

What is the Command Line?

Created by Brennen Bearnes

```
brennen@desiderata 16:24:43 /home/brennen ★ cowsay -f gnu GET YOU A SHELL
< GET YOU A SHELL >
-----
      \   ^__^
      (oo)\_____
         (__)\       )\/\
            ||----w |
            ||     ||

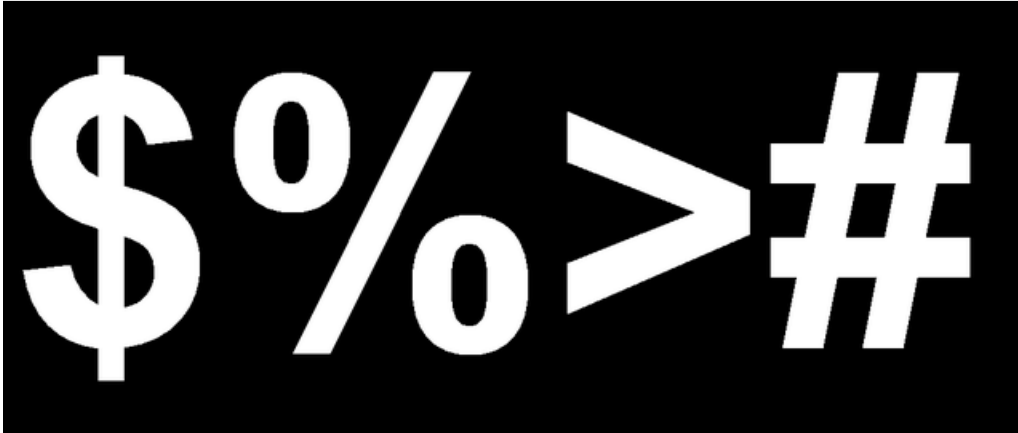
brennen@desiderata 16:25:00 /home/brennen ★
```

Last updated on 2018-08-22 03:45:32 PM UTC

Guide Contents

Guide Contents	2
Overview	3
Ancient History	4
What's a Terminal?	8
Why terminals?	9
Getting a Terminal on Your Raspberry Pi	11
Use the HDMI Console	11
Using a Console Cable	12
Run a Terminal Emulator from Your Desktop	13
Connect to Your Pi Over the Network with SSH	14
At the Prompt	16

Overview



OK now you have a Linux computer (<https://adafru.it/jDZ>), and you are ready to make it do your bidding! We're going to cover how to send commands to your Linux machine using the **command line** - you know, a line where you send commands!

In order to start typing those commands (list files! print this one out! open the web browser!) you will use a **shell** - a shell is a little like the Finder on Macs, or the desktop Explorer on windows, but is not graphical - it uses only TEXT.

A shell is a program with a **command-line interface** which you talk to with a **terminal**. Terminals are software or hardware that let you send text back and forth to a computer. Shells offer a very specific *kind* of command line, but what we mean in general by that term is an interface where a user types commands and the computer responds.

The **shell** provides the command line dialect used to navigate the files on a machine, run programs, and generally stitch things together to solve problems.

You may be a little puzzled about the way "command line", "terminal" and "shell" work together - don't worry! Although *technically* they are different parts of a whole, they are often used interchangeably, as in "open up a terminal", "get to the command line", or "start up a shell" - these are all basically the same thing.

My shell looks like this:

```
brennen@desiderata 10:30:54 /var/www (master) ★
```

In this guide, we'll talk a little about where the command line comes from and why it's still so important, explain how to get a shell on the Raspberry Pi (and other systems), and begin to outline how it can be useful in your own work.

Ancient History

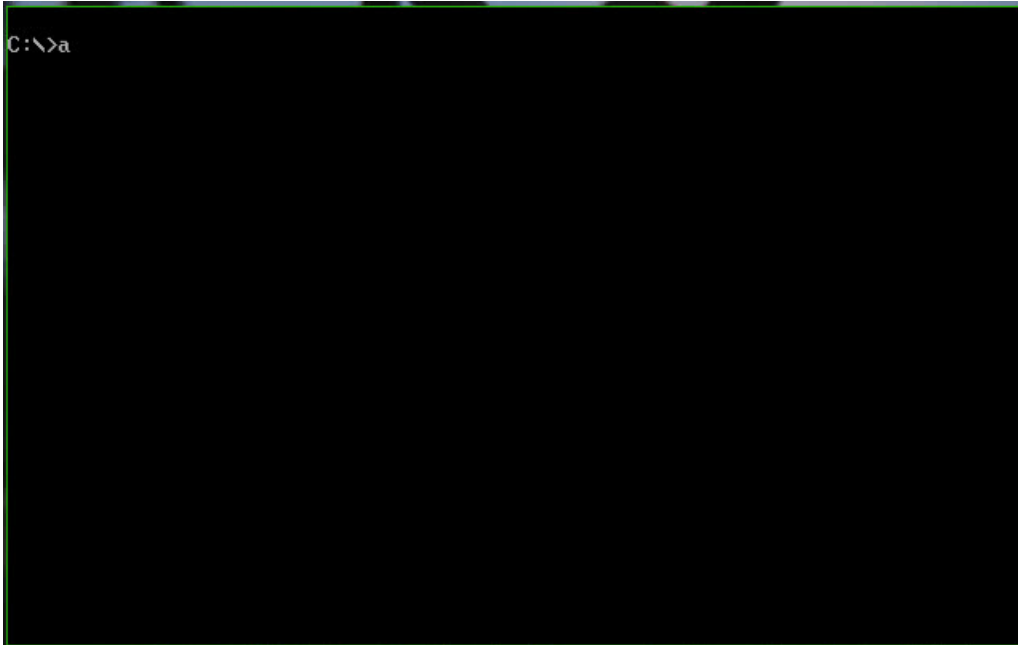
Command lines aren't the oldest way to tell a computer what to do. Depending on how you think of it, that distinction probably belongs to plugging wires into switchboards, toggling banks of switches, or [moving beads on a wire](https://adafru.it/ekj) (<https://adafru.it/ekj>).



The command line is, however, one of the most *durable* ways for humans to talk to a computer. It's been used continuously for half a century, and seems to be reinvented at least a few times by every generation of computer users.

For example, until Windows became popular, **DOS** was the way you would talk to your home or business PC

DOS looks like this:



To modern eyes, the command line can appear primitive, or outmoded by newer interfaces. That's not quite right. The CLI isn't old so much as it's *eternal*. It keeps cropping up everywhere you look.



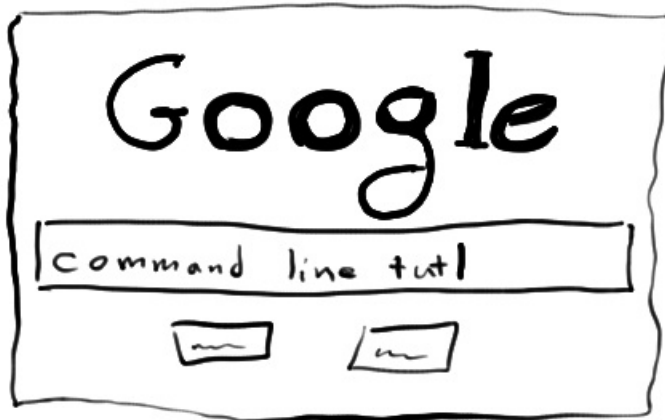
Remember
DOS?



Or that
graphing
calculator
from high
school?

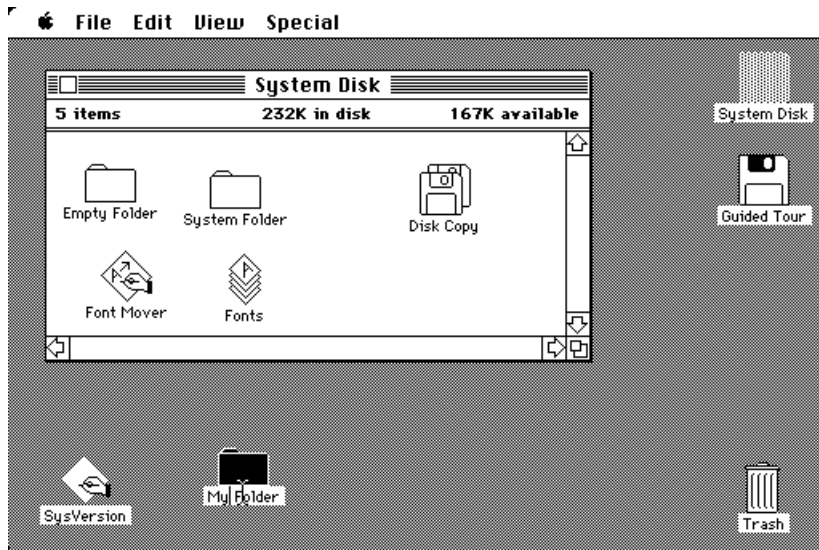
📍 http://www./

The URL bar in
your browser...



...or the search engine most
of us use?

Point-and-click graphical interfaces have been in wide use for a long time now. The Macintosh hit the market in 1984, and Windows machines were common by the early 90s. What's more, both systems leaned heavily on ideas that were [first developed in the 60s](https://adafruit.it/ekk) (<https://adafruit.it/ekk>).



Now that we have all of those pointers, icons, and windows full of polished visual elements, why do command lines keep reappearing, even in the middle of our shiniest and most up-to-date programs?

One way to answer that is to think about programming languages. Just about every piece of software you've ever used is written in a little language with its own grammar and vocabulary. It turns out that language is *really* expressive, and that once you've written a useful sentence in some language, you can usually save it for later use, or remix it to have a different effect.

Language - expressed as text - also has the benefit of being easy to keep in files, easy to send over the wire, and easy to combine with other bits of language. Command lines are one of the most powerful tools at our disposal for expressing and sharing useful ideas, both with the computer and with other humans.

GNU/Linux systems like Raspbian are conceptual descendants of an operating system called Unix, which [dates to the 1960s \(https://adafru.it/ekl\)](https://adafru.it/ekl). These days, Unix is a broad family of OSes, serving all sorts of purposes.

Out of all the modern, widely-used operating system families you'll encounter, Unix is probably the one where command-line interfaces are still the most fundamental and widely used.

If you're interested in the history of how all this came to be, there's a lot of good writing out there.

- ["In the Beginning was the Command Line" \(https://adafru.it/ekm\)](https://adafru.it/ekm) is a now-classic essay on operating systems and interfaces by Neal Stephenson
- [The Unix Programming Environment \(https://adafru.it/ekh\)](https://adafru.it/ekh), by Brian Kernighan and Rob Pike, is a deep and rewarding exploration of the philosophy that produced much of the modern CLI toolset
- [Hackers \(https://adafru.it/ekn\)](https://adafru.it/ekn), by Steven Levy, explores the culture of early computing in several important scenes
- [The Evolution of the Unix Time-sharing System \(https://adafru.it/eks\)](https://adafru.it/eks), by Dennis M. Ritchie

And if you've got half an hour to kill and want a real vintage computing fix, this Unix documentary from 1982 is *amazing*.

We'll come back to the history, but right now let's focus on the what and how of getting your own shell. You're going to need a **terminal**.

What's a Terminal?

Once upon a time, terminals were dedicated pieces of hardware, and looked something like this:



Note the maker, digital and the name of the product, **VT100: Video Terminal 100** (<https://adafru.it/ekv>)

Nowadays, they're almost always a software program, and look something like this:

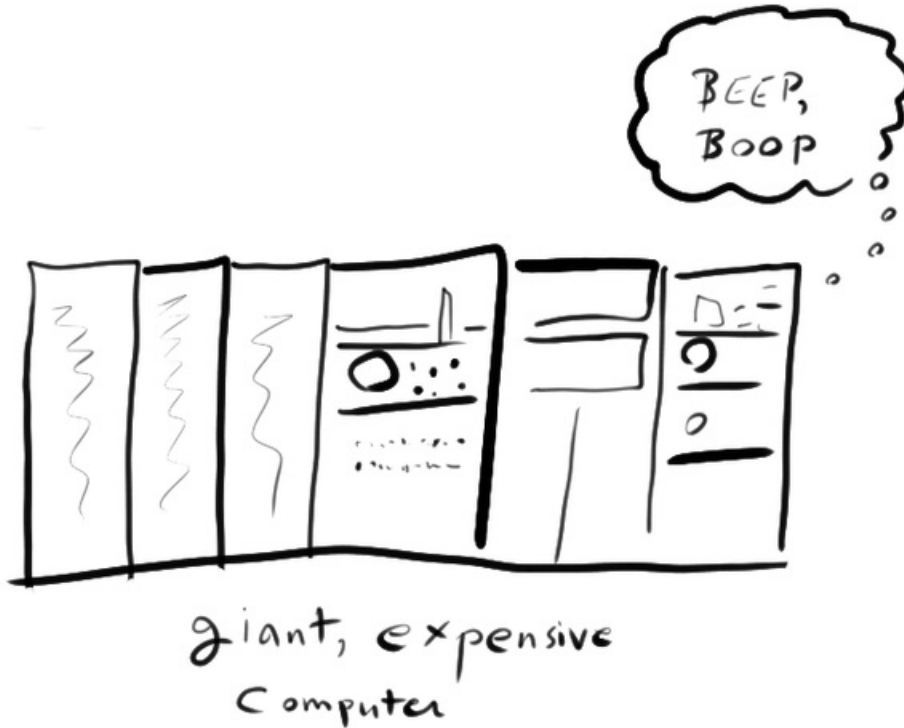
```
brennen@desiderata 22:55:02 /home/brennen/code/bas * ls
FROG.BAS icon.mix QBASIC.EXE QBASIC.HLP QBASIC.INI trader
brennen@desiderata 22:55:03 /home/brennen/code/bas * cd ..
brennen@desiderata 22:55:07 /home/brennen/code * ls
according-to-pete          displaygo                p5
adt                       display.php              p5.zip
adt-bundle-linux-x86_64-20140702 git-feed                  reddit
arduino-1.0.6             heather                  sparkfun
bas                       jdk1.8.0_11             squiggle.city.admin
book                     jdk-8u11-linux-x64.tar.gz test-phamt
bpb-kit                  learning_ada              tilde.club
cordova                  mru                      YouTube_Captions
display                  node-v0.10.30
brennen@desiderata 22:55:08 /home/brennen/code * cd /var/www/userland-book
brennen@desiderata 22:55:19 /var/www/userland-book (master) * ls
aidan.html  general_purpose  js                montaigne.txt  shared_space
chapters   get_shell     links.md          programmerthink slides
diff       header.html   literary_environment readme.html    timebinding_animals
endmatter  images       literary_problem  README.md     userland.css
feed.xml   index.html   Makefile         render.pl     web
footer.html introduction  miscellany       script        wordcount.sh
brennen@desiderata 22:55:20 /var/www/userland-book (master) * |
```

Note the new coloring, not possible with original mono-color terminals. And there's almost always a scroll bar so you

can look at your command history

Why terminals?

What possessed the designers of the original VT100 terminal to design such a beast? Well, imagine that you're a university, big company, or research lab back in the middle of last century, and you have a few of these:



We're talking *really* huge and *really* expensive. So what happens when lots of users want to share the scarce resource that is time on a computer?

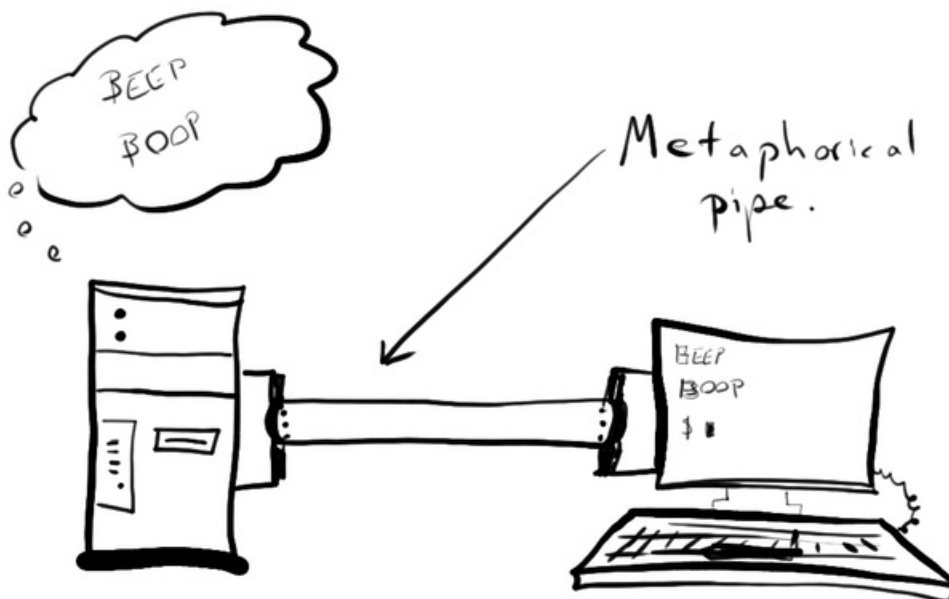
Well, you could hook up a bunch of cheap devices that display output and take input from users...



As it happened, by the time computers came along and needed input/output devices, there was already a [technology in widespread use](https://adafru.it/eko) for sending text back and forth over the wire.

Early on, teletype devices printed output on actual paper and transmitted input from a typewriter-like keyboard. After a while, these were replaced by terminals which incorporated monitors, which in turn were replaced by desktop computers running software to emulate the behavior of the older hardware terminals. (Well, *mostly* replaced - you can still see terminal hardware in service at places like airline check-in desks and some stores.)

Things have changed a lot, but modern terminals still work in ways that reflect this heritage. A terminal, at its most basic, pipes text back and forth between a computer and a user.



Getting a Terminal on Your Raspberry Pi

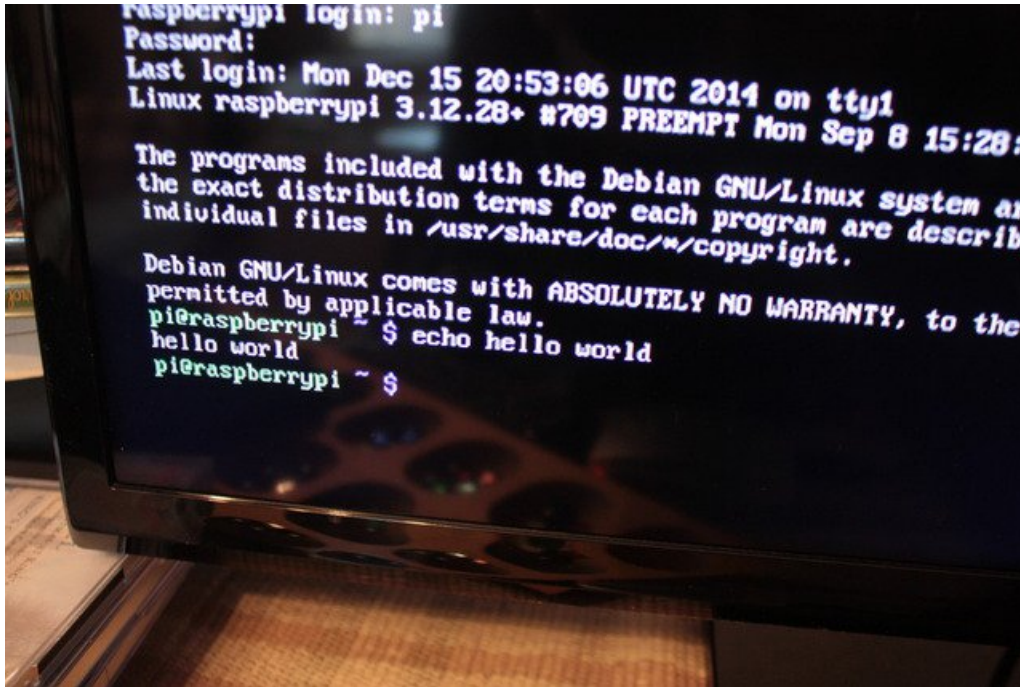
We'll be exploring the Linux command line using the Raspberry Pi as a baseline system. Let's go over the basic options for getting to a shell/command line in a terminal.

Use the HDMI Console

The easiest thing you can do is to just plug a display and keyboard into your Pi, running Raspbian, and work right on the **console**.

This just means that the "terminal" you're using to talk to the shell on your Pi is provided directly by the Linux kernel. You'll need a USB keyboard (mouse optional) and an [HDMI display \(https://adafru.it/CeH\)](https://adafru.it/CeH) (or an adapter for other kinds of display).





Using a Console Cable

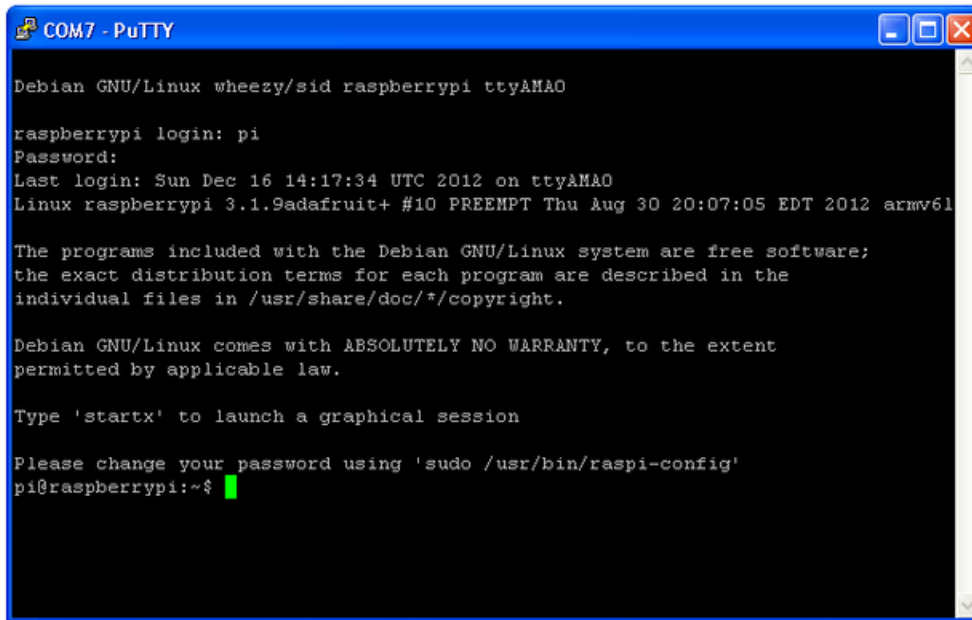


Another option which requires a little more tinkering (but can save you the need for an extra keyboard and monitor) is to hook up a [console cable](https://adafruit.it/kgF) to the Pi from another computer. Check out [our guide on using a console cable \(https://adafruit.it/kgF\)](https://adafruit.it/kgF), which comes complete with instructions for installing a terminal emulator on Mac and Windows machines.

The console cable connects to a couple pins on the Pi, and converts the text into a standard "serial" USB port, which is

supported by tons of free software for Mac, Linux, and Windows like [PuTTY, minicom, or screen](https://adafru.it/dDd) (<https://adafru.it/dDd>).

This is called running the Raspberry Pi headless - because you are not using the 'monitor (head)' as a console.



```
COM7 - PuTTY
Debian GNU/Linux wheezy/sid raspberrypi ttyAMA0
raspberrypi login: pi
Password:
Last login: Sun Dec 16 14:17:34 UTC 2012 on ttyAMA0
Linux raspberrypi 3.1.9adafruit+ #10 PREEMPT Thu Aug 30 20:07:05 EDT 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

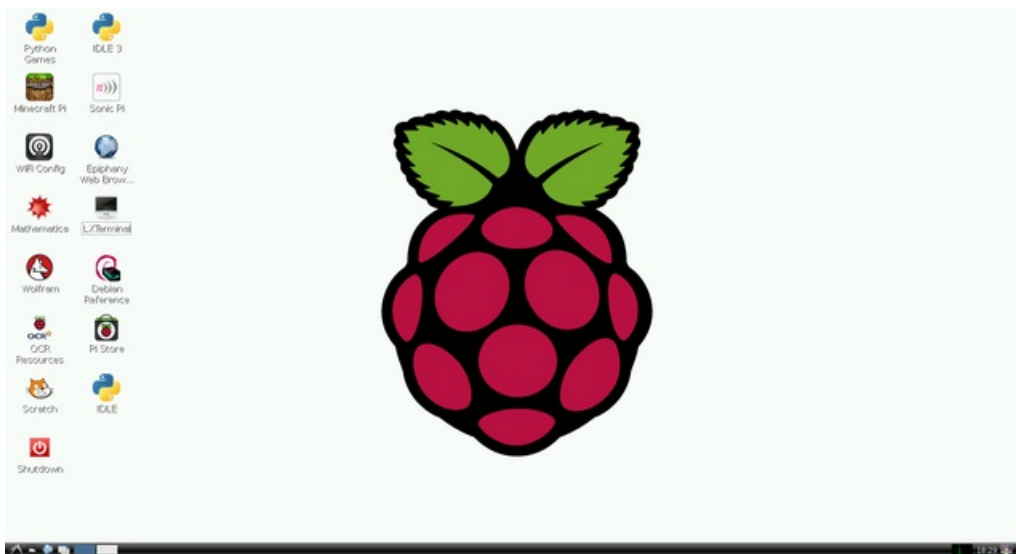
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

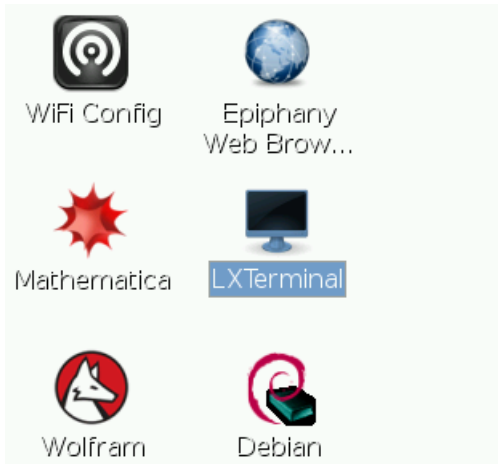
Please change your password using 'sudo /usr/bin/raspi-config'
pi@raspberrypi:~$
```

Run a Terminal Emulator from Your Desktop

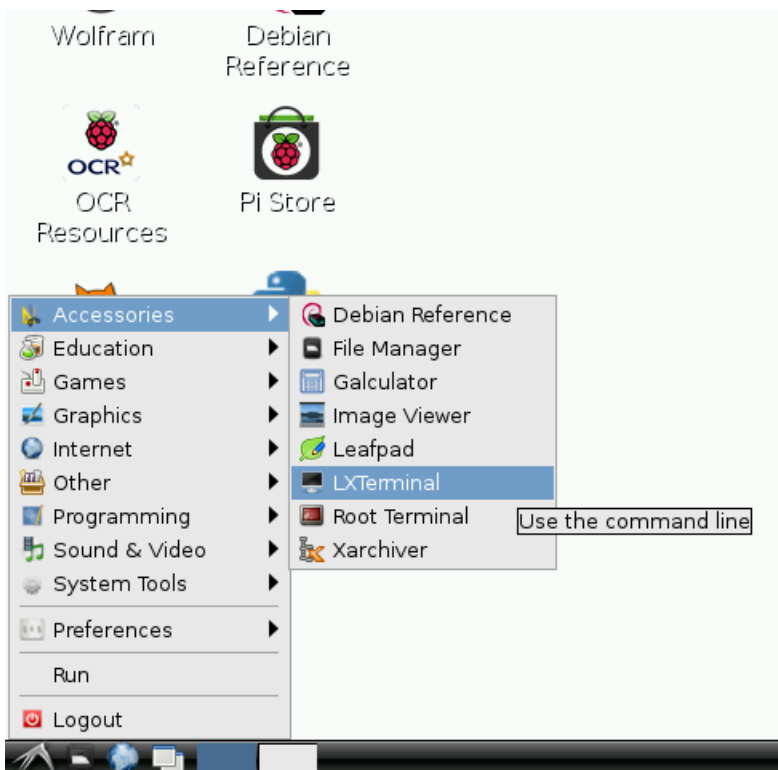
If your Pi is configured to start a graphical desktop when it boots, or if you just prefer to use one, you'll probably be looking at something like this:



Look for a program called "LXTerminal" and fire it up. There should be an icon on the desktop:



...or you can find it under "Accessories" in the main menu:



Connect to Your Pi Over the Network with SSH

Most of the time, when professional nerds need to access the command line on a remote Linux system, they rely on a standard protocol called SSH, which stands for **Secure SHell**.

SSH clients (the Pi is the SSH 'host') are available for all the major operating systems. It's easy to make the Raspberry Pi speak SSH once its connected to Ethernet or WiFi, and like using a console cable, it can save you cluttering your workspace with extra monitors and keyboards. *However*, you need to have the Pi running and connected to WiFi first, which is a bit of a chicken-and-egg problem if you don't already have a way to console in.



Check out our [guide to enabling SSH on the Pi \(https://adafru.it/jvB\)](https://adafru.it/jvB), or try our new [Raspberry Pi Finder \(https://adafru.it/enj\)](https://adafru.it/enj), a friendly desktop application for finding and connecting to a Pi on your local network.

At the Prompt

If everything goes well, you should find yourself looking at a little bit of text followed by a cursor.

For the Pi, by default you'll be using a shell program called **Bash** (**Bourne Again SHell** (<https://adafru.it/ekw>)). There are lots of other shells, but right now Bash is popular enough to be a de facto standard. It's probably what you'll first encounter on Linux systems and the Macintosh. You can even [install it on Windows](https://adafru.it/ekq) (<https://adafru.it/ekq>)!

```
pi@raspberrypi ~ $
```

This is a **prompt**. It's Bash's way of saying "ok, talk to me".

Prompts on different systems will vary (and in fact you can customize your own) but this one is pretty standard. It's set up to tell you a few things about where you're at and what you can do.

Let's break this down.

The diagram shows the prompt `pi@raspberrypi ~ $` with handwritten annotations. An arrow from "Your username" points to "pi". An arrow from "current directory" points to "~". An arrow from "the hostname" points to "raspberrypi".

`pi` is the user you're logged in as.

`raspberrypi` is the hostname of your Raspberry Pi. This is in the prompt because lots of people have a bunch of terminals open at once, and you don't want to forget what computer you're typing at if you can help it.

`~` is the current **working directory**. The tilde character is actually a special shorthand in the shell for your **home directory** (more about that in a minute).

`$` is an indicator that it's your turn to type now. (Sometimes you'll see other characters here, like `%`, `>`, or `#`.)

Try a simple command, typing into either:

- The Pi's USB keyboard if you are using the HDMI console
- Your computer keyboard if using a console cable or SSH

This is the command to try:

```
echo hello world
```




Ready to go exploring? Check out [An Illustrated Shell Command Primer \(https://adafru.it/Cel\)](https://adafru.it/Cel)!