USS2

IT'S UNIX. I KNOW THIS

Some scripting, some running, some chmod'ing

12 steps **90 minutes**

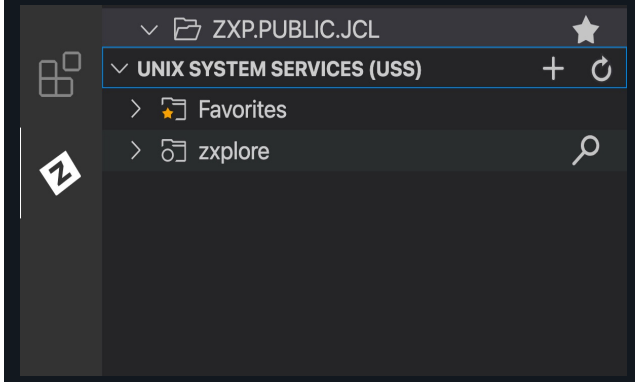
THE CHALLENGE

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

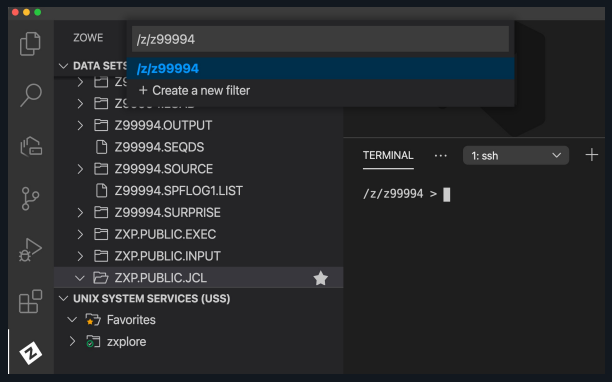
You should have VS Code set up and have completed the first USS challenge in Fundamentals before starting this. Other than that, nothing else is required, we'll be entirely on the USS side of things.



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). In the example above, it is called **zxplore.** If not, hit the + button to the right of the UNIX SYSTEM SERVICES (USS) bar and add it.



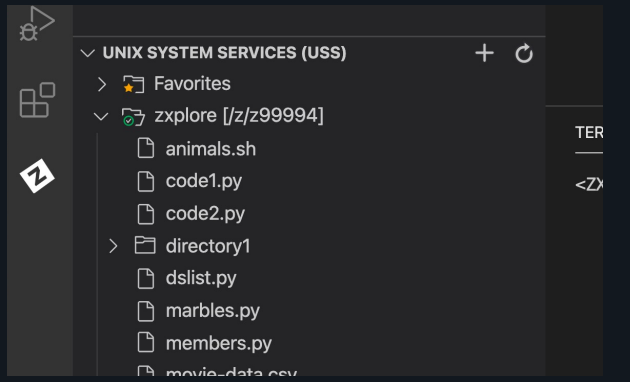
2. FIX YOUR FILTER

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 remembering 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).

Two things to note:

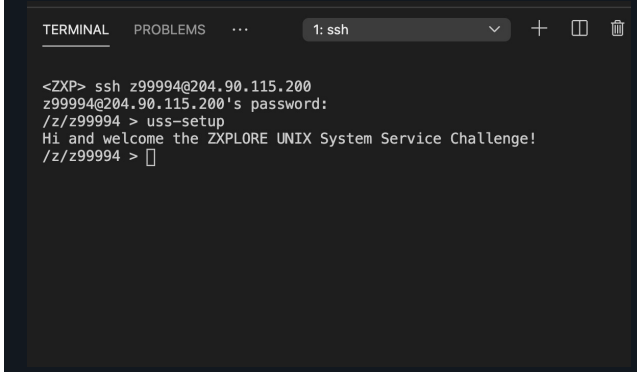
1. *Make sure you use lowercase letters here.*
2. *When you SSH, if you get an error saying "Retrieving response from uss-file-list Error", you can fix this by typing the following into your terminal:* **/z/public/fixfiles.sh**



3. TAKE A LOOK AROUND

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 something? Check out the light blue box at the bottom left of Page 3 of USS for some tips on how to delete files and directories. **Want to rename something?** Right-click on a file or directory and click rename. Although, we suggest not renaming unless instructed to do so.



4. A REFRESHER FOR YOU

Before we go full steam ahead into the USS2 challenge, let's take a moment to remember everything we've learned.

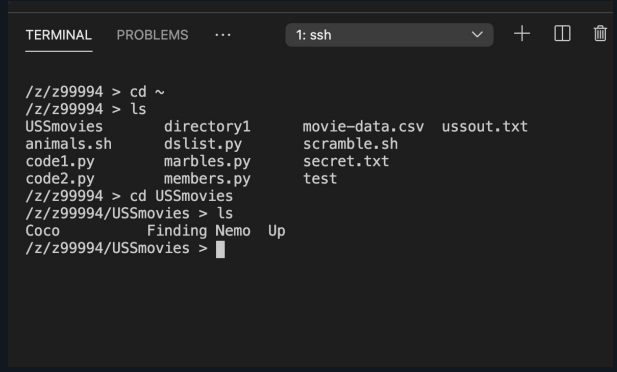
First, make sure your terminal is open, connected through ssh and set up correctly. If you don't remember how to do this or if your default directory is empty, take a look at the screenshot above to find the the connection and *uss-setup* from USS fundamentals.

You know how to make a directory (**mkdir),** make a file (**touch)**, change directories (**cd)**, look around (**ls)** and print the working directory **(pwd).**

WHAT AM I LOOKING AT?

When you look in a directory (or folder, if you're of a Windows fan), you need a handy tool to identify what's in the directory, and what you might be able to DO with the contents (entries). The *ls* command in it's simplest form will show you the names of (unhidden) files in the directory, but it has some options commonly used to show more details (check out ***man ls*** for more):

- -l -- shows the owner information, last update timestamp, and permissions
- -a -- shows all the files, including hidden files (in Unix, filenames that start with "." aren't usually listed
- -t -- sorts the directory entries by last modified time
- -r -- sorts the directory entry listing in reverse
- -F – adds a character code after the entry name to show if the file is "special"



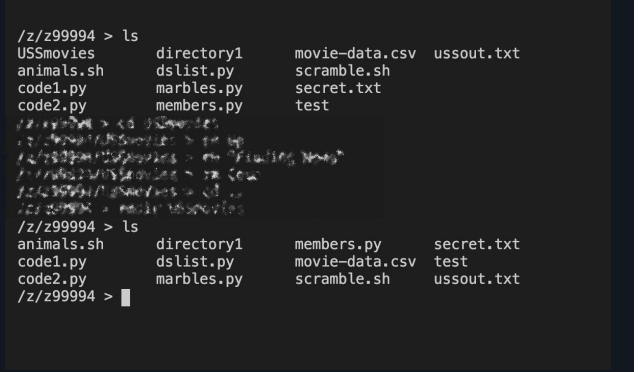
5. PUT IT ALL TOGETHER

Let's practice creating a directory and putting files within it.

Move back to your home directory in terminal (/z/zxxxxx) and create a directory called **USSmovies**.

Put three files in USSmovies named after your favorite movies.

Remember: If the name of your file has spaces, you'll have to put it in quotes, for example: **touch "mac and cheese"**.



6. CLEAN IT UP

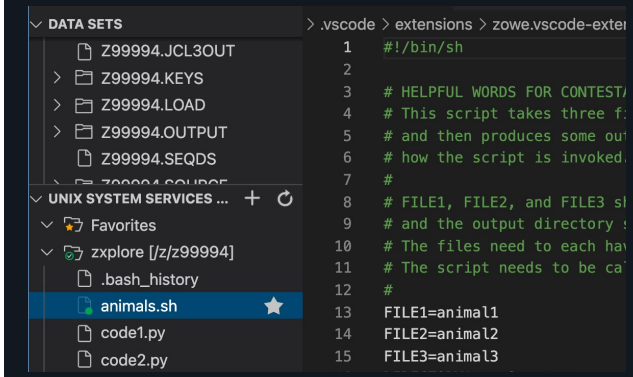
If this felt unfamiliar, keep practicing. You can continue adding files to your home directory until you become comfortable with the shortcuts and structure.

If you successfully created the directory and files, now it is time to delete them. Remember how to do this?

Hint: Refer to the blue box on page 3 of USS fundamentals challenge to remember.

Once you have successfully removed the information you just created, you should no longer see **USSmovies** in your directory (see screenshot above).

You're ready to move on to **Step 7.** We loosen the reins and let you take on UNIX on your own.

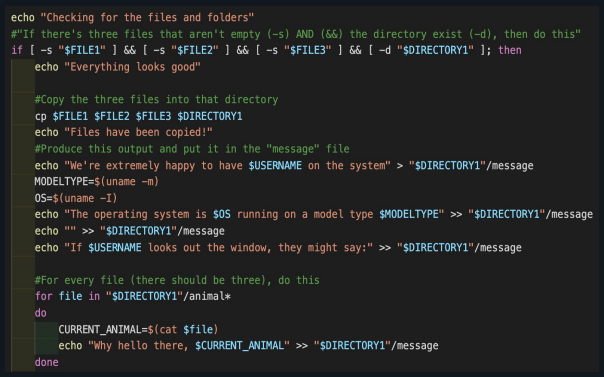


7. OPEN UP animals.sh

Look in your home directory and try to find **animals.sh.** This should have been copied over in USS fundamentals, when you ran the script **uss-setup**.

If you don't see it, copy it out of **/z/public** or run the **uss-setup** script again, as outlined in the previous USS fundamentals challenge.

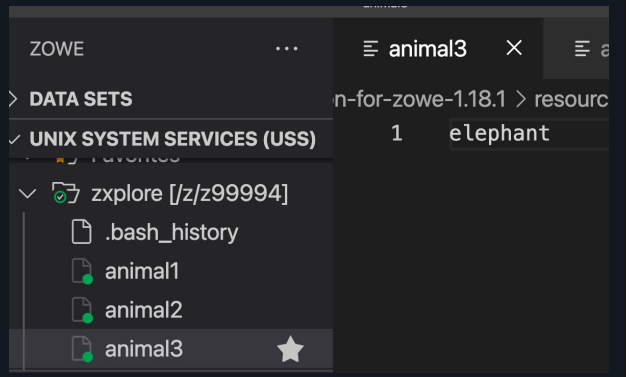
This is a script, just like the other one, except we're not just going to run this, we're going to fix it and make it work.



8. WHAT'S GOING ON?

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.



9. FIX UP THE INPUT

The script has lots of clues about what it requires to run. Create three files (**animal 1, animal 2, 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. Remember to complete each line with an Enter or return so it is a line, not just a word! (***cat animal[123]*** should show 3 lines)

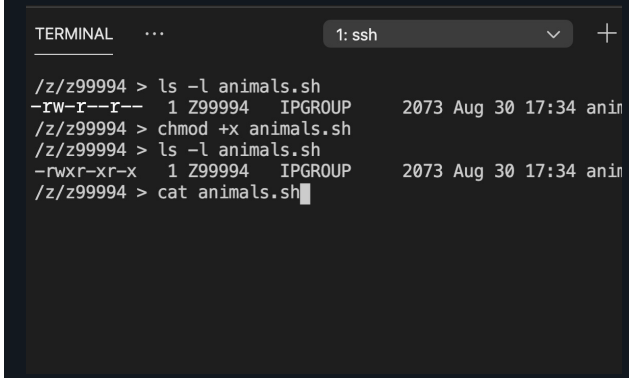
Observe the screenshot above. The file 'animal1' needs to have an animal named in it. Same with animal2 and animal3.

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.



10. MAKE IT EXECUTABLE

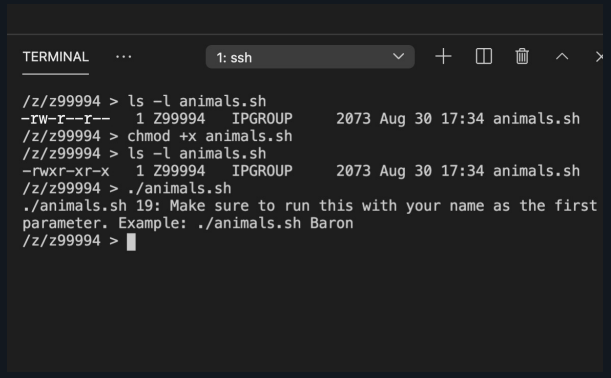
Flip back over to your terminal session (ssh) and do a **ls –l** on **animals.sh**, with **ls –l animals.sh**

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

Do another **ls –l animals.sh** and see if you can notice the difference. There's an **X** added to the permissions, letting you know that it's executable, 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 animals.sh**

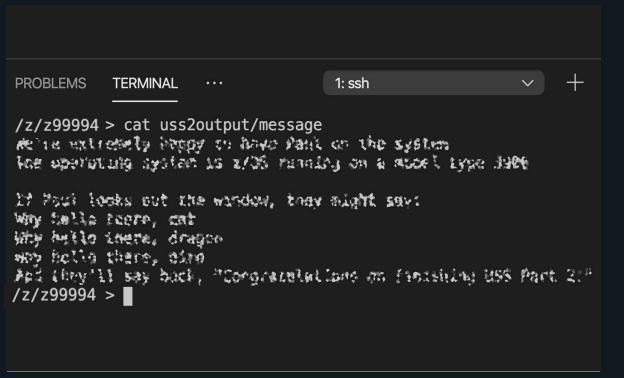


11. RUN THE SCRIPT

Type **./animals.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.



12. 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 **CHKUSS2** job in ZXP.PUBLIC.JCL and right-click on it, then select SUBMIT JOB to mark it 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 JCL2 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.