Create a directory called **module11**

1. In your new directory create a file called **groceries** with the following information in it:

lettuce

basil

bread

celery

pasta

tomatoes

Type *sort groceries*

Try this again but this time redirect output to a file called **groceries1**. How did you do this?

Append **groceries** with the word beer Type: *echo beer >> groceries*

Use sort to sort it

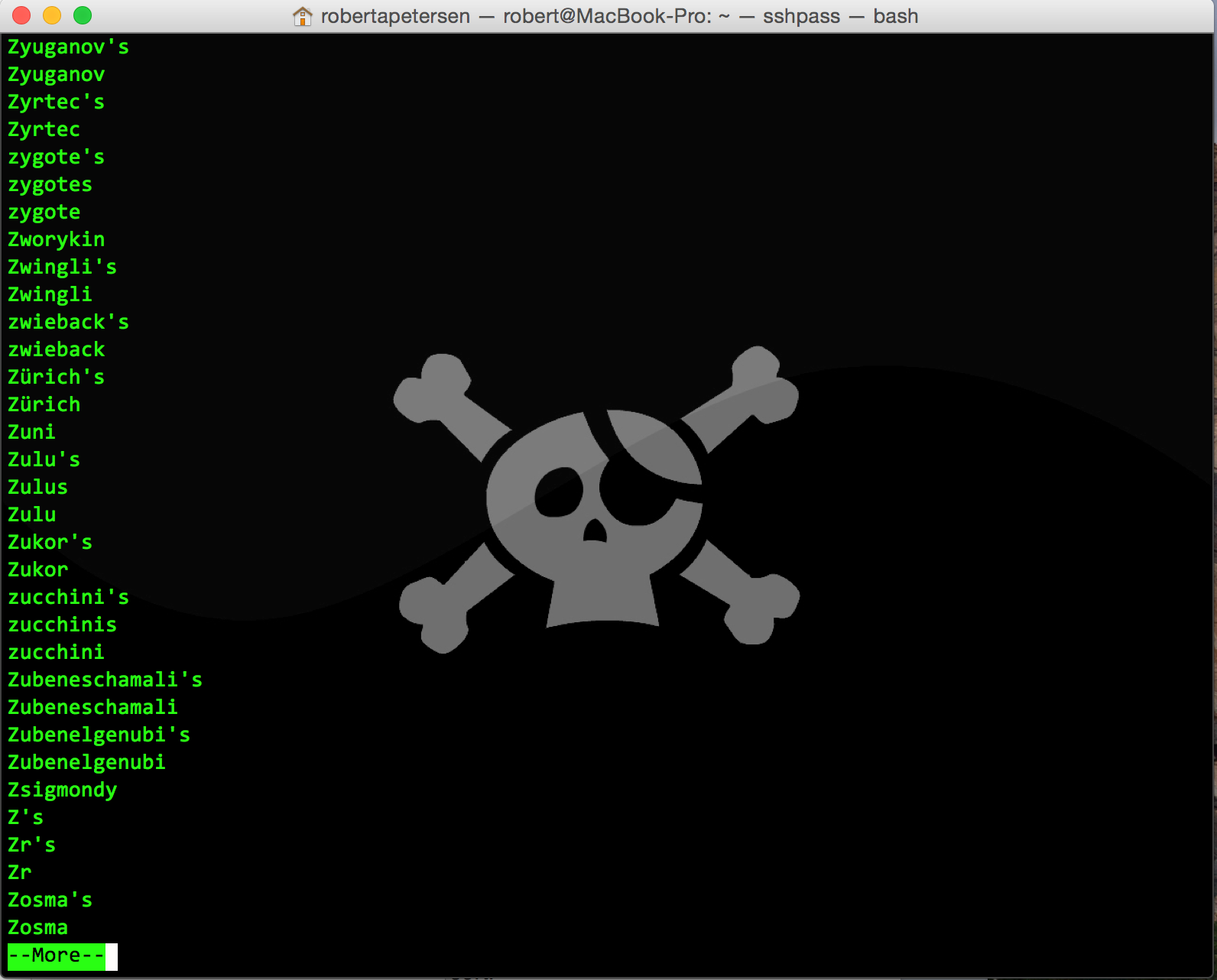
Display your screenshot here.



2. There is a list of words used in Linux dictionary in the /usr/share/dict/words file. Use the more command to read the file (it is big so either use more or pipe it into more) How did you do this?

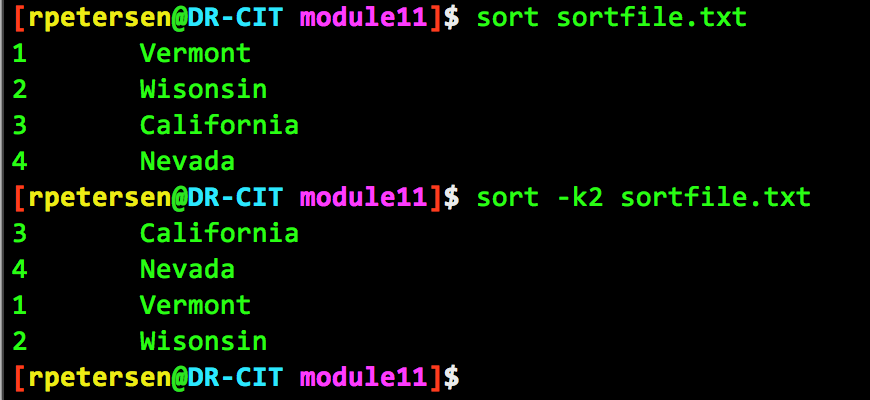
cat /usr/share/dict/words | more

Use sort to sort this file in reverse and pipe it into more. Show me a screenshot here.

cat /usr/share/dict/words | sort -r | more

In the /tmp directory is a file called **sortfile.txt**. Copy it into your **module11** directory and sort it on the first column (default). Then sort it on the second column. Insert screenshot here

3. If we want to see just the first 10 lines of this file we would:

Type: *head /usr/share/dict/words*

You can also see only 5 top lines by

Type: *head -5 /usr/share/dict/words*

What were your results?

The first 5 words in /usr/share/dict/words

Let's get creative. Do the above command but pipe it into sort to sort it in reverse. How did you do this?

head -5 /usr/share/dict/words | sort -r

4. Let's use the tail command to see the last 10 lines in that file.

Type: *tail /usr/share/dict/words*

We can also see more lines by

Type: *tail -11 /usr/share/dict/words*

Use tail to see the last 4 lines of the words file and this time redirect output to append to your file **groceries**. How did you do this?

tail -4 /usr/share/dict/words >> groceries

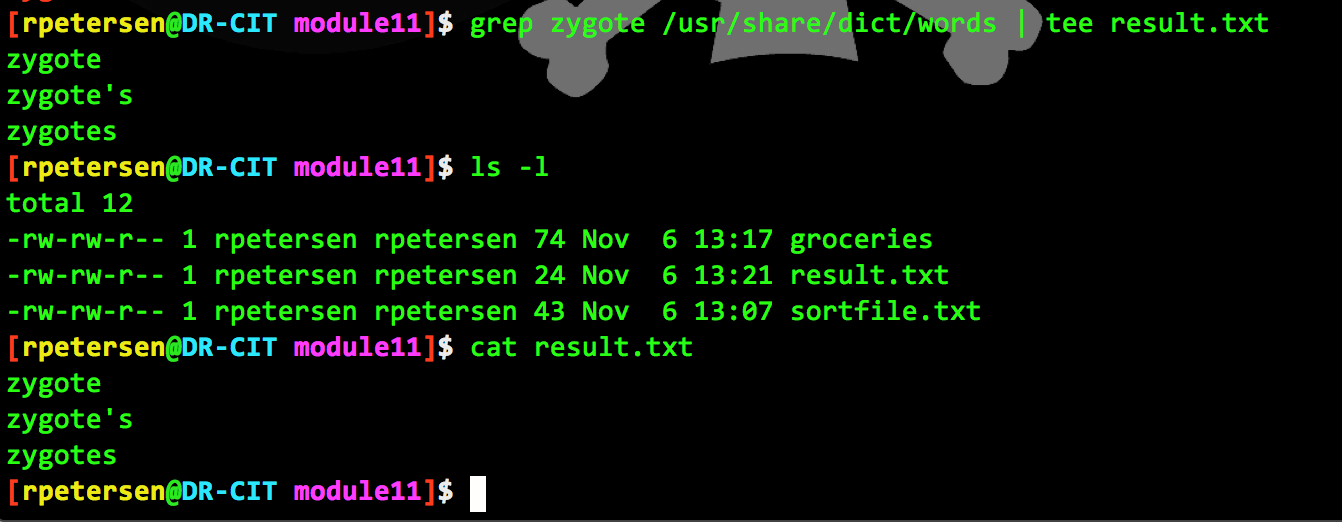
5. Use grep to look for the word “zygote” in /usr/share/dict/words

How did you do this?

grep zygote /usr/share/dict/words

Do the same command only this time use tee to redirect the output to a file in **module11** called **results.txt** AND display it on the screen by typing:

Type: *grep zygote /usr/share/dict/words | tee result.txt*

Do a long listing to show me the file in your **module11** directory and use cat to read it. Put your screenshot here.

6. Type: *printenv*

Narrow down printenv using grep to find all the lines that have your login name in them. What are your results?

**[rpetersen@DR-CIT module11]$** printenv | grep rpetersen

USER=**rpetersen**

MAIL=/var/mail/**rpetersen**

PWD=/home/**rpetersen**/module11

HOME=/home/**rpetersen**

LOGNAME=**rpetersen**

OLDPWD=/home/**rpetersen**

7. Let's looks at some of these built-in variables.

Type: *echo $HOME*

Type: *echo $LOGNAME*

So do variables have to be uppercase? No, but it is always a good idea to do that. But let's test it out.

Type: *me=student*

Type: *echo $me*

How long will that variable hang around? Until a)You change it b)until you log out

Type: *me=YourName* (substitute your name there)

Type: *echo $me*

Another thing to know is this variable works while you are in this shell, but if you go to another one it won't follow you. Let's test this out.

Change shells by typing: *sh*

This puts you in the Bourne shell. See if you variable works there. What happens?

Type: *exit* to go back to your shell (bash)

Type: *export me=Yourname* (again change it to your name)

Go back to the Bourne shell by typing: *sh*

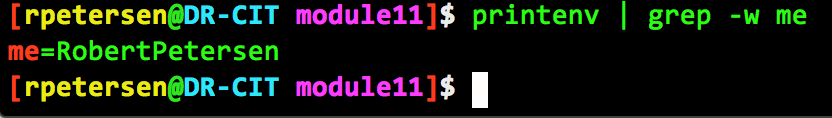
Now test out the variable $me with the echo command. Did it work?

Yes.

Type: *exit*

8. Type: *printenv*

to see if your new variable is there.

Now pipe it through grep to get only the line with “me” in it. Put your screenshot here.

9. Normally, when a program runs under Linux, it inherits the permissions of the user who is running it, so if I run a program under my account, the program runs with the same permissions that I would have if that program were me. Thus, if I cannot open a certain file, the program I am running also cannot open the file in question.

If I set the SUID bit for a file, this causes any persons or processes that run the file to have access to system resources as though they are the owner of the file.

Let's test this out.

Create an empty file called **myfile.txt** in your **module11** directory

Type: *chmod u+s myfile.txt*

Do a long listing and tell me what looks different than the other files.

myfile.txt is highlighted in red, permissions show a capital ’S’ for sticky

You can also do this the numerical way:

Type: *chmod 4750 myfile.txt*

Do along listing so see what it looks like now. Paste your screenshot here.



Just by looking at it, why do you think one has a capital “S” and the other has a lowercase “s”? Take a guess.

Depends on whether or not the user or group has the execute permission?

10. Sticky Bit is used for directories to protect files within them. Files in a directory with the sticky bit set can only be deleted or renamed by the root user or the owner of the directory.

Go back to your home directory to set a sticky bit on your **module11** directory

Type: *chmod 1755 module11*

What do you see in the permissions of module11 that aren't on the others?

A ’t’ where ‘execute’ permission would normally be, indicating ‘sticky’ x.