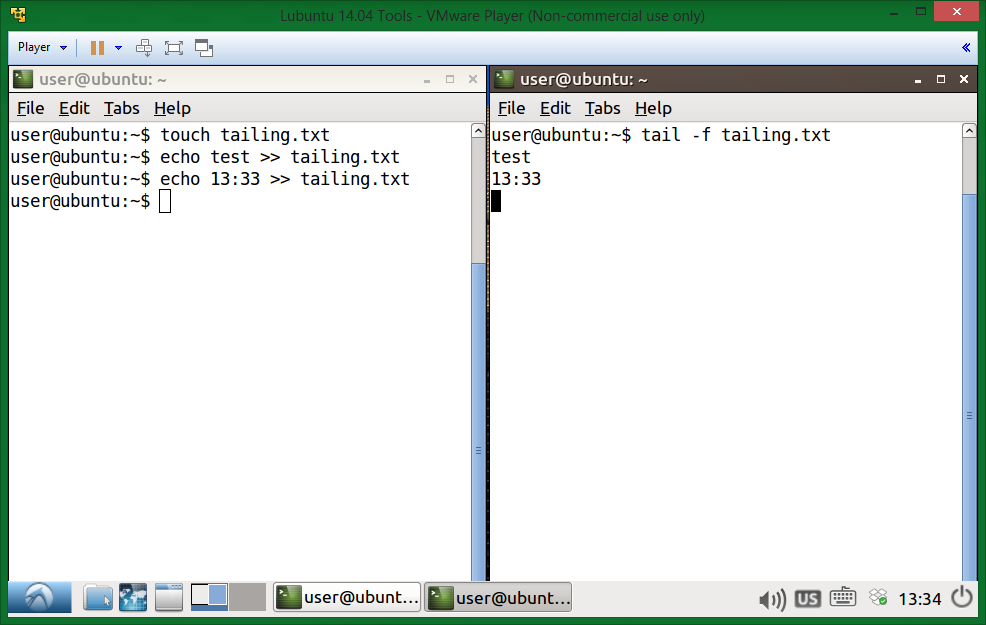
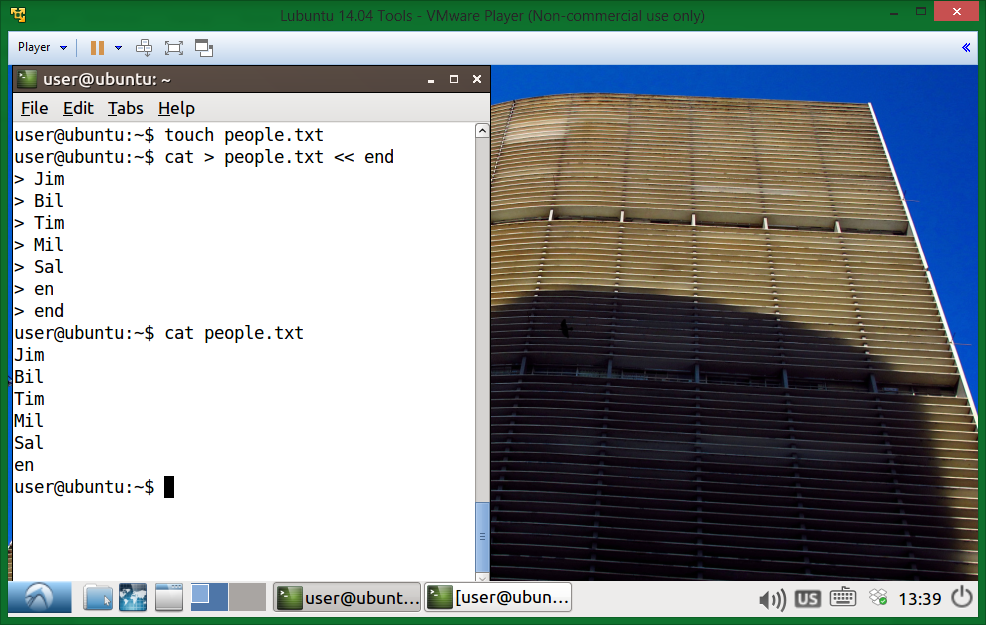
# Some tasks are not included because they are not mentioned to be covered in the lab manual ( example : ‘CUT’ command)

# CHAPTER 18

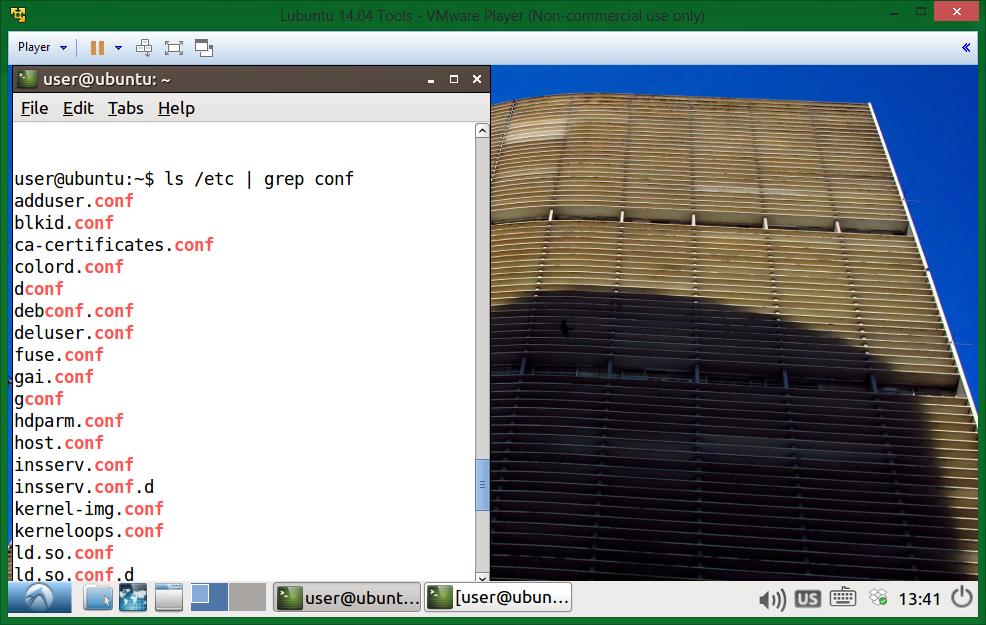
5.Make sure you have two shells open on the same computer. Create an empty tailing.txt file. Then type tail -f tailing.txt. Use the second shell to append a line of text to that file.Verify that the first shell displays this line.

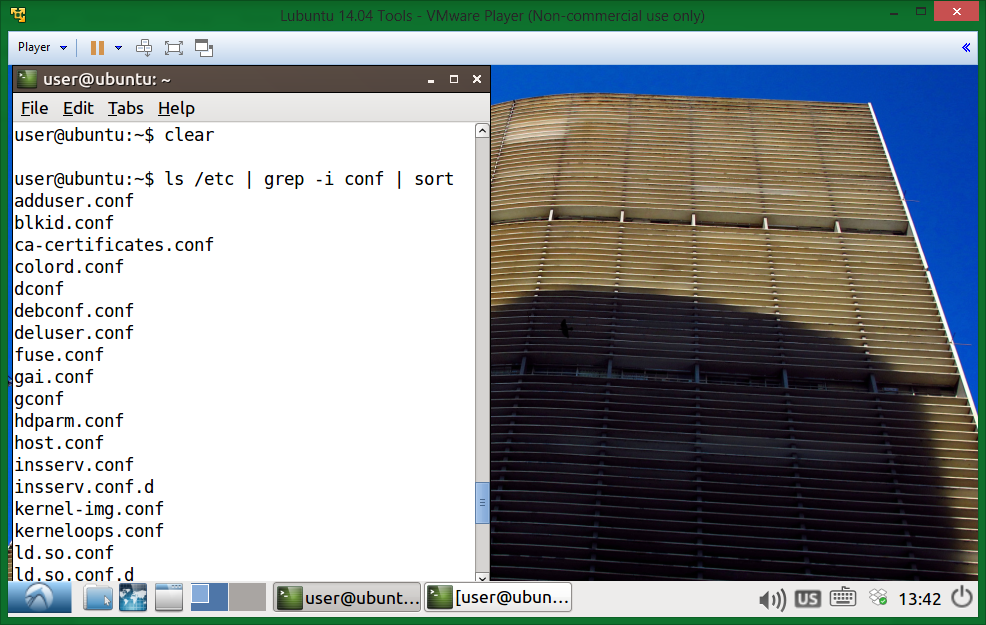


6. Create a file that contains the names of five people. Use cat and output redirection to create the file and use a here document to end the input.

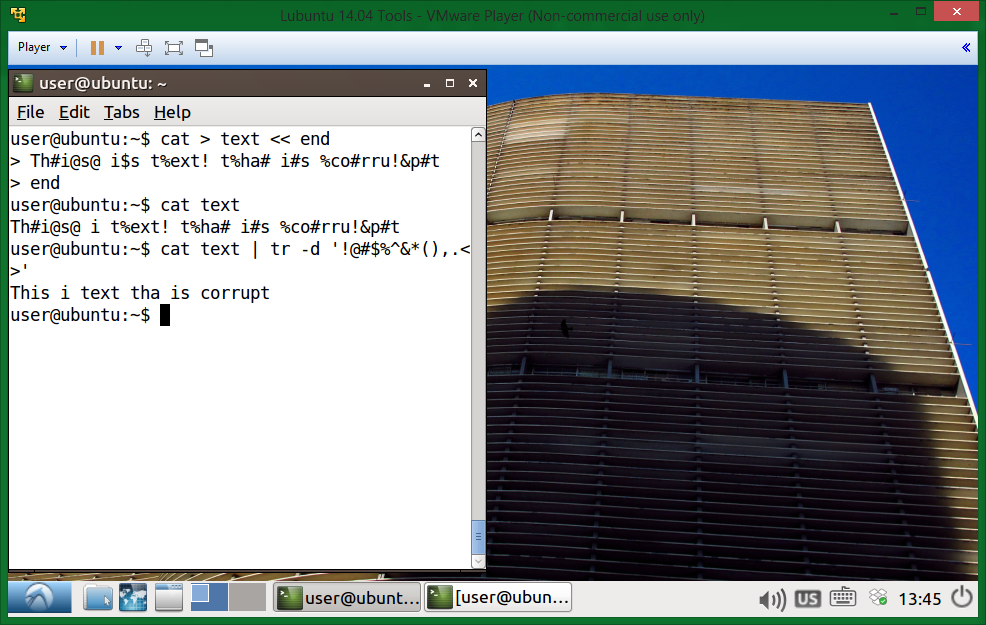


19

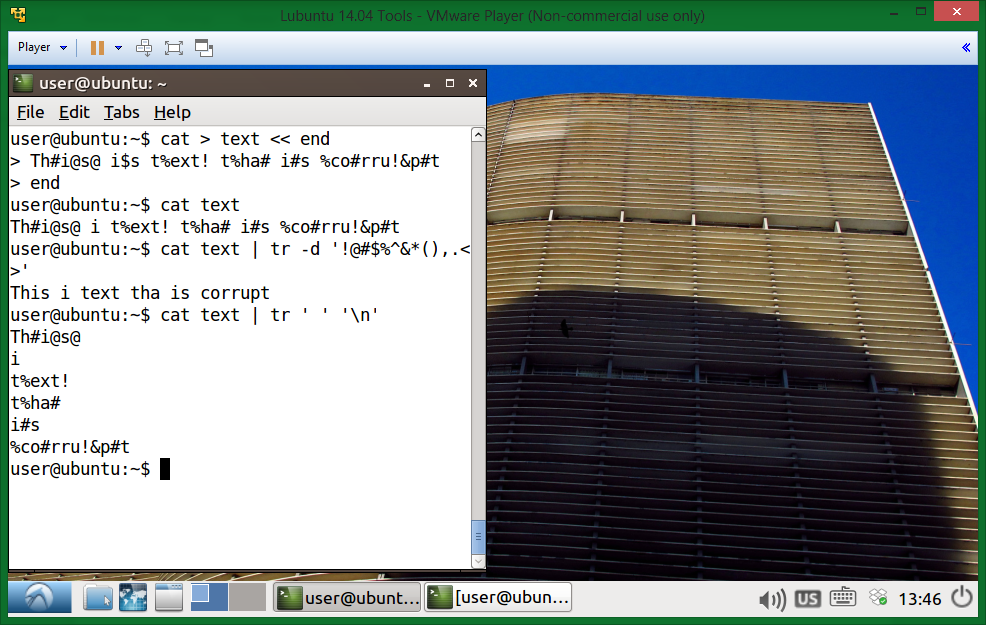
3. Make a list of all filenames in /etc that contain the string conf in their filename. 

4. Make a sorted list of all files in /etc that contain the case insensitive string conf in their filename. 

6. Write a line that removes all non-letters from a stream.



7. Write a line that receives a text file, and outputs all words on a separate line.



# CHAPTER 20

1. Explain the difference between these two commands. This question is very important. If you don't know the answer, then look back at the shell chapter.

find /data -name "\*.txt"

find /data -name \*.txt

The quotes preserve what is found, while without quotes it could be treated as a variable and ‘expanded’

2. Explain the difference between these two statements. Will they both work when there are 200 .odf files in /data? How about when there are 2 million .odf files ?

find /data -name "\*.odf" > data\_odf.txt

find /data/\*.odf > data\_odf.txt

The first one looks for file names, collects them and then outputs them to the file

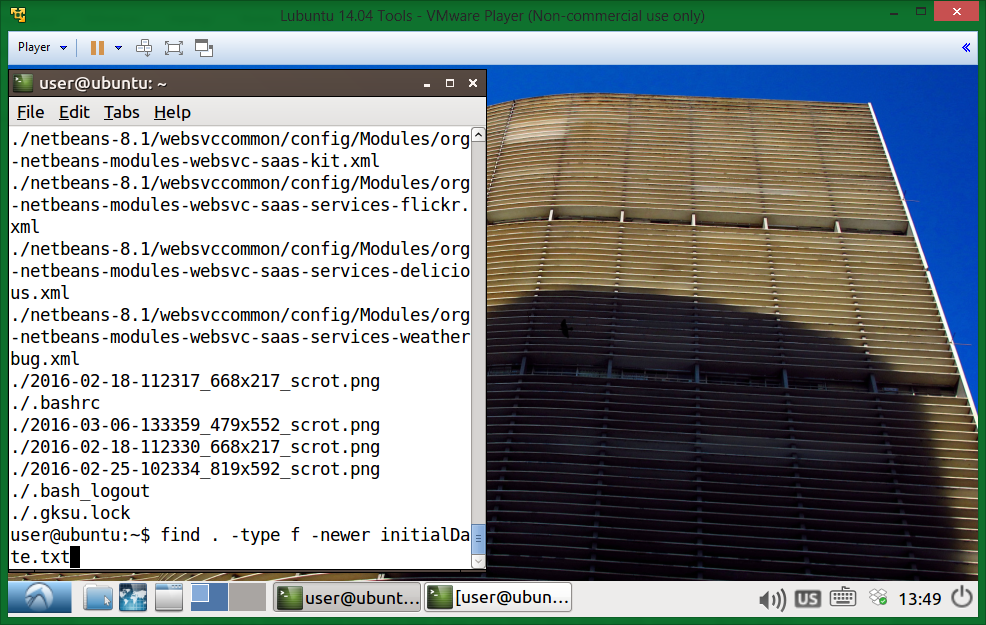
The second one looks for all files, collects them and then outputs them to the file  
They both would work for 200

They both might not work for 2 million. Depends on how long the file names would be. The second approach requires more memory because it keeps the whole files’ path.

3. Write a find command that finds all files created after January 30th 2010.

touch -t 201001302359 initialDate.txt

find . -type f -newer initialDate.txt



4. Write a find command that finds all \*.odf files created in September 2009.

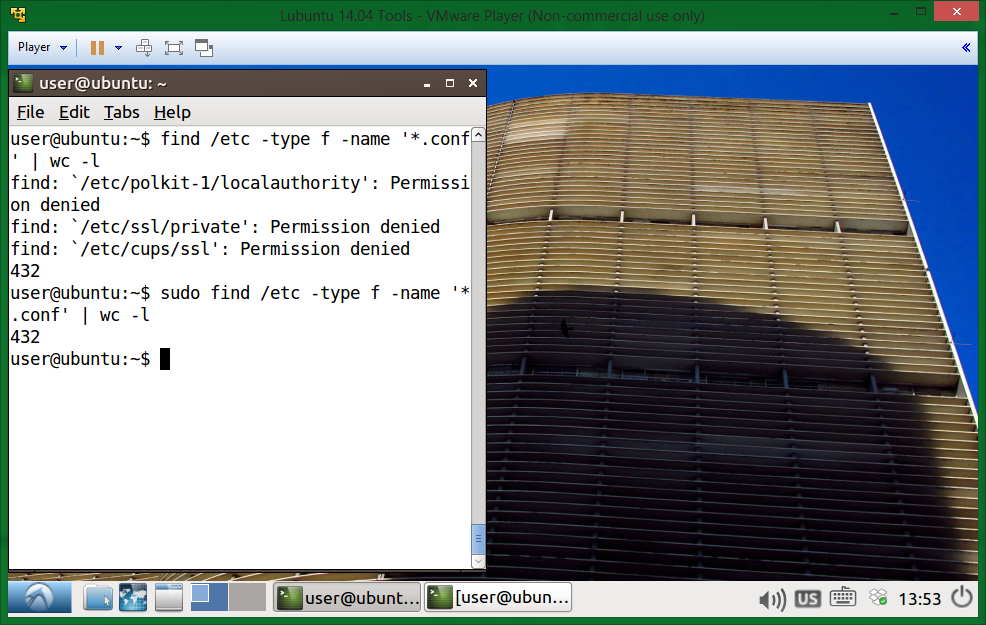
touch -t 200908312359 from.txt

touch -t 200910010000 to.txt

find . -type f -name "\*.odf" -newer from.txt ! -newer to.txt

Nothing found

5. Count the number of \*.conf files in /etc and all its subdirs.



6. Here are two commands that do the same thing: copy \*.odf files to /backup/ . What would be a reason to replace the first command with the second ? Again, this is an important question.

cp -r /data/\*.odf /backup/

find /data -name "\*.odf" -exec cp {} /backup/ \;

the first one copies recursively ALL files found while the second one copies each individually

7. Create a file called loctest.txt. Can you find this file with locate? Why not? How do you make locate find this file ?

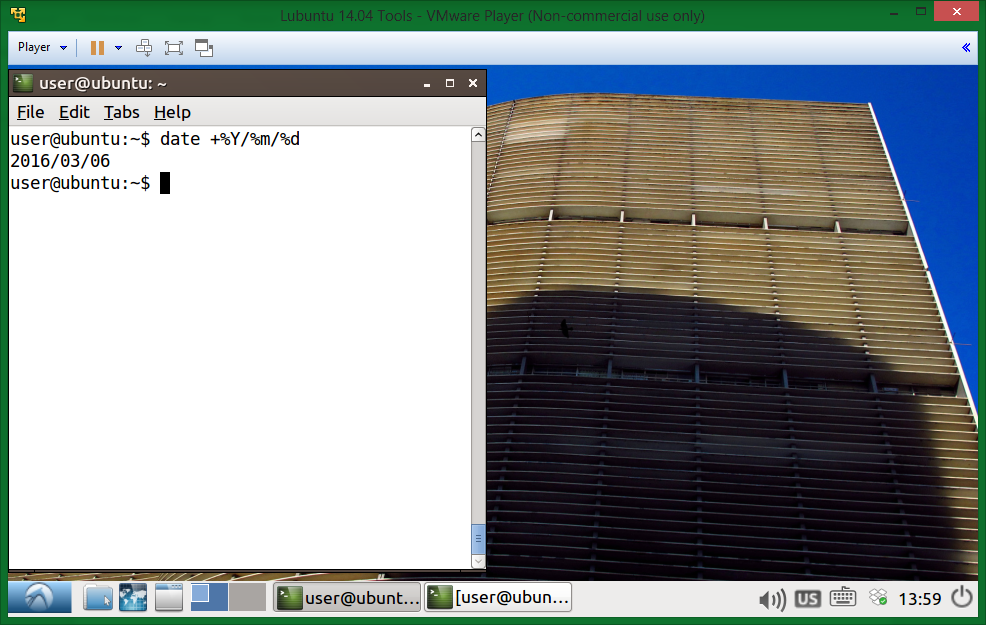
No, the file database is not up to date.

‘updatedb’ has to be called

8. Use find and -exec to rename all .htm files to .html.

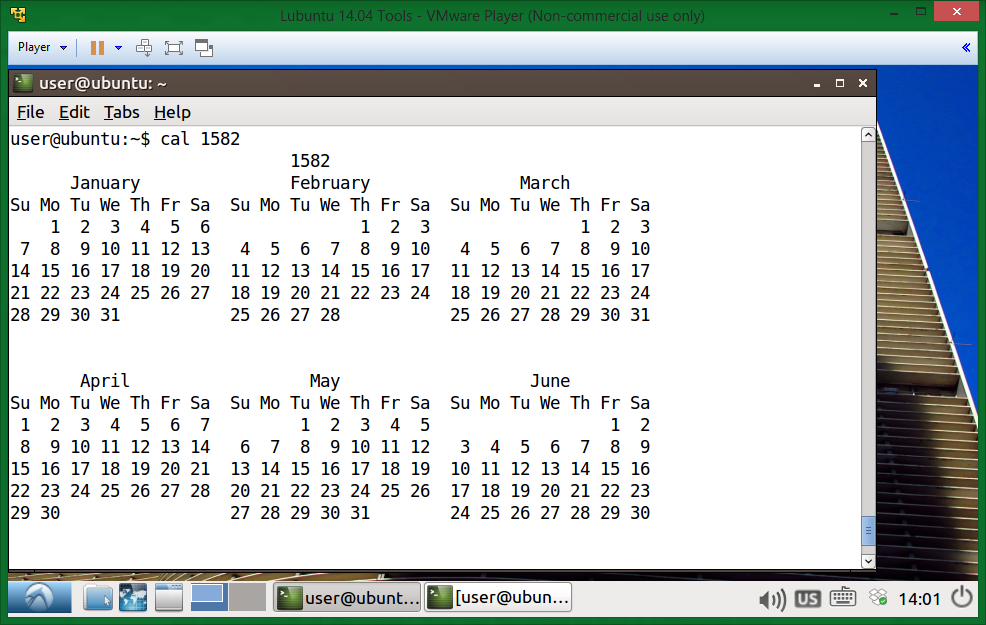
find . -name '\*.htm' -exec mv {} {}l \;

9. Issue the date command. Now display the date in YYYY/MM/DD format.



10. Issue the cal command. Display a calendar of 1582 and 1752. Notice anything special ?

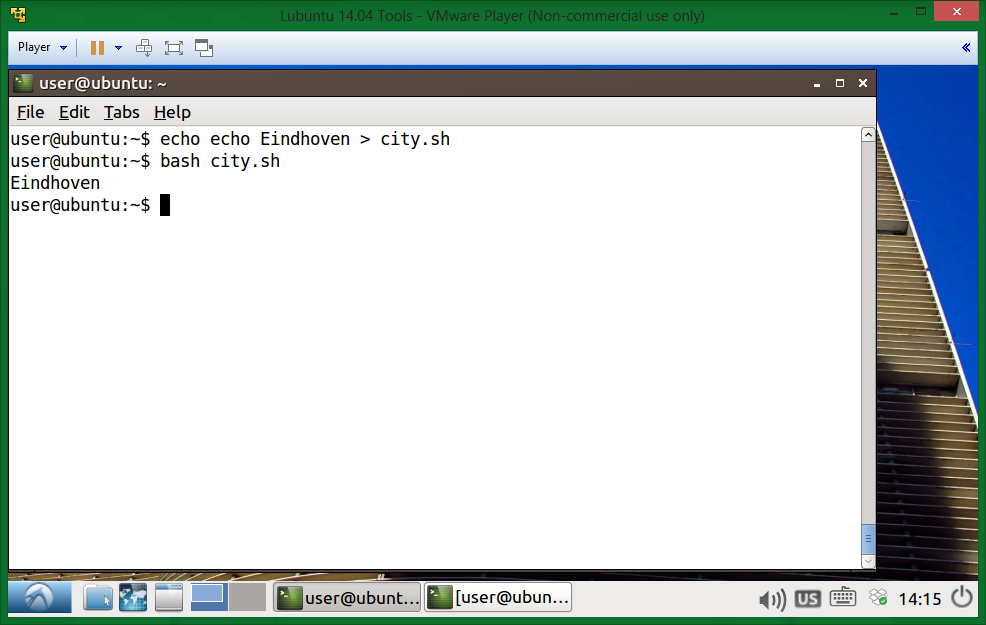
No. ‘Regional settings’ come to mind, but not in our case



23

0. Give each script a different name, keep them for later!

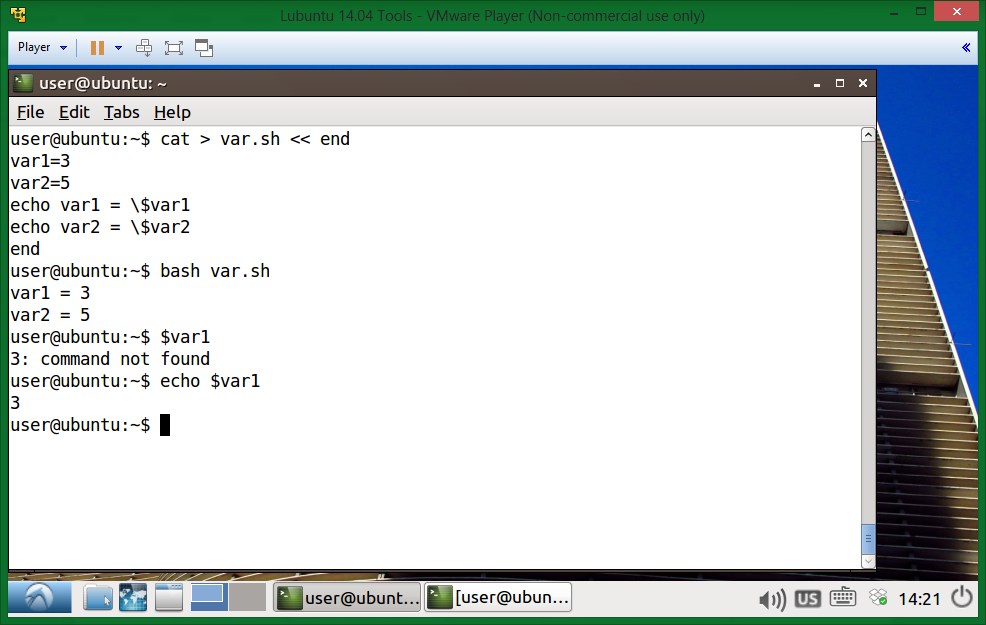
1. Write a script that outputs the name of a city.



2. Make sure the script runs in the bash shell.

3. Make sure the script runs in the Korn shell.

4. Create a script that defines two variables, and outputs their value.



5. The previous script does not influence your current shell (the variables do not exist outside of the script). Now run the script so that it influences your current shell.

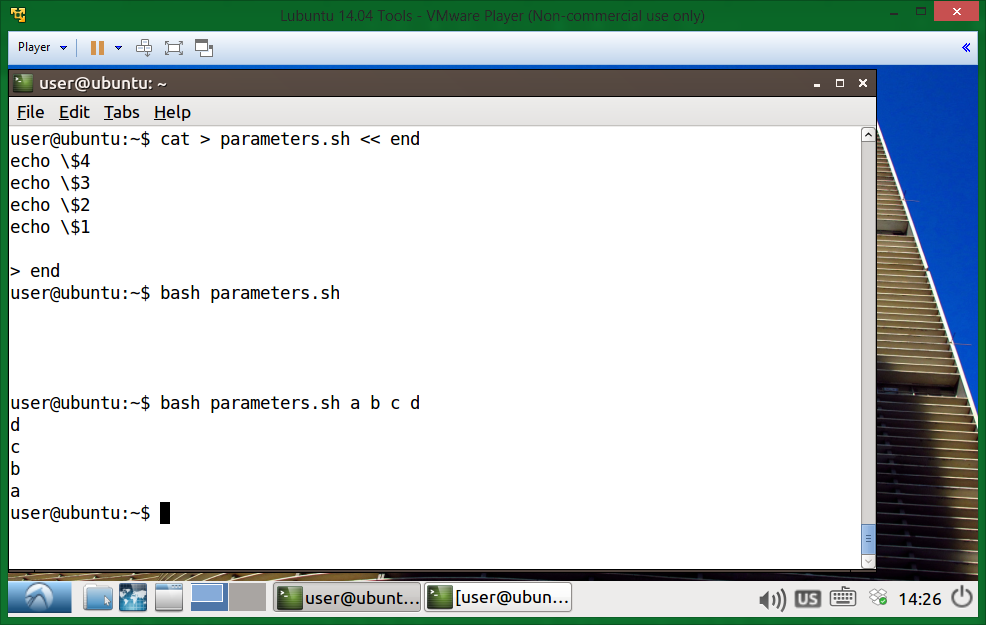
6. Is there a shorter way to Source the script ?

Bash; Source; . <scriptPath>

7. Comment your scripts so that you know what they are doing.

25

1. Write a script that receives four parameters, and outputs them in reverse order.



2. Write a script that receives two parameters (two filenames) and outputs whether those files exist.

