USS: A Prompt Lesson

There's a bit of UNIX in your mainframe.
That's a good thing.

12 steps 75 minutes

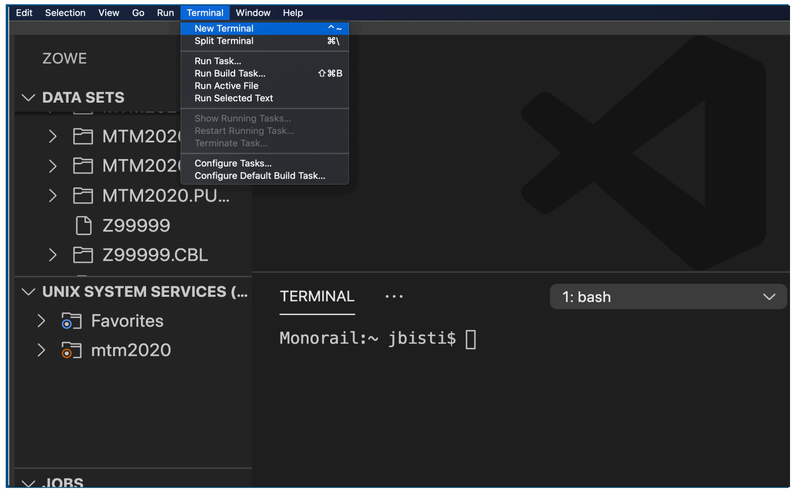
THE CHALLENGE

Sometimes you want z/OS, sometimes you just want to get to an interactive UNIX shell and hammer out commands. Fortunately, there is a UNIX interface within z/OS called UNIX System Services, or USS, so you can log in through ssh and hack around that way.

If you've done any sort of *NIX before, this will be a breeze, but it's still good to see what things look like over here.

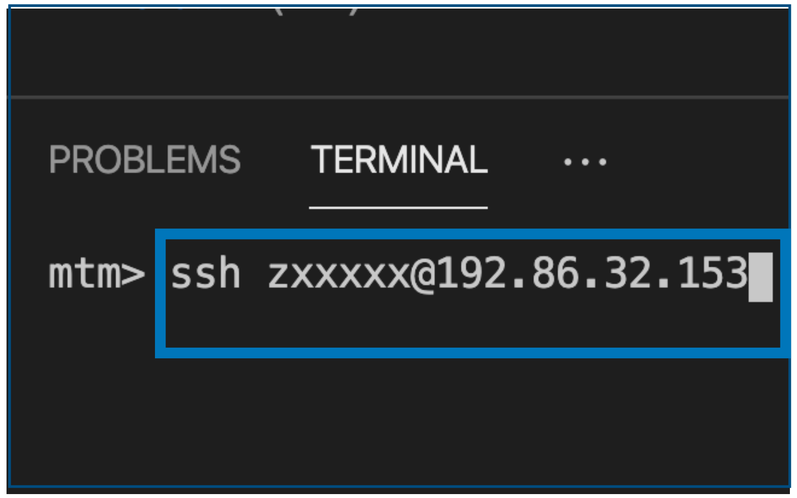
BEFORE YOU BEGIN

If you've got VS Code, then you're all set, there's a terminal in there. You can also use whatever terminal is built into your operating system, OR download and use something new and fancy!



1. FIND THE TERMINAL

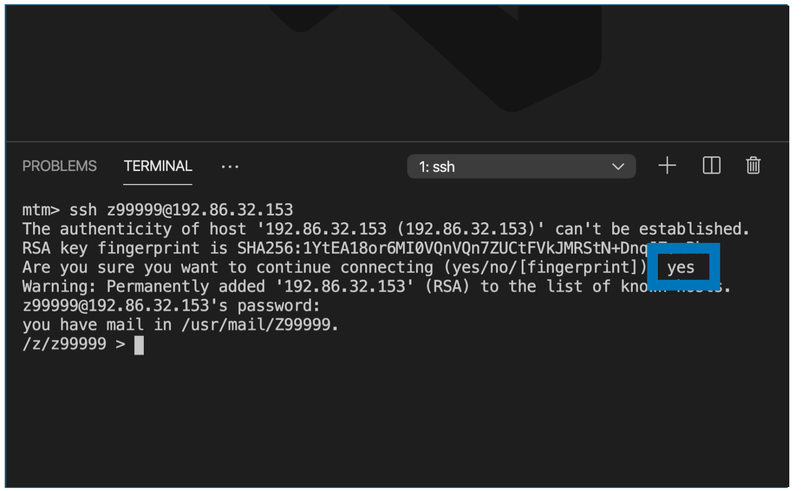
Look for the Terminal section in the lower portion of your VS Code window. If it's not there, try using the Terminal menu and selection "New Terminal". This is a text-based method of interacting with your own personal system.



2. CONNECT THROUGH SSH

Log into the z/OS system with the following command: **ssh zxxxxx@192.86.32.153** (replacing zxxxxx with your own userid. Put another way, this says "Use the ssh command to connect me (using this userid) to this system (at this IP address)"

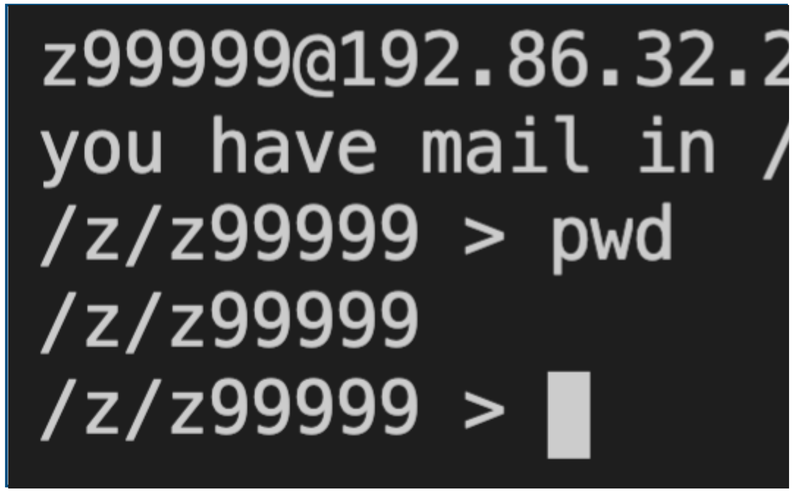
If you get a scary message saying ***"Remote Host Identification Has Changed"***, it's probably because you've connected to this system before, at a different IP address. Welcome back, returning contestant! Follow the instructions in the message to remove the older entry from your known_hosts file.



3. VERIFY ME, THANK YOU

You will be asked for your password, which is the same password you used to log into the z/OS system through VS Code. ***You will not see any characters as you type your password!*** This is to make it so people looking over your shoulder can't steal your password, but the system can still see it. You may also be prompted to trust or accept a key from the remote system. You can safely say **yes** to any of these prompts.

Important to note: The basic UNIX shell can sometimes be tricky. If you mess up a command, try to backspace over the error, and try again.



4. MAKE YOURSELF AT HOME

Type **pwd**. Just the three letters, and then hit Enter. You've just Printed the Working Directory.

In UNIX, files and folders are kept in a hierarchical file system, meaning things go within other things. The output of the pwd command says you're in a directory named after your userid, which is within another directory (or folder) labeled "*z*".

This is known as your "Home Directory", and it's usually where you start out when logging into a UNIX system.

WHENEVER I USE DELETE, SOMETHING GOES WRONG. HELP?

If you're using Windows, try Control-H to backspace over something you want to change
If you're using Mac, try **Control-Delete**
If neither work, or you're getting frustrated trying to re-train your fingers, then enter the command **bash** and hit enter. This switches your shell to Bash, which you'll learn more about in ZOAU1.

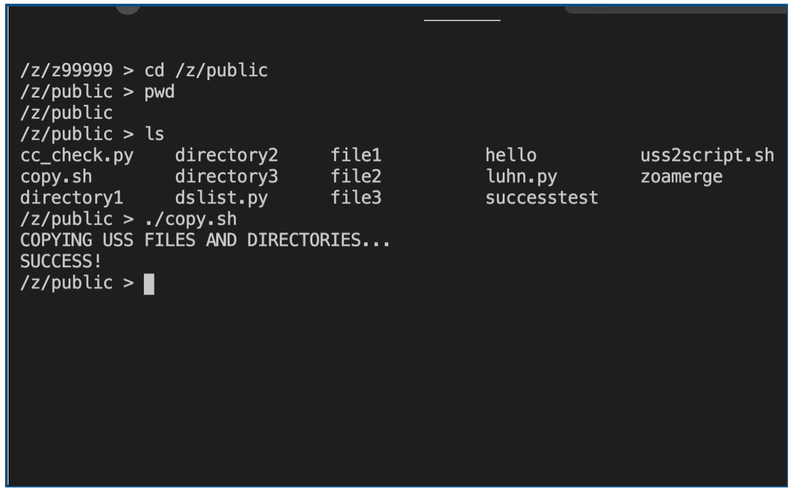
The shell is the program that gathers your commands and shows you the output, and there's more than one. There's more than a dozen, even. The basic shell in USS is based on the Bourne shell, and Bash is the Bourne-Again SHell (computer engineers love being clever) We wanted to start you out with the basic shell so you know what to do when logged into a system without other options. Plus it builds character.

5. CHANGE DIRECTORY

To move around in a UNIX filesystem, use the **cd** command (short for *change directory)*, followed by where you want to go.

The first place you need to go is /z/public so type the command:
cd /z/public
Notice the slash (/) before the z, and the space before that. We need the slash because it's an absolute path. Learn more about absolute and relative paths here.

Then use another **pwd** command to make sure you made it.



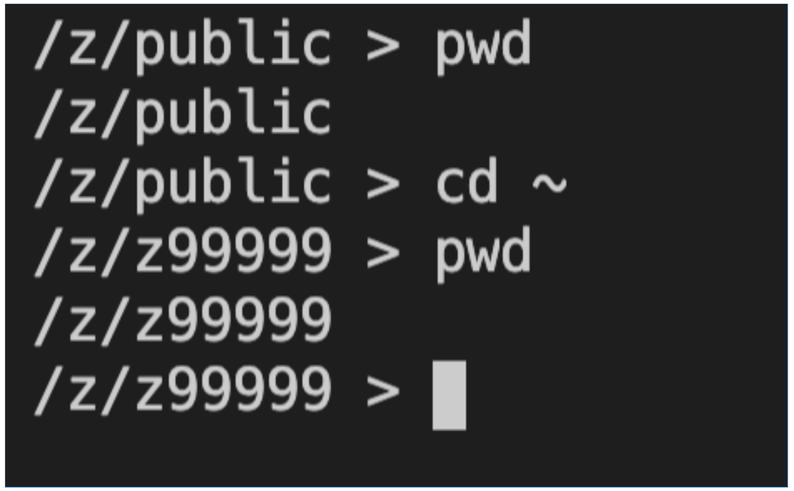
6. RUN A SCRIPT

Before we can do any UNIX tasks, we need to populate our home directory with some goodies. Fortunately, we've got something set up to do that for you. Type **ls** (short for *list*) to look around and see the files and directories in /z/public.

Do you see a file called copy.sh? It's a script, which we'll learn more about later. For now, all you need to do is type the following command to run it:

./copy.sh
(notice the period before the slash. That's important)

It should run and give you a message similar to the one in the screenshot above. If it did, move on to the next step.

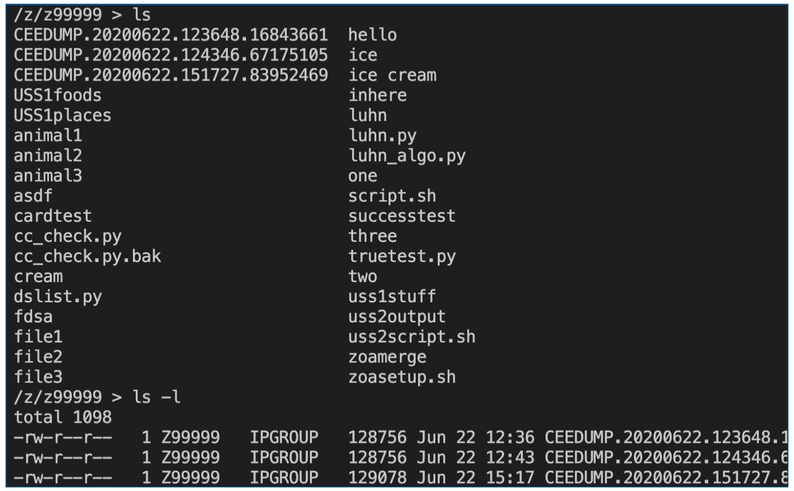


7. MAKE YOUR WAY BACK HOME

UNIX is full of shortcuts and symbols. To go back to your home directory, we could use the **cd** command followed by the absolute path (something like cd /z/zxxxxx) but there's an easier way to find your way back home.

cd ~

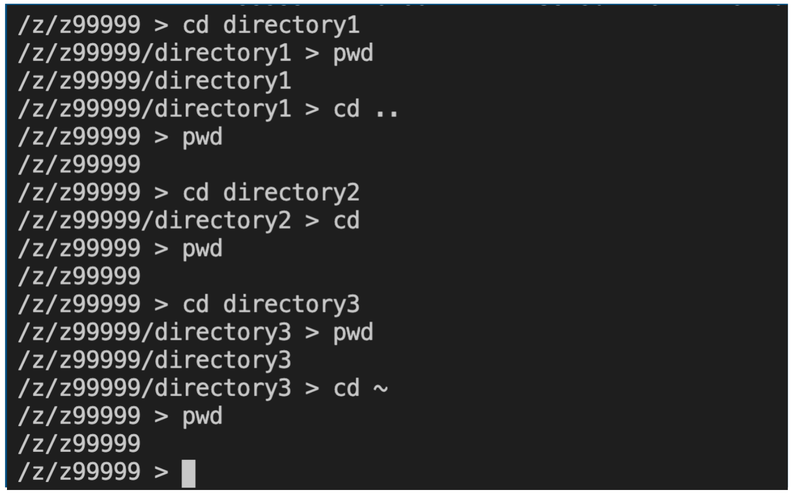
That squiggly line is a tilde, and they key to make it is usually located on the top left corner of your keyboard, next to the number 1. In UNIX, the tilde represents your home directory.



8. LOOK AROUND YOU

Now that we're back home, enter an **ls** to look around and you should see a lot of those files and directories that were out in /z/public. The script we ran copied them over. Your listing will look different from the screenshot above, but a few of the items should look familiar.

If you ever want to start a UNIX-based challenge over, or want a fresh copy of a file, you can find them in /z/public, and you can use that copy.sh script again to copy everything over. Just be aware that it will overwrite existing files.



9. FILES AND FOLDERS

Look at the first letter of the first column of output. If it's a 'd', that means it's a directory. Otherwise, it's probably a file.

Use the **cd** command to 'change directory' into one of those directories, so for example **cd directory1** will put us in directory1, and we can follow that up with **pwd** to print the directory we're in.

To move back a level, to your home directory, type **cd ..**
The two dots mean "back a level".

Or, to just go back to the directory you were most recently in, type **cd -** *(that's cd followed by a single dash)* . Practice moving into the directories and back into the home directory.

And remember, if you get totally lost, you can type **cd ~** (that's a tilde, Shift + the key at or near the top left corner of your keyboard) to take you back Home.

WAIT, SO IS THIS LIKE A VIRTUAL SYSTEM OR WHAT?

USS within z/OS isn't a virtual operating system or a port of Linux or anything like that. It's what we're specifically calling an ***interface***. Unix System Services provides an interface to the user that can run and understand commands in the style of UNIX. When the z/OS Operating System gets those requests, it translates them into what z/OS has to do, performs the work, and returns the result to the user through that same UNIX interface.

This is very useful for programs and scripts that have come to z/OS from a UNIX or Linux-type system. The original UNIX operating system has been ported and re-worked many times since its original creation. Linux, MacOS, the Android OS, and many other OS's you use every day

10. MAKING DIRECTORIES

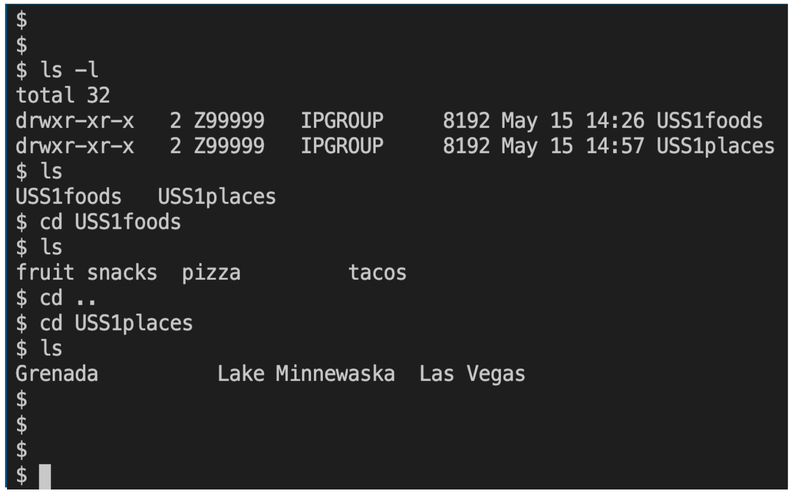
Create a new directory with the **mkdir** command. The command by itself needs to be followed by the name of the directory you want to make so **mkdir mystuff** will make a new directory which you can use to hold files or other directories.

And if you're wondering, directory is just a fancy way of saying folder. Because a directory can be used for purposes other than just containing other things, they're usually referred to as folders, but for now, don't think you have to learn about a whole new thing.

11. CREATING FILES

Use the **touch** command to make a new file. Typically, the 'touch' command is used to update the "last modified" timestamp of a file, but if the file specified does not exist, it will create a new empty file. It's a handy little trick, and we're going to use it here to do just that.

Go into the directory you just made, and then type **touch file1** to create a new file, and follow that up with an **ls** to see it.



12. PUT IT ALL TOGETHER

Now that you know how to make a directory (**mkdir**), make a file (**touch**), change directories (**cd**), look around (**ls**), and print the working directory (**pwd**), you can create your own structure of files and directories.

To mark this challenge complete, do the following:
Move back to your home directory (/z/zxxxxx)
Create two directories: USS1places and USS1foods
Put three files in in USS1places named after your favorite places, and three files in USS1foods named after your favorite foods.

If the name of your file has spaces, you'll have to put it in quotes, for example: **touch "ice cream"**

The screenshot above should help explain the overall structure if you're having trouble with the request. Find the CHK job in MTM2020.PUBLIC.JCL and right-click on it, then select SUBMIT JOB to mark it as complete.

COPYING, MOVING, DELETING

To copy a file, use the **cp** command. For example **cp file1 file2** will make a copy of file1 called file2.
To copy a file into a different directory, just specify the destination (or source) directory.
Example: **cp file1 directoryz** (makes a copy of file1 in the directory called directoryz)
Notice that we left off the name in the destination (directoryz) because it's assumed we want to give it the same name.
If you want to move a file, not just copy it, use the **mv** command. The syntax is very similar
Example: **mv file1 file2.**
This will move file1 to file2, essentially renaming it. If we specify different source or destination directories, then it will change its location.
To delete a file, use the **rm** command. For example, **rm file1** will remove the file called file1;
To delete an empty directory, use **rmdir**. For example, **rmdir mydirectory**