.......Amrita School of Engineering, Amritapuri Campus.

19CSE101: Computer Systems Essentials

**LAB SHEET 5- Operating System**

**Filters and Shell Programming**

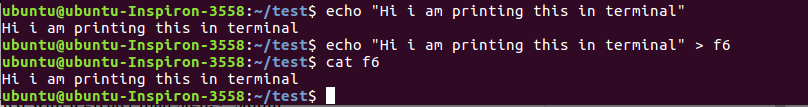
Section 5 Filters

**stdin, stdout, and stderr**

The bash shell has three basic streams; it takes input from stdin (stream 0), it sends output to stdout (stream 1) and it sends error messages to stderr. The keyboard often serves as stdin, whereas stdout and stderr both go to the display. This can be confusing to new Linux users because there is no obvious way to recognize stdout from stderr.

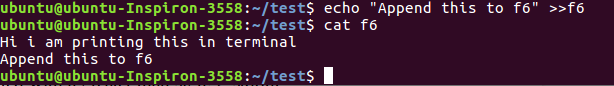
**> stdout**

stdout can be redirected with a greater than sign. > stdout

stdout can be redirected with a greater than sign. While scanning the line, the shell will see the > sign and will create a new file and the output will be written to it.

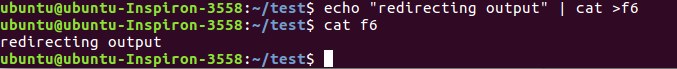
>> **append**

Use >> to append output a file.



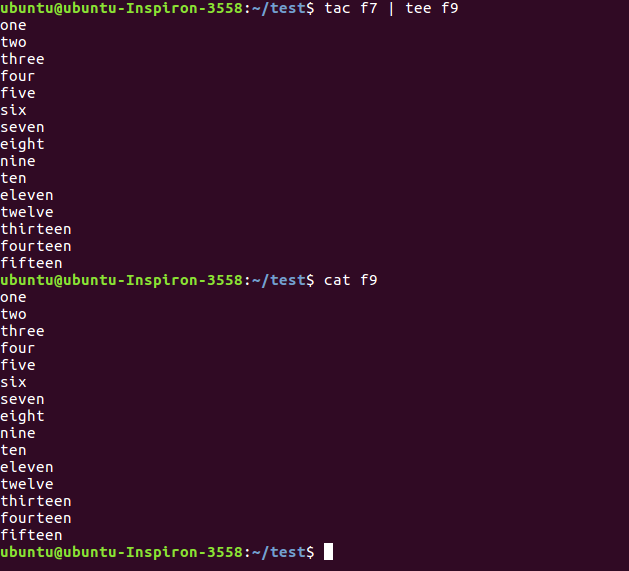
**Pipes**

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command’s output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. The command line programs that do the further processing are referred to as filters.



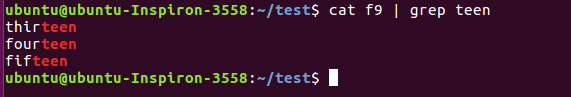
**tee**

The tee filter puts stdin on stdout and also into a file. So tee is almost the same as cat, except that it has two identical outputs.

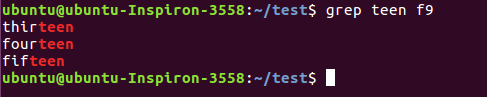


**grep**

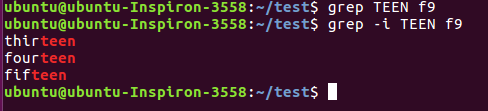
The grep filter is famous among Unix users. The most common use of grep is to filter lines of text containing (or not containing) a certain string.



You can write this without the cat.



One of the most useful options of grep is grep -i which filters in a case insensitive way.



Another very useful option is grep -v which outputs lines not matching the string.

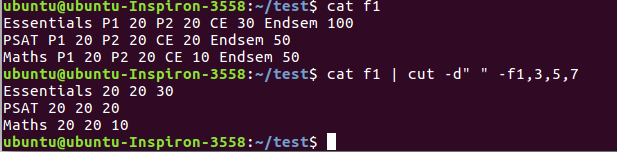


And of course, both options can be combined to filter all lines not containing a case insensitive string.

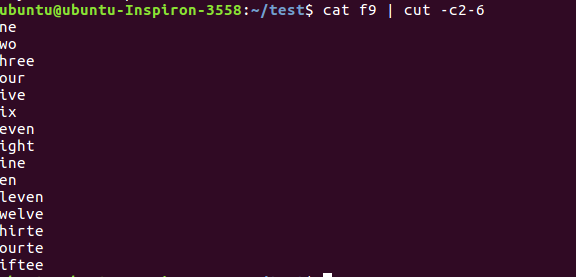


**cut**

The cut filter can select columns from files, depending on a delimiter or a count of bytes.

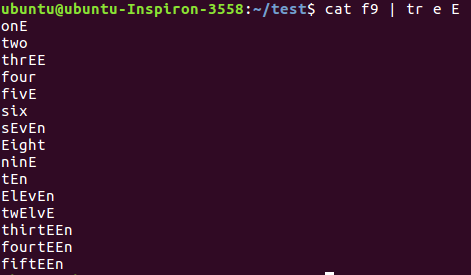


To display second to seventh character of all lines of f9

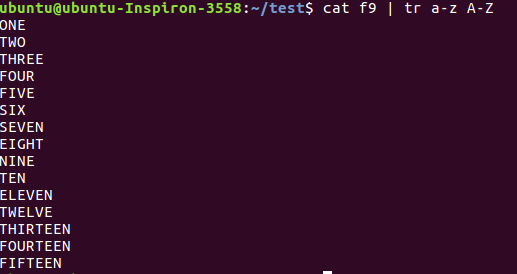


**tr**

You can translate characters with tr. The screenshot shows the translation of all occurrences of e to E.



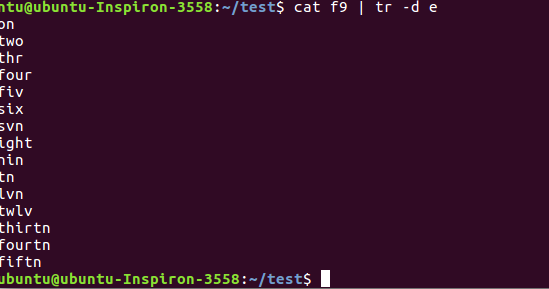
Here we set all letters to uppercase by defining two ranges.



Here we translate all newlines to spaces.

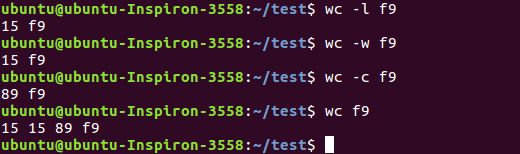


This last example uses tr -d to delete characters.



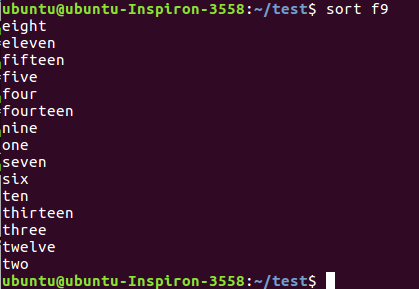
**wc**

Counting words, lines and characters is easy with wc.



**sort**

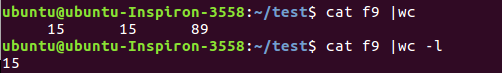
The sort filter will default to an alphabetical sort.



Section 6 Pipe Examples

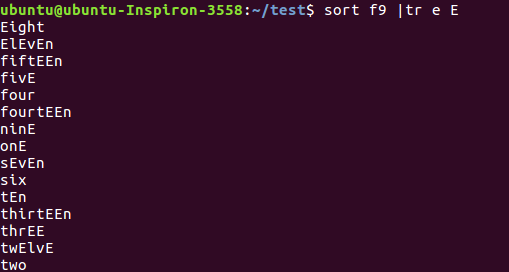
See the examples given below and understand the working of each.

Example 1



Example 2

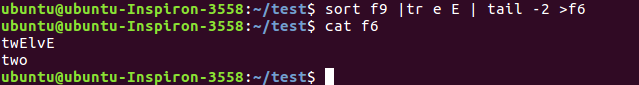
Example 3



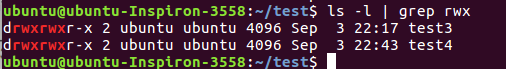
Example 4



Example 5



Example 6



Section 7 Shell scripting introduction

Shells have support for programming constructs that can be saved as scripts. These scripts in turn then become more shell commands. Many Linux commands are scripts.

**Starting with the first shell program**

Shell scripting can be defined as a group of commands executed in sequence. The steps needed for developing shell scripts are

Step 1: Open a file with .sh extension.

gedit example.sh

Step 2: All shell scripts should begin with #!/bin/bash or whatever other shell you prefer. This line is called the shebang, and although it looks like a comment, it's not: it notifies the shell of the interpreter to be used for the script. The provided path must be an absolute one (you can't just use "bash", for example), and the shebang must be located on the first line of the script without any preceding space.

Step 3: Now type your actual Shell Program you want to develop and save it. (You can use the manual uploaded in lab-1 and understand the syntax for shell program).

Our first shell script will be the usual "Hello World" routine.

#!/bin/sh

echo "Hello World"

Step 4: The next step is to make the script executable by using chmod command.

Take a terminal and type

chmod 744 example.sh

or

chmod +x example.sh

Step 5: Now you can simply run the shell program file as

./example.sh

# Lab Exercise

1. Create a file **demo** with the following contents

Student Alice Essentials 20 PSAT 22 Maths 34 Cultural 25 English 70

Student Bob Essentials 23 PSAT 21 Maths 32 CulturS al 18 English 94

Student Clara Essentials 18 PSAT 16 Maths 27 Cultural 12 English 45

Student Dirck Essentials 25 PSAT 23 Maths 48 Cultural 25 English 98

Student Eve Essentials 8 PSAT 6 Maths 12 Cultural 13 Eng

1. Find the marks obtained by Clara in all the subjects
2. Print the marks for essentials in the increasing order
3. Find the maximum marks scored in PSAT
4. Find the minimum marks obtained in Cultural
5. Save the marks obtained by all the students in maths into a file and display it in the terminal using a single command
6. Print the first 3 letters of all student names.
7. Print the contents of file demo in terminal with all alphabets in capital letters.
8. Print all student names after deleting the letter ‘a’
9. Count the number of lines, words and characters in demo file after removing the letter ‘S’
10. Write a shell program to perform the following actions in the given order.



* 1. Create a file file1 in directory Test3 with the contents same as output of the command ls -l
  2. Go to directory Test3
  3. Find the names of all files and folders in file1
  4. Find the names of all files and folders starting with d(case insensitive)
  5. Print all words of file1 on a separate line.
  6. Go back to your home directory.