What Is the Bash Shell, and Why Is It So Important to Linux?

DAVE MCKAY

MAY 19, 2021, 6:40 AM EDT | 6 MIN READ

The Bash shell is over 30 years old and still going strong. What does it do, where did it come from, and why is it still the most common shell on Linux systems?

What Is a Shell?

When you open a terminal window and type commands, something has to take what you've typed, figure out what you intended, and run the tasks you asked for. The software that does this is the shell. A shell is a command interpreter. It scans what you've typed and picks out the commands, directory names, file names, and program names so that it can figure out what you're trying to achieve.

People often use the phrases "terminal windows," "command line," and "shell" interchangeably, but they're three distinct things. A terminal window is a software representation of a physical teletype terminal. It gives you a connection to the computer. In order to do anything useful, you must be able to type instructions at a command line. The command line is provided by the shell, and the terminal window lets you access the shell.

Shells also allow you to parcel up a collection of commands into a text file called a script. All the commands in the script are executed for you each time you run the script. Scripts deliver efficiency, repeatability, and convenience.

The first Unix shell was the Thompson shell, called `sh`. It was written by Ken Thompson, who is possibly the most key member of the original Unix founding fathers at Bell Labs. The Thompson shell was used as the default Unix shell up to and including Unix Version 6. It was replaced by the Bourne shell in Version 7 of Unix in 1979.

The Bourne Shell

The Bourne shell, written by Stephen Bourne, was an upgraded replacement for the Thompson shell. It was even started using the same command as the Thompson shell, `sh`, to maintain backward compatibility with existing scripts. Backward compatibility was important, but new features were included that provided much functionality that we still use today.

The Bourne shell was an interactive shell and a scripting language. It supported foreground and background task execution and elementary job control. Pipes and redirection were added, along with improvements in handling loops.

The shell now contained some built-in commands, meaning that it didn't need to pass everything out to external utilities, making it more efficient. The Bourne shell even supported "here documents," an elegant way to automate sending data into commands.

The Bourne shell raised the bar and became the new standard.

The Birth of Bash

In 1984, when the GNU project announced its intention to make a free Unix clone—written from the ground up and with a new, permissive licensing—the team needed a shell. When a volunteer who had been working on a shell for the GNU project repeatedly failed to deliver anything at all, Brian Fox was tasked with writing a clone of the Bourne shell.

It was dubbed the Bourne Again Shell, or Bash. This was partly in homage to Stephen Bourne and partly wordplay for the sake of it. After its release in 1989, Chet Ramey contributed some bug fixes to Bash. He eventually became a co-maintainer of the Bash shell. Nowadays, he is still the maintainer of the Bash project.

Linus Torvalds, the creator of the Linux kernel, has said that the first two programs that he ran on his new kernel in 1991 were Bash and `gcc`, GNU's compiler. The pairing of the GNU utilities with the Linux kernel was mutually beneficial. The GNU operating system needed a kernel, and the Linux kernel needed everything else that makes up a Unix clone.

Because Bash is the standard GNU shell, it became the standard shell on all GNU/Linux distributions. Linux flourished to the point that it now underpins an astonishing amount of the modern world. The Bash shell surfed that wave of success, too.

Bash incorporates and improves on the feature set of the Bourne shell, but it also took inspiration from other shells, such as the C shell (`csh`) and the KornShell (`ksh`). For example, the expansion of the tilde "`~`" to the value held in the `$HOME` environment variable comes from the C shell, and the `fc` command that invokes the default editor on commands in the command history comes from the KornShell.

Bash introduced config files such as the ".bashrc" and ".bash_profile" files. Command-line editing on Bash far surpassed the capabilities of previous shells. The manipulation of previously executed commands in the command history was an improved version of the C shells "bang history" feature. Brace expansion was a feature that was missing from the Bourne shell that was implemented in Bash as a superset of the functionality found in the C shell. Arrays were improved by removing their size limits. Parameter expansion in the command prompt allows users to customize their Bash prompt.

The Bash shell aims to be compliant with the POSIX P1003.2/ISO 9945.2 Shell and Utilities standard.

Why Bash Is Still Important

Bash couldn't have lasted this long—over 30 years—as the default Linux shell if it wasn't up to the job. Because of its long service life and massive user base, Bash is mature and very stable. There are many alternative shells available, from veterans like the C shell and the KornShell to newer shells like the Z shell (`zsh`) and the Friendly Interactive Shell (`fish`). Both the Z shell and the Fish shell have some features that Bash doesn't as well as arguably better ways to achieve some of the same things that Bash does. So why is Bash still the dominant shell?

Out of all the Linux machines I have ever been called upon to administer, I don't recall a single one that didn't have Bash as the shell. Unix machines, yes, but Linux boxes, no. It's Bash every time. That familiarity lets you get to work quickly and be effective straight away. You already know Bash, so there's no learning curve. You don't get hamstrung by tiny differences in syntax that have you going around in circles trying to figure out why something doesn't work. Time spent figuring out what the incantation should be on *this* shell is dead time, so it's in the interests of the client business to use a well-known and widely used shell.

Using a shell that is—or is trying very hard to be—POSIX compliant matters to many Linux distributions, but what matters more is compatibility with previous releases. Making changes that could break existing scripts is obviously unattractive. Attractive or not, sometimes, you just have to bite the bullet. On Sept. 3, 1967, Sweden swapped from driving on the left to driving on the right. At 4:50 a.m., all traffic had to stop, slowly make its way to the other side of the road, and stop once more. At 5:00 a.m., traffic could proceed, with everyone now driving on the right.

Will Bash Ever Be Replaced?

What might seem inconceivable now can actually happen later. Unless we want to cling to the belief that we'll all use Bash until doomsday, the truth likely is that one day, Bash will be replaced as the default Linux shell—whether it's still the GNU standard shell or not. Or maybe it will be Bash, but enhanced far beyond the shell that we use today. But whatever takes the place of today's Bash will either have to be completely (or very nearly) backward compatible or be worth the upheaval, whatever the benefits may be.

This isn't without precedent. Since version 10.15 of macOS, Apple has dropped Bash and adopted the Z shell as the default shell. Apple has issues with the GNU General Public License (GPL) v.3. Unfortunately, that is the license that Bash uses. The last version of Bash released under GPL v.2 was 2007's version 3.2. The current version is 5.1. Apple was nearly a decade and a half behind. The only way that Apple could include an up-to-date shell without moving to GPL v.3 was to move to a different shell altogether. To Apple, that was worth the upheaval. (However, you can still switch back to Bash on macOS if you prefer it!)

There's a world of difference between a power user's workstation and a line-of-business Linux server that you have to remotely administer over an SSH connection. Out of almost 1.5 million Amazon EC2 hosted servers, over 93% are running Linux. Almost 75% of web servers are running Linux. Organizations like Red Hat, Amazon, and Google use Linux in-house.

It's hard to imagine what benefits a new shell could offer that would justify that kind of global upheaval. That's why Bash is cemented in place.

Even Microsoft now offers a way to run a Linux-based Bash shell on Windows 10!

DAVE MCKAY

Dave McKay first used computers when punched paper tape was in vogue, and he has been programming ever since. After over 30 years in the IT industry, he is now a full-time technology journalist. During his career, he has worked as a freelance programmer, manager of an international software development team, an IT services project manager, and, most recently, as a Data Protection Officer. His writing has been published by howtogeek.com, cloudsavvyit.com, itenterpriser.com, and opensource.com. Dave is a Linux evangelist and open source advocate. **READ FULL BIO »**