IT'S UNIX. I KNOW THIS

Some scripting, some running, some chmod'ing

9 steps 90 minutes

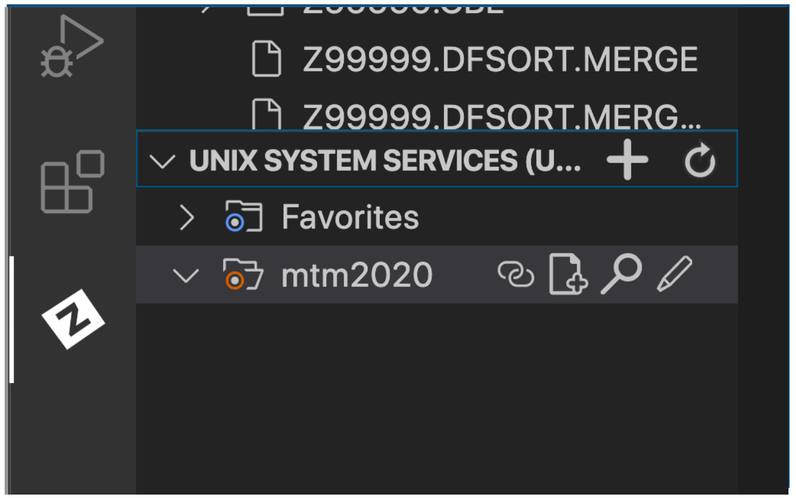
THE CHALLENGE

There are ways to edit files through the terminal, but for most people, it's far easier and intuitive to use the text editor built into VS Code.

For this challenge, we're going to look at USS using VS Code, work on a shell script, and make it run correctly in order to mark everything here as complete.

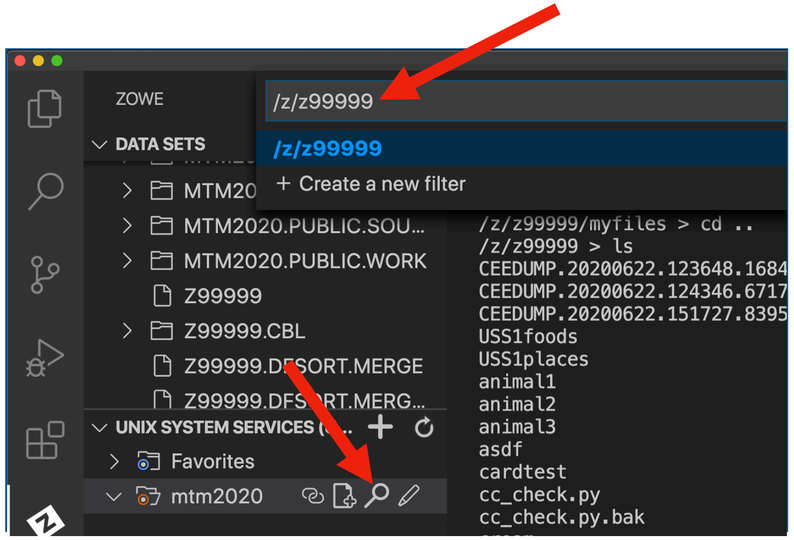
BEFORE YOU BEGIN

You should have VS Code set up and have completed the first USS challenge before starting this. Other than that, nothing else is required, we'll be entirely on the USS side of things



1. OPEN THE USS VIEW

Look in the Unix System Service (USS) section of VS Code. If you didn't change the default location, it should be on the left side, between Data Sets and Jobs. It should also have a profile associated with it (the same one used for the other two). If not, hit the + button to the right of the bar and add it.›

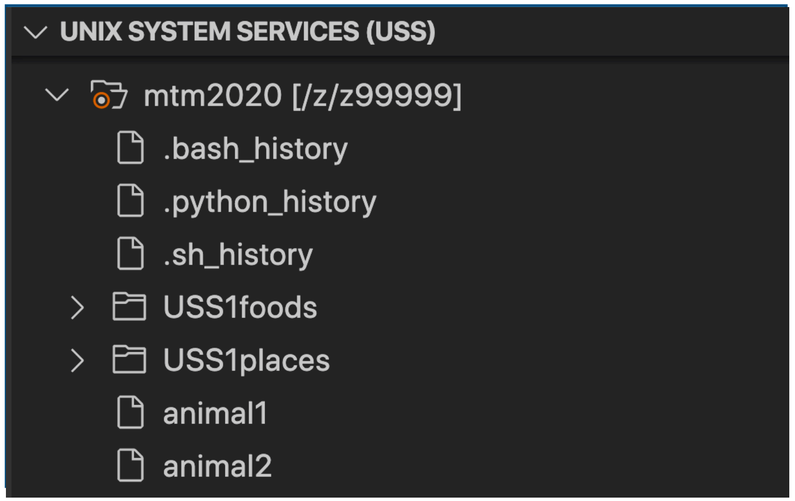


2. FIX YOUR FILTER

Click the Filter button (magnifying glass) to the right of your profile under USS in VS Code. Set it to your home directory.

If you're having trouble remember your home directory, it's the output of the **pwd** command, which should look something like **/z/zxxxxx** (of course, replacing that with YOUR userid. Also, make sure you use lowercase letters here)

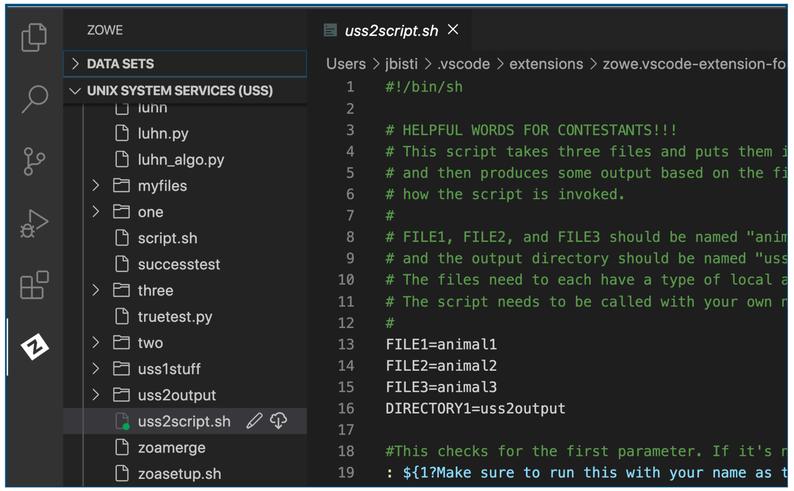
If you get an error saying "Retrieving response from uss-file-list Error", you can fix this by typing the following into your terminal: **/z/public/fixfiles.sh**



3. INVESTIGATE THE OUTPUT

Anything in there look familiar? You can use VS Code to do some basic file and folder work through the USS interface in z/OS, but you'll still need the terminal to do things like issue commands and run scripts. Take a minute and explore, hopefully you see everything you noticed before, just presented a little differently.

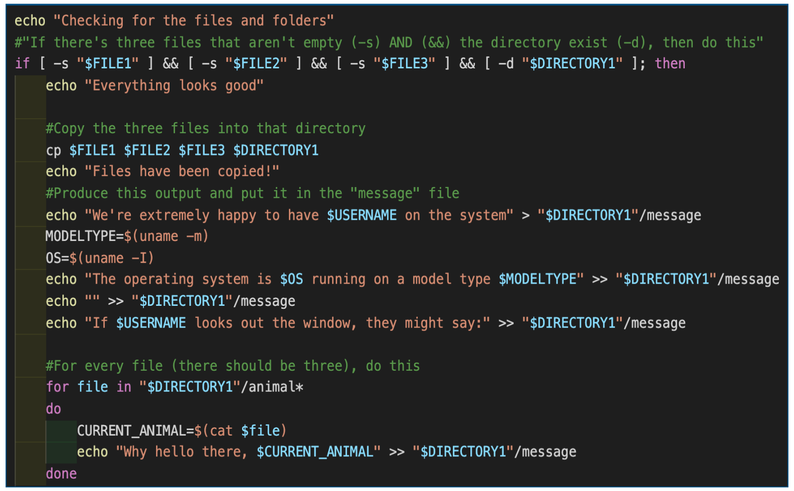
Want to delete or rename something? Check out the grey box at the bottom of Page 4 of USS1 for some tips on how to handle that.



4. OPEN UP uss2script.sh

Look in your home directory and try to find **uss2script.sh**. This should have been copied over in USS1 when you ran the copy.sh script. If you don't see it, copy it out of /z/public or run the copy.sh script in /z/public again, as outlined in the previous challenge.

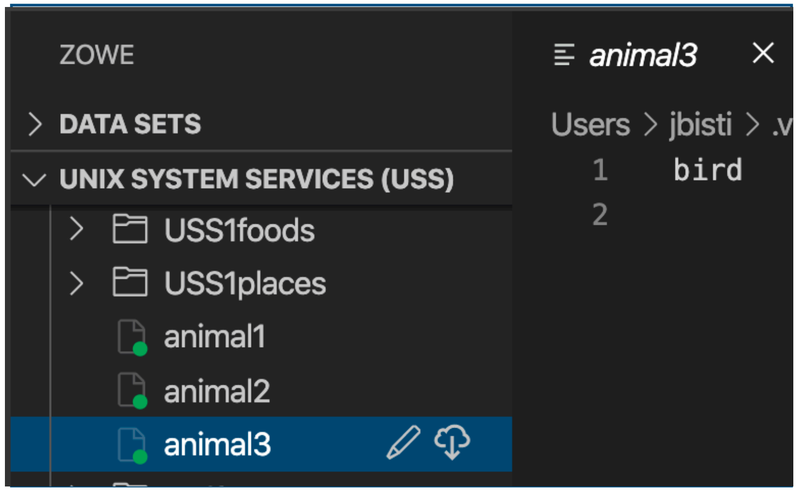
This is a script, just like the other one, except we're not just going to run this, we're going to fix it and make it work.



5. VIEW THE CODE

Open up the shell script and look around. There is a large portion of the script that lives within an “IF” statement, and a lot of it will only run correctly if certain conditions are met.

Read the comments for additional information, and try to figure out what’s going on in the script before proceeding.



6. FIX UP THE INPUT

The script has lots of clues about what it requires to run. Create three files (animal1, animal2, animal3) and a directory called uss2output.

Use VS Code to open those files and enter a single line into each of them, simply naming an animal you’re likely to see in your area. Don’t think too hard on it, but make sure to put something in there otherwise the script won’t work.

Observe the screenshot above. The file ‘animal1’ needs to have an animal named in it. Same with animal2 and animal3.

As you work between the terminal and the GUI, there may be moments when the views become inconsistent. Remember that you may need to reload the VS Code view to catch the latest updates.

WHY DOES THE WORLD NEED ANOTHER SCRIPTING LANGUAGE?

A growing number of programs running on Z came over from Linux or other UNIX platforms, where shell scripts are used to simply and transparently perform setup steps, like verifying there is enough disk space, creating destination directories, and making sure pre-requisite software is installed.

One of the reasons people really like scripting in UNIX is that the commands are the same ones you can use when typing on a shell in interactive mode, but with the added benefit of adding in “IF” and “FOR” statements. The sh shell in USS is somewhat limited, but you can use the more full-featured bash shell later on if that sounds like fun.

```
/z/z99999 > ls -l uss2script.sh
-rw-r--r--   1 Z99999   IPGROUP
/z/z99999 > chmod +x uss2script.sh
/z/z99999 > ls -l uss2script.sh
-rwxr-xr-x   1 Z99999   IPGROUP
/z/z99999 > cat uss2script.sh
```

7. MAKE IT EXECUTABLE

Flip back over to your terminal session (ssh) and do an ls –l on uss2script.sh, with **ls -l uss2script.sh**

Next, make that file executable, by issuing the command **chmod +x uss2script.sh**

Do another **ls -l uss2script.sh** and see if you can notice the difference. There's an X added to the permissions, letting you know that it's executeable, meaning it can be run like a program. R stands for Read, W stands for Write, and X stands for Execute.

One more quick command, the **cat** command (short for concatenate) outputs the contents of a file, by default, to the screen. Use this to peek into the contents of a file that you don't need to edit.
Example: **cat uss2script.sh**

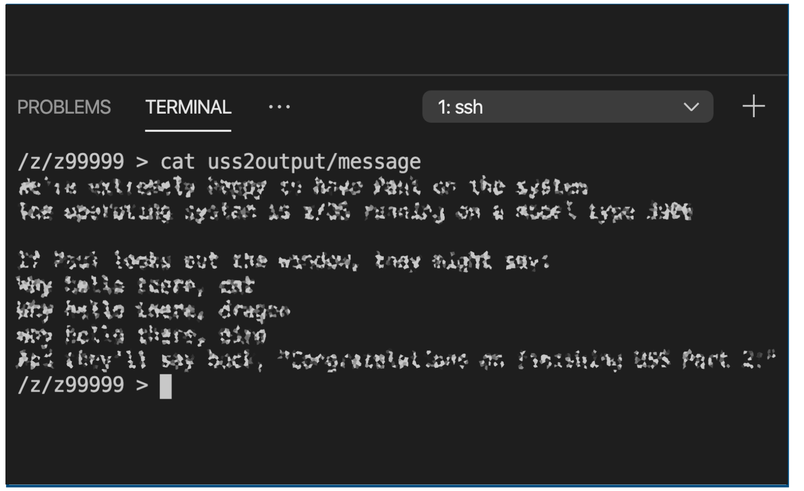
```
-rw-r--r--   1 998347   IPGROUP     2077 Sep 16 08:30 uss2script.sh
 > chmod +x uss2script.sh
 > ls -l uss2script.sh
-rwxr-xr-x   1 998347   IPGROUP     2077 Sep 16 08:30 uss2script.sh
 > ./uss2script.sh
./uss2script.sh 19: Make sure to run this with your name as the first parameter.
 Example: ./uss2script.sh Baron
 >
```

8. RUN THE SCRIPT

Type **./uss2script.sh**

In UNIX, when we give the path to anything, it's assumed we're trying to run, or execute, it. Instead of typing out the full path to the script, we can just use a single dot and a slash to say "From the directory I'm in right now", then the script we want to run.

The script will run, but it will complain about a missing first parameter. Follow its example **with your name** and try again.



9. VERIFY COMPLETION

Run the script again, this time giving it the first argument (the part that comes after the script itself). The script itself says what this should be, along with everything else you need to make this run correctly. When you're finished, check the output in uss2output/message and you should see a message very clearly congratulating you on completing your task.

Look good? Feeling good? Good! Find the CHK job in MTM2020.PUBLIC.JCL and right-click on it, then select SUBMIT JOB to mark it as complete.

NICE JOB! LET'S RECAP

You took a script, figured out what it's doing, and made it run. Working in UNIX System Services means you'll likely be doing this type of thing a lot, so it's good you were able to figure it out.

We made the script full of examples on purpose, since you may want to borrow some of those expressions and conditional checks later on for your own creations. (hint hint)

NEXT UP...

If you've already done the PDS and JCL challenges, your next task is to go after those ZOAU challenges. These use everything you've learned so far, and puts it together in a unique way that really lets you show off your skills.