Linux Fu: Alternative Shells

Originalartikel

Backup

<html> <p>On Unix &#8212; the progenitor of Linux &#8212; there was

```
/bin/sh
```

. It was simple, by comparison to today&#8217;s shells, but it allowed you to enter commands and &#8212; most importantly &#8212; execute lists of commands. In fact, it was a simple programming language that could make decisions, loop, and do other things to allow you to write scripts that were more than just a list of programs to run. However, it wasn&#8217;t always the easiest thing to use, so in true Unix fashion, people started writing new shells. In this post, I want to point out a few shells other than the ubiquitous

```
bash
```

, which is one of the successors to the old

```
sh
```

program.</p> <p>Since the 7th Edition of Unix,

```
sh
```

was actually the Bourne shell, named after its author, Stephen Bourne. It replaced the older <a href=„http://man.cat-v.org/unix-1st/1/sh“ target=„_blank“>Thompson shell</a> written in 1971. That shell had some resemblance to a modern shell, but wasn&#8217;t really set up for scripting. It did have the standard syntax for redirection and piping, though. The PWB shell was also an early contender to replace Thompson, but all of those shells have pretty much disappeared.</p> <p>You probably use

```
bash
```

and, honestly, you&#8217;ll probably continue to use

```
bash
```

after reading this post. But there are a few alternatives and for some people, they are worth considering. Also, there are a few special-purpose shells you may very well encounter even if your primary shell is

```
bash
```

.</p> <h2>Two Philosophies</h2> <p>There are really two ways to go when creating a new shell. Unix and Linux custom, as well as several standards, assume you will have

```
/bin/sh
```

available to execute scripts. Of course, a shell script can also ask for a specific interpreter using the

```
#!
```

syntax. That&#8217;s how you can have scripts written in things like

```
awk
```

.</p> <p>That leads to two different approaches. You can create a new shell that is compatible with

```
sh
```

, but extended. That&#8217;s the approach things like the Korn shell (

```
ksh
```

) or the Bourne Again shell (

```
bash
```

) take. On the other hand, you can completely replace the shell with something new like the C shell (practically, now,

```
tcsh
```

which has pretty much replaced C shell). These shells don&#8217;t look anything like the classic shell. Of course, neither does

```
bash
```

if you look at the details, but superficially, most things you can do with

```
sh
```

will work with

```
bash
```

, too, but

```
bash
```

adds a lot more.</p> <h2>Korn Shell</h2> <p><a href=„https://hackaday.com/wp-content/uploads/2020/04/corn.png“ target=„_blank“><img data-attachment-id=„410626“ data-permalink=„https://hackaday.com/2020/05/21/linux-fu-alternative-shells/corn/“ data-orig-file=„https://hackaday.com/wp-content/uploads/2020/04/corn.png“ data-orig-size=„705,945“ data-

comments-opened=„1“ data-image-meta=„{&quot;aperture&quot;:&quot;0&quot;,&quot;credit&quot;:&quot;&quot;,&quot;camera&quot;:&quot;&quot;,&quot;caption&quot;:&quot;&quot;,&quot;created_timestamp&quot;:&quot;0&quot;,&quot;copyright&quot;:&quot;&quot;,&quot;focal_length&quot;:&quot;0&quot;,&quot;iso&quot;:&quot;0&quot;,&quot;shutter_speed&quot;:&quot;0&quot;,&quot;title&quot;:&quot;&quot;,&quot;orientation&quot;:&quot;0&quot;}“ data-image-title=„corn“ data-image-description=„“ data-medium-file=„https://hackaday.com/wp-content/uploads/2020/04/corn.png?w=298“ data-large-file=„https://hackaday.com/wp-content/uploads/2020/04/corn.png?w=466“ class=„alignright wp-image-410626“ src=„https://hackaday.com/wp-content/uploads/2020/04/corn.png?w=187“ alt=„“ width=„140“ height=„187“ srcset=„https://hackaday.com/wp-content/uploads/2020/04/corn.png 705w, https://hackaday.com/wp-content/uploads/2020/04/corn.png?resize=187,250 187w, https://hackaday.com/wp-content/uploads/2020/04/corn.png?resize=298,400 298w, https://hackaday.com/wp-content/uploads/2020/04/corn.png?resize=466,625 466w“ referrerpolicy=„no-referrer“ /></a>David Korn at AT&amp;T wrote a shell that bears his name. If you only know

```
bash
```

, you&#8217;d be a lot more comfortable with

```
ksh
```

than with

```
sh
```

. It is a compatible shell, but offers things we take for granted today. For example,

```
ksh
```

provided command line editing, coroutines, and new control structures like

```
select
```

. It also borrowed ideas from the C shell such as history, functions, and aliases.</p> <p>The only problem with

```
ksh
```

is that AT&amp;T held it pretty close to its chest for years. So even though not many people use

```
ksh
```

today, the ideas in

```
ksh
```

spread to other shells and are widely used today. There is a public domain version,

```
pdksh
```

, if you want to try it out.</p> <h2>Ash and Dash</h2> <p>The Almquist shell, or

```
ash
```

, is basically a clone of the Bourne shell written by Kenneth Almquist. It doesn&#8217;t add a lot of features, but it is very small and fast. This makes it a popular choice for tiny Linux distributions like rescue disks or embedded systems. In 1997 Herbet Xu ported

```
ash
```

for use with Debian and it became

```
Dash
```

&#8212; the Debian Almquist shell. If you use any of the Debian-derived distributions, you&#8217;ll probably find that

```
/bin/sh
```

is a link to

```
dash
```

.</p> <h2>Fish</h2> <p><a href=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg“ target=„_blank“><img data-attachment-id=„410627“ data-permalink=„https://hackaday.com/2020/05/21/linux-fu-alternative-shells/fish-9/“ data-orig-file=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg“ data-orig-size=„1300,1109“ data-comments-opened=„1“ data-image-meta=„{&quot;aperture&quot;:&quot;0&quot;,&quot;credit&quot;:&quot;&quot;,&quot;camera&quot;:&quot;&quot;,&quot;caption&quot;:&quot;&quot;,&quot;created_timestamp&quot;:&quot;0&quot;,&quot;copyright&quot;:&quot;&quot;,&quot;focal_length&quot;:&quot;0&quot;,&quot;iso&quot;:&quot;0&quot;,&quot;shutter_speed&quot;:&quot;0&quot;,&quot;title&quot;:&quot;&quot;,&quot;orientation&quot;:&quot;0&quot;}“ data-image-title=„fish“ data-image-description=„“ data-medium-file=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?w=400“ data-large-file=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?w=733“ class=„alignleft wp-image-410627“ src=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?w=250“ alt=„“ width=„159“ height=„135“ srcset=„https://hackaday.com/wp-content/uploads/2020/04/fish.jpg 1300w, https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?resize=250,213 250w, https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?resize=400,341 400w, https://hackaday.com/wp-content/uploads/2020/04/fish.jpg?resize=733,625 733w“ referrerpolicy=„no-referrer“ /></a>

```
Fish
```

isn&#8217;t named after anyone &#8212; not even a TV detective. It stands for Friendly Interactive Shell. Unlike

```
ksh
```

,

```
ash
```

,

```
dash
```

, and

```
bash
```

,

```
fish
```

doesn&#8217;t try to be compatible with the old classic shell programs. Instead, it tries to be very user friendly. For example, it automatically suggests things as you type.</p> <p>A big feature of

```
fish
```

is that it doesn&#8217;t implicitly create subshells. Consider this (contrived) example:</p> <p>

```
SUCCESS=0; cat /etc/passwd | if grep ^kirk: ; then SUCCESS=1; fi
```

</p> <p>Change &#8220;kirk&#8221; out for a user in your passwd file and try this under

```
bash
```

. Then print out

```
$SUCCESS
```

and you will see it stays zero no matter what. The reason is the part of the command to the right of the pipe character spawned a new shell. You set the variable in that shell, which then exits and the shell you started in still has

```
SUCCESS
```

as zero. With fish, this doesn&#8217;t happen.</p> <p>If you were setting up Linux for a new user,

```
fish
```

might be a good choice for their default shell. For most power users, though, they&#8217;ll want to stick to something more conventional. If you do want to learn more, check out the video, below.</p> <p><iframe class=„youtube-player c1“ width=„800“ height=„480“ src=„https://www.youtube.com/embed/ux1SP9B5lSQ?version=3&amp;rel=1&amp;fs=1&amp;autohide=2&amp;showsearch=0&amp;showinfo=1&amp;iv_load_policy=1&amp;wmode=transparent“ allowfullscreen=„allowfullscreen“>[embedded content]</iframe></p> <h2>Z Shell</h2> <p>The Z shell is newer, dating from 1990. This may be the most popular shell outside of

```
bash
```

on this list. One of the biggest reasons is that it has a plugin architecture that allows for lots of customization including themes and very sophisticated command line completion. You can edit multiline commands easily. Some plugins even provide things like an FTP client.</p> <p>Many of the things you get out of the box with

```
zsh
```

can be added to

```
bash
```

, but it would be a lot of work. If you start

```
zsh
```

as

```
sh
```

, it pretends to be

```
sh
```

&#8212; a lot of advanced shells do that.</p> <p>Because of the plugin architecture, there&#8217;s something like an app store for

```
zsh
```

called &#8220;<a href=„https://ohmyz.sh/“ target=„_blank“>Oh My ZSH</a>.&#8221; If you browse through it, you&#8217;ll probably be tempted to try

```
zsh
```

. If you ask a seasoned Linux user &#8212; at least in the year 2020 &#8212; what shell they use, and they don&#8217;t answer

```
bash
```

, they&#8217;ll probably answer

```
zsh
```

. If you have an hour and a half to kill, you might enjoy the video below.</p> <p><iframe class=„youtube-player c1“ width=„800“ height=„480“ src=„https://www.youtube.com/embed/MSPu-lYF-A8?version=3&amp;rel=1&amp;fs=1&amp;autohide=2&amp;showsearch=0&amp;showinfo=1&amp;iv_load_policy=1&amp;wmode=transparent“ allowfullscreen=„allowfullscreen“>[embedded content]</iframe></p> <h2>And There&#8217;s

More</h2> <p>There are probably more shells, but ultimately it is a matter of personal preference. One we are watching is <a href=„https://github.com/nushell/nushell“ target=„_blank“>Nu shell</a>. It has some interesting ideas about extending the idea of a pipe and stream in Linux. I haven&#8217;t tried it yet, but as it becomes more stable, I might. If you are an

```
emacs
```

fan, there is

```
eshell
```

&#8212; something I&#8217;ll talk about in a future post.</p> <p>Wikipedia has a good <a href=„https://en.wikipedia.org/wiki/Comparison_of_command_shells“ target=„_blank“>comparison matrix of shells</a> if you are curious. Personally? I use

```
bash
```

, but I am always tempted to learn

```
zsh
```

better. I&#8217;ve used all of these at some point except

```
fish
```

. How about you? Leave a comment with your favorite shell, especially if it isn&#8217;t on this list.</p> </html>