Linux Fu: Shell Script File Embedding

Originalartikel

Backup

<html> You need to package up a bunch of files, send them somewhere, and do something with them at the destination. It isn’t an uncommon scenario. The obvious answer is to create an archive — a zip or tar file, maybe — and include a shell script that you have to tell the user to run after unpacking. That may be obvious, but it assumes a lot on the part of the remote user. They need to know how to unpack the file and they also need to know to run your magic script of commands after the unpack. However, you can easily create a shell script that contains a file — even an archive of many files — and then retrieve the file and act on it at run time. This is much simpler from the remote user’s point of view. You get one file, you execute it, and you are done. In theory, this isn’t that hard to do, but there are a lot of details. Shell scripts are not compiled — at least, not typically — so the shell only reads what it needs to do the work. That means if your script is careful to exit, you can add as much “garbage” to the end of it as you like. The shell will never look at it, so it’s possible to store the payload there. <h2>So Then What?</h2> The only trick, then, is to find the end of the script and, thus, the start of the payload. Consider this file,

```
deliver.sh
```

: <pre class=„brush: bash; title: ; notranslate“ title=„“>#!/bin/bashWORKDIR=$(mktemp -d)#find last line +1SCRIPT_END=$(awk ' BEGIN { err=1; } /^\w*_END_OF_SHELL_SCRIPT_\w*$/ { print NR+1; err=0; exit 0; } END { if (err==1) print „?“; }' „$0“)# check for errorif [„$SCRIPT_END“ == '?']then echo Can\'t find embedded file exit 1fi# Extract filetail -n +$SCRIPT_END $0 >„$WORKDIR/testfile“# Do something with the fileecho Here\'s your file:cat „$WORKDIR/testfile“echo Deleting…rm -r „$WORKDIR“exit 0# Here's the end of the script followed by the embedded file_END_OF_SHELL_SCRIPT_A man, a plan, a canal, Hackaday!Not exactly a palindrome, but there's no pleh for it.</pre><h2>Multiple Files</h2> If you don’t mind transmitting script files full of binary garbage at the end, the recovered file might just as well be a compressed tar file or a zip file. The trick is to create your base script and append the file to it. So I might have

```
deliver.sh0
```

as the entire file up to and including the

```
___END_OF_SHELL_SCRIPT___
```

identifier. Then to create the final script you can say: <pre>cat deliver.sh0 bundle.zip >deliver.sh</pre><h2>Encode, Reuse, Recycle</h2> Sometimes you don’t want binary characters cluttering up your shell script. Maybe you want to e-mail the script and you are afraid of what the various mail systems in the path might do to your data. It is easy enough to encode your binary data as text strings (with the associated size penalty, of course). For example, you could just as easily say: <pre>cp deliver.sh0 deliver.shbase64 bundle.zip >>deliver.sh</pre> To recover the file, you’d need some additional work in the main body of the script, specifically after the

```
tail
```

command. <pre>tail -n +$SCRIPT_END $0 | base64 -d >„$WORKDIR/bundle.zip“</pre> Of course, you don’t have to store the file. You could just feed it to another program. A tar archive, for example, might have the line: <pre>tail -n +$SCRIPT_END $0 | base64 -d | tar xf</pre> Naturally, your script can do whatever you need to do to get ready and then maybe process the files after you unpack. You might, say, install a library or a font or merge a patch to the system’s existing files. You could even embed an executable file in a script — even another script — and then execute that script which might unpack another script. It boggles the mind. Just remember that not every system will allow executables to reside on /tmp or on some mounted file systems, so plan accordingly. <h2>Script Doctor</h2> While bash scripting is often maligned and not without reason, it is very flexible and powerful, as this example shows. It is dead easy to embed files in a script and that opens up a lot of flexible options for distributing complex file setups and applications. If you are writing serious bash scripts, we suggest you write them carefully. You can even find a “lint” program that can test for errors for you. </html>

From:
https://schnipsl.qgelm.de/ - **Qgelm**

Permanent link:
https://schnipsl.qgelm.de/doku.php?id=wallabag:wb2linux-fu_-shell-script-file-embedding

Last update: **2025/06/27 11:17**