.....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.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ echo "Hi i am printing this in terminal"
Hi i am printing this in terminal
ubuntu@ubuntu-Inspiron-3558:~/test$ echo "Hi i am printing this in terminal" > f6
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f6
Hi i am printing this in terminal
ubuntu@ubuntu-Inspiron-3558:~/test$
```

>> **append**

Use >> to append output a file.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ echo "Append this to f6" >>f6
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f6
Hi i am printing this in terminal
Append this to f6
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ echo "redirecting output" | cat >f6
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f6
redirecting output
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ tac f7 | tee f9
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 | grep teen
thirteen
fourteen
fifteen
ubuntu@ubuntu-Inspiron-3558:~/test$
```

You can write this without the cat.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ grep teen f9
thirteen
fourteen
fifteen
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ grep TEEN f9
ubuntu@ubuntu-Inspiron-3558:~/test$ grep -i TEEN f9
thirteen
fourteen
fifteen
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ grep -v teen f9
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
ubuntu@ubuntu-Inspiron-3558:~/test$
```

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ grep -vi TEEN f9
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
ubuntu@ubuntu-Inspiron-3558:~/test$
```

cut

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f1
Essentials P1 20 P2 20 CE 30 Endsem 100
PSAT P1 20 P2 20 CE 20 Endsem 50
Maths P1 20 P2 20 CE 10 Endsem 50
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f1 | cut -d" " -f1,3,5,7
Essentials 20 20 30
PSAT 20 20 20
Maths 20 20 10
ubuntu@ubuntu-Inspiron-3558:~/test$
```

To display second to seventh character of all lines of f9

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 | cut -c2-6
ne
wo
hree
our
ive
ix
even
ight
ine
en
leven
welve
hirte
ourte
iftee
```

tr

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 | tr e E
onE
two
thrEE
four
fivE
six
sEvEn
Eight
ninE
tEn
ElEvEn
twElvE
thirtEEn
fourtEEn
fiftEEn
```

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

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 | tr a-z A-Z
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE
TEN
ELEVEN
TWELVE
THIRTEEN
FOURTEEN
FIFTEEN
```

Here we translate all newlines to spaces.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 | tr '\n' ' '
one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen ubu
ntu@ubuntu-Inspiron-3558:~/test$
```

This last example uses tr -d to delete characters.

```
ntu@ubuntu-Inspiron-3558:~/test$ cat f9 | tr -d e
on
two
thr
four
fiv
six
svn
ight
nin
tn
lvn
twlv
thirtn
fourtn
fiftn
ubuntu@ubuntu-Inspiron-3558:~/test$
```

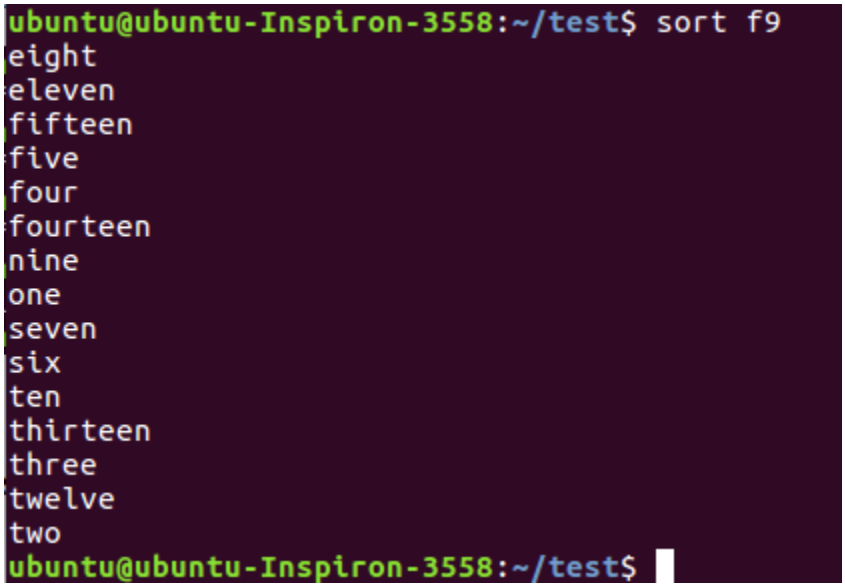
wc

Counting words, lines and characters is easy with wc.

```
ubuntu@ubuntu-Inspiron-3558:~/test$ wc -l f9
15 f9
ubuntu@ubuntu-Inspiron-3558:~/test$ wc -w f9
15 f9
ubuntu@ubuntu-Inspiron-3558:~/test$ wc -c f9
89 f9
ubuntu@ubuntu-Inspiron-3558:~/test$ wc f9
15 15 89 f9
ubuntu@ubuntu-Inspiron-3558:~/test$
```

sort

The sort filter will default to an alphabetical sort.


ubuntu@ubuntu-Inspiron-3558:~/test$ sort f9
eight
eleven
fifteen
five
four
fourteen
nine
one
seven
six
ten
thirteen
three
twelve
two
ubuntu@ubuntu-Inspiron-3558:~/test$

Section 6 Pipe Examples

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

Example 1

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 |wc
     15      15      89
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9 |wc -l
15
```

Example 2

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f9| grep -v teen | tail -4
nine
ten
eleven
twelve
ubuntu@ubuntu-Inspiron-3558:~/test$
```

Example 3

```
ubuntu@ubuntu-Inspiron-3558:~/test$ sort f9 |tr e E
Eight
ElEvEn
fiftEEn
fivE
four
fourtEEn
ninE
onE
sEvEn
six
tEn
thirtEEn
thrEE
twElvE
two
```

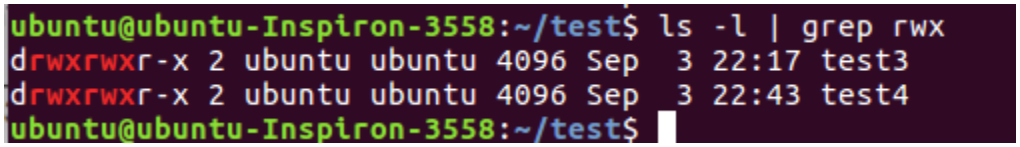
Example 4

```
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f1 | cut -d" " -f1,3,5,7 | wc -w
12
ubuntu@ubuntu-Inspiron-3558:~/test$
```

Example 5

```
ubuntu@ubuntu-Inspiron-3558:~/test$ sort f9 |tr e E | tail -2 >f6
ubuntu@ubuntu-Inspiron-3558:~/test$ cat f6
twElvE
two
ubuntu@ubuntu-Inspiron-3558:~/test$
```

Example 6


ubuntu@ubuntu-Inspiron-3558:~/test$ ls -l | grep rwx
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep 3 22:17 test3
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep 3 22:43 test4
ubuntu@ubuntu-Inspiron-3558:~/test$

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 English 5

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

    Test1 ➡ Test2 ➡ Test3

    b. Create a file file1 in directory Test3 with the contents same as output of the command ls -l
    c. Go to directory Test3
    d. Find the names of all files and folders in file1
    e. Find the names of all files and folders starting with d(case insensitive)
    f. Print all words of file1 on a separate line.
    g. Go back to your home directory.