documents-excel-spreadsheets-powerpoint-presentations-without-microsoft-office/)
* [Tips, tricks and tools for Linux and Unix](http://matt.might.net/articles/best-tools-for-using-and-learning-linux-and-unix/)