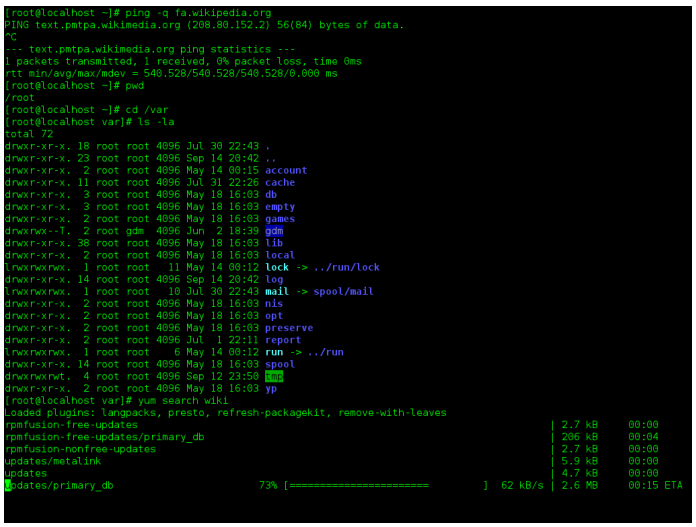
What is Command Line Interface?

Command line interface (CLI) is a program on your computer where you can create files, run programs, and navigate through your folders and other files. It's called **Terminal** if you're using a Mac, and it's called **Command Prompt** for Windows.

This is what it looks like:



Why Learn Command Line?

Why All Developers Should Learn Command Line

Some developers cringe at the mere thought of opening a terminal window. For the uninitiated it can be daunting, stressful, and downright annoying. But devs who understand the command line would argue it's **one of the best tools at your disposal**.

1. You Gain Greater Control over System Functions

Perhaps the most obvious reason to learn the command line is for its original function: **greater control**. There are commands accessible only via shell that can **control very complex operations** on Unix/Linux and Windows machines.

2. You Can use NPM for Package Installs

Node Package Manager is easily the most popular tool for modern developers. This is built on top of Node.js which behaves as a JavaScript framework for other scripts (like NPM).

With NPM you have instant access to a number of tools like:

- Grunt
- Gulp

- Less
- Jade
- CoffeeScript
- Express.js

Another popular choice is to install Ruby gems alongside NPM packages. However Ruby gems are not exactly part of a package manager – though they can act in a similar fashion.

3. You Can Utilize Git Version Control / Version Control System

Programmers and developers alike should understand the power of version control. The ability to **split/merge a project into separate versions** is simply unparalleled.

Unfortunately Git is also difficult to learn if you don't already understand how it operates. Now add the confusion of working inside the command line and it's clear why Git scares off so many developers.

Thankfully there are many free resources online to help you understand the basics.

4. You Need It To Use Preprocessors, Task Runners, and Bundlers

Frontend development has changed a lot in recent years. We have preprocessors like Sass/LESS for CSS and Haml/Jade for HTML. We also have task runners like Gulp and Grunt for automating tasks via Node.js as well as bundlers like Webpack and Parcel to bunlde your website assets and create code splitting for better performance on loading the assets. It's almost a completely new landscape where **these techniques are practically required to build modern websites**.

5. It's For Local Backend Development

Everything from PHP to Rails and Python requires some command line interaction. Installation tools like MAMP have made things easier, but there's no substitution for the raw power of working in the command line.

Shells and Shell Scripts

What is a shell?

A shell is the outermost part of an operating system which you interact with. It takes the commands that you type into it and gives it to the **kernel** (the core of the operating system which has complete control over everything in the system) to perform.

A **shell script** is a text file that contains a sequence of commands for a UNIX-based operating system. This is what you run when you want the kernel to execute specific tasks.

Some examples of tasks that you can do with a shell script are:

- opening a file
- writing a file
- getting information from a file
- executing a process
- and many more!

Why use shell scripts?

1. Shell scripts are useful for automating frequently executed tasks. They can combine several lengthy and repetitive sequences into a single line of code, instead of you having to type them every time.
2. They are used to get routine backups by admins.
3. They are easier to write and debug than other programming languages like C or C++.
4. You can transfer the shell script to other UNIX and similar operating systems and execute.
5. Shell scripts are used to monitor systems regularly.

PowerShellLinks to an external site. is an automated command-line shell and associated scripting language created by Microsoft. It is used for the majority of **Windows operating system**.

BashLinks to an external site., which is short for Bourne Again Shell, is the command shell and scripting language for the majority of the **Linux operating system**.

Secure Shell (SSH)

What is SSH?

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

SSH also refers to the suite of utilities that implement the SSH protocol. Secure Shell provides strong password authentication (Links to an external site.) and public key (Links to an external site.) authentication, as well as encrypted (Links to an external site.) data communications between two computers connecting over an open network, such as the internet.

In addition to providing strong encryption, SSH is widely used by network administrators to manage systems and applications remotely, enabling them to log in to another computer over a network, execute commands and move files from one computer to another.

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. SSH uses the client-server model (Links to an external site.), connecting a Secure Shell client application, which is the end where the session is displayed, with an SSH server, which is the end where the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers.

SSH can also be used to create secure tunnels for other application protocols, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

How does SSH work?

Secure Shell was created to replace insecure terminal emulation or login programs, such as Telnet (Links to an external site.), rlogin (Links to an external site.) (remote login) and rsh (remote shell). SSH enables the same functions -- logging in to and running terminal sessions on remote systems. SSH also replaces file transfer programs, such as File Transfer Protocol (FTP) and rcp (remote copy).