What is Bash?

The first lesson gives you a high-level view of what bash is, and a little on its history.

WE'LL COVER THE FOLLOWING

Bash is a *shell program*.

A *shell program* is typically an executable **binary** that takes commands that you type and (once you hit return), translates those commands into (ultimately) system calls to the Operating System API.

> Note: A binary is a file that contains the instructions for a program, ie it is a 'program' file, rather than a 'text' file, or an 'application' file (such as a Word document).

If you're not sure what an Operating System API is, then don't worry. You only need to know that a shell program is a program that allows you to tell the computer what to do. In that way, it's not much different to many other kinds of programming languages.

What makes bash different from some other languages is that it is a language designed to 'glue' together other programs.

How Important is this Lesson?

How important is this Lesson?

This lesson will delve into the history of bash. This information isn't essential for the topics covered in the rest of the course. However, it'll be useful for a thorough understanding of the subject.

Other shells

Bash is not the only kind of shell. Other shells include:

- `sh`
- `ash`
- `dash`
- `ksh`
- `tcsh`
- `zsh`
- `tclsh`

You might have heard about some of them or may have used them before. The other shells mentioned have different rules, conventions, logic, and histories meaning they can look similar, however, can differ in subtle ways.

Since other shells are also programs, they can be run from within one another!

In the following code sample you run `tcsh` from within the terminal. Note that you will get a different prompt after the `tcsh` command is run.

```
tcsh            # Enter the tcsh shell
echo $dirstack  # Output a variable in the tcsh shell
exit            # Quit the tcsh program, back to bash shell
echo $dirstack  # The variable does not exist in the bash shell
```

Type the above code into the terminal in this lesson.

Type the above commands in to the terminal below.

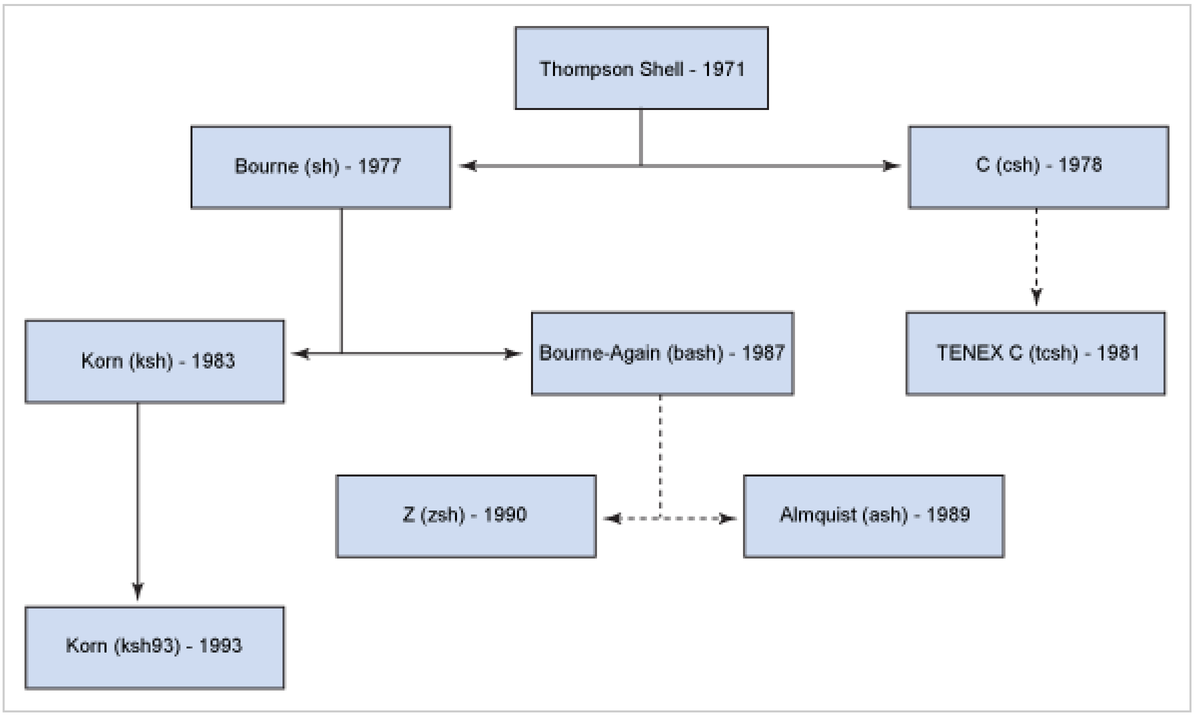
Terminal

> Note: Throughout this course it is assumed that you complete each lesson by typing in the commands in the provided terminal for that lesson in the order the commands are given.

The `dirstack` variable is set by tcsh and will output something meaningful. But once you have `exit`ed from the `tcsh` program, you are back in the bash shell, which does not have a `dirstack` variable by default.

History of bash

This diagram helps give a picture of the history of bash:



A family tree of shells

Bash is called the 'Bourne Again SHell'. It is a descendant of the 'Thompson Shell' and then the Bourne 'sh' shell. Bash has other 'siblings' (eg, `ksh`), 'cousins' (eg, `tcsh`), and 'children', (eg, `zsh`).

The details aren't necessary to know at this point, but it is important to know that different shells exist that can be related and somewhat compatible.

Bash is the most widely seen, used and available shell. However, it is still not unheard of to end up on servers that do not have bash!

What is Bash? Quiz

1 What is bash's historical 'parent' shell?

COMPLETED 0%

1 of 3

What You Learned

- What a *shell* is
- How to start up a different *shell*
- The family tree of shells

What Next?

Next you look at two thorny but ever-present subjects in bash: **globbing** and **quoting**.

Exercises

1) Run `sh` from a bash command line. What happens?

2) What commands can you find that work in `bash`, but do not work in `sh`?