

We select and review products independently. When you purchase through our links we may earn a commission. [Learn more.](#)

How-To Geek

What's the Difference Between Bash, Zsh, and Other Linux Shells?



CHRIS HOFFMAN [@chrisbhoffman](#)

UPDATED JUN 20, 2017, 5:28 PM EDT | 6 MIN READ

```
chris@ubuntu:~$ bash
chris@ubuntu:~$ zsh
chris@ubuntu ~ % tcsh
ubuntu:~> dash
$
```

Most Linux distributions include the bash shell by default, but you could also switch to another shell environment. Zsh is a particularly popular alternative, and there are other shells, like ash, dash, fish, and tcsh. But what's the difference, and why are there so many?

What Do Shells Do?

When you sign in at the command line or launch a terminal window on Linux, the system launches the shell program. Shells offer a standard way of extending the command line environment. You can swap out the default shell for another one, if you like.



building on the concept ever since, adding a variety of new features, functionality, and speed improvements.

Advertisement

For example, Bash offers [command and file name completion](#), [advanced scripting features](#), [a command history](#), configurable colors, command aliases, and a variety of other features that weren't available back in 1971 when the first shell was released.

The shell is also used in the background by various system services. Linux distributions include many functions written as shell scripts. These scripts are commands and other advanced shell scripting functions run through the shell environment.

Shells Leading Up to Bash: sh, csh, tsh, and ksh

RELATED: [What Is Unix, and Why Does It Matter?](#)

The most prominent progenitor of modern shells is the Bourne shell—also known as “sh”—which was named after its creator Stephen Bourne who worked at AT&T's Bell Labs. Released in 1979, it became the default command-interpreter in [Unix](#) because of its support for command substitution, piping, variables, condition testing, and looping along with other features. It did not offer much



The C shell, or “csh”, was developed in the late 1970s by Bill Joy at University of California, Berkley. It added a lot of interactive elements with which users could control their systems, like aliases (shortcuts for long commands), job management abilities, command history, and more. It was modeled off the C programming language, which the Unix operating system itself was written in. This also meant that users of the Bourne shell had to learn C so they could enter commands in it. In addition, csh had quite a few bugs that had to be hammered out by users and creators alike over a large period of time. People ended up using the Bourne shell for scripts because it handled non-interactive commands better, but stuck with the C shell for normal use.

```
chris@ubuntu:~$ tcsh
ubuntu:~>
```

Over time, lots of people fixed bugs in and added features to the C shell, culminating in an improved version of csh known as “tcsh”. But csh was still the default in Unix-based computers, and had added some non-standard features. David Korn from Bell Labs worked on the KornShell, or “ksh”, which tried to improve the situation by being backwards-compatible with the Bourne shell’s language but adding many features from the csh shell. It was released in 1983, but under a proprietary license. It wasn’t free software until the 2000s, when it was released under various open-source licenses.

The Birth of bash

```
chris@ubuntu:~$ bash --version
GNU bash, version 4.3.46(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
chris@ubuntu:~$
```



The Portable Operating System Interface for Unix, or POSIX, was another response to the hectic proprietary csh implementations. It successfully created a standard for command interpretation (among other things) and eventually mirrored a lot of the features in the KornShell. At the same time, the GNU Project was attempting to create a free, Unix-compatible operating system. The GNU Project developed a free software shell to be part of its free operating system and named it the “Bourne Again Shell”, or “bash”.

Bash has been improved in the decades since its first release in 1989, but it's still the default shell on most Linux distributions today. It's also the default shell on Apple's macOS, and is [available for installation on Microsoft's Windows 10](#).

THE BEST TECH NEWSLETTER ANYWHERE

Join **425,000** subscribers and get a daily digest of features, articles, news, and trivia.

By submitting your email, you agree to the [Terms of Use](#) and [Privacy Policy](#).

Newer Shells: ash, dash, zsh, and fish

While the Linux community has settled on Bash in the years since, developers didn't stop creating new shells when Bash was first released 28 years ago.

Kenneth Almquist created a Bourne shell clone known as Almquish shell, A Shell, “ash”, or sometimes just “sh”. it was also POSIX compatible and became the default shell in [BSD](#), a different branch of Unix. The ash shell is more lightweight than bash, which makes it popular in embedded Linux systems. If you have a [rooted Android phone](#) with BusyBox installed—or any other device with the BusyBox



```
chris@ubuntu:~$ dash
$
```

Debian developed a shell environment based on ash and called it “dash”. It’s designed to be POSIX-compliant and lightweight, so it’s faster than Bash, but won’t have all its features. Ubuntu uses the dash shell as its default shell for non-interactive tasks, speeding up shell scripts and other tasks running in the background. Ubuntu still uses bash for interactive shells, however, so users still have the full-featured interactive environment.

```
chris@ubuntu:~$ zsh
chris@ubuntu ~ % setopt correct
chris@ubuntu ~ % rsnc
zsh: correct 'rsnc' to 'rsync' [nyae]?
```

One of the most popular newer shells is Z shell, or “zsh”. Created by Paul Falstad in 1990, zsh is a Bourne-style shell that contains the features you’ll find in bash, plus even more. For example, zsh has spell-checking, the ability to watch for logins/logouts, some built-in programming features like bytecode, support for scientific notation in syntax, allows for floating-point arithmetic, and more features.

```
chris@ubuntu:~$ fish
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
chris@ubuntu -> echo hello world
hello world
chris@ubuntu -> touch "File"
chris@ubuntu -> this-command-is-not-valid-so-it's-red
```

Another newer shell is the Friendly Interactive Shell, or “fish”, released in 2005. It has a unique command-line syntax that’s designed to be a bit easier to learn, but isn’t derived from either the Bourne shell or C shell. It’s an interesting idea, but what you learn through using fish won’t necessarily help you use bash and other Bourne-derived shells.



You don't need to choose a shell. Your operating system chooses your default shell for you, and that choice is almost always bash. Sit down in front a Linux distribution—or even a Mac—and you'll almost always have a bash shell environment. Bash has quite a few advanced features, but you probably won't use them unless you program shell scripts.

On embedded Linux systems or BSD systems, you'll end up with the ash shell. But ash is a Bourne-based shell and is largely compatible with bash. Any knowledge you have from using bash will transfer to using an ash or dash shell, although some advanced scripting features are not available in this lightweight shell.

Almost every shell you'll encounter is Bourne-based and works similarly—including zsh.

That's why zsh is popular. This newer shell is compatible with bash, but includes more features. The zsh shell offers built-in spelling correction, improved command-line completion, loadable modules that act as plug-ins for your shell, global aliases that allow you to alias file names or anything else on the command line instead of just commands, and more theming support. It's like bash, but with a lot of extras, additional features, and configurable options you might appreciate if you spend a lot of time at the command line.

If you're familiar with bash, you can switch to zsh without learning a different syntax—you'll just gain additional features. If you're familiar with zsh, you can switch to bash without learning a different syntax—you just won't have access to those features.



```
Checking connectivity... done.
Looking for an existing zsh config...
Found ~/.zshrc. Backing up to ~/.zshrc.pre-oh-my-zsh
Using the Oh My Zsh template file and adding it to ~/.zshrc
Time to change your default shell to zsh!
Password:

oh my zsh ....is now installed

Please look over the ~/.zshrc file to select plugins, themes, and options
p.s. Follow us at https://twitter.com/ohmyzsh.
p.p.s. Get stickers and t-shirts at http://shop.planetargon.com.
→ Downloads
```

“[Oh My ZSH](#)” is a tool that helps you more easily enable zsh plug-ins and switch between premade themes, quickly customizing your zsh shell without spending hours tweaking things.

There are other shells, too. For example, the tcsh shell is still around and is still an option. FreeBSD uses tsch as its default root shell and ash as its default interactive shell. If you use the C programming regularly, tsch might be a better fit for you. However, it's nowhere near as commonly used as bash or zsh.

How to Switch Between Shells

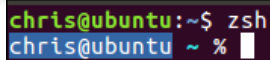
It's easy to switch to a new shell to try it out. Just install the shell from your Linux distribution's package manager and type the command to launch the shell.

For example, let's say you want to try zsh on Ubuntu. You'd run the following commands to install and then launch it:

```
sudo apt install zsh

zsh
```





```
chris@ubuntu:~$ zsh
chris@ubuntu ~ %
```

This is just temporary. Whenever you open a new terminal window or sign into your system at the command line, you'll see your default shell. To change the shell you see when you sign in—known as your login shell—you can generally use the `chsh`, or “Change Shell”, command.

To use this command, you'll first need to find the full path to your shell with the `which` command. For example, let's say we wanted to change to the `zsh` shell. We'd run the following command:

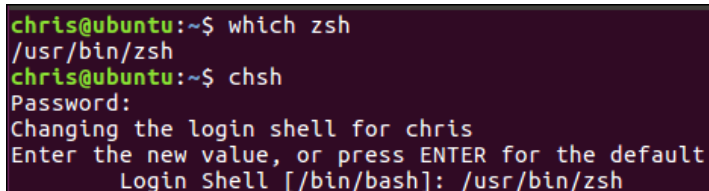
```
which zsh
```

On Ubuntu, this tells us the `zsh` binary is stored at `/usr/bin/zsh`.

Run the following command, enter your password, and you'll be prompted to choose a new login shell:

```
chsh
```

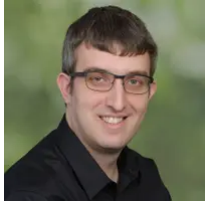
According to the above command, we'd enter `/usr/bin/zsh`. The `zsh` shell would then be our default until we ran the `chsh` command and changed it back.



```
chris@ubuntu:~$ which zsh
/usr/bin/zsh
chris@ubuntu:~$ chsh
Password:
Changing the login shell for chris
Enter the new value, or press ENTER for the default
  Login Shell [/bin/bash]: /usr/bin/zsh
```



CHRIS HOFFMAN



Chris Hoffman is Editor-in-Chief of How-To Geek. He's written about technology for over a decade and was a PCWorld columnist for two years. Chris has written for *The New York Times* and *Reader's Digest*, been interviewed as a technology expert on TV stations like Miami's NBC 6, and had his work covered by news outlets like the BBC. Since 2011, Chris has written over 2,000 articles that have been read more than one billion times—and that's just here at How-To Geek. [READ FULL BIO »](#)

How-To Geek is where you turn when you want experts to explain technology. Since we launched in 2006, our articles have been read more than 1 billion times. [Want to know more?](#)

