wc (option) [file]

shows: [lines] [characters] [bytes] [filename] of a file

-c shows only bytes

-m shows only characters

-w shows only words

-l shows lines

**find command**

The **find** command is useful for locating specific files within the File System.  
**The find** command searches the directory tree starting from the directory specified by a pathname to locate

files and directories.

*Find pathname options argument*

a field is a unit of information arranged in columns

grep (option) [pattern] [file]

shows lines with a certain pattern

-v prints lines excluding pattern

-c counts the lines with the pattern

-e shows pattern 1 **or** pattern 2..

grep -e AZ -e CO temp.dat

sort (option) [filename]

shows only specific fields in a dataset, can be sorted

-u sorts and removes duplicates

-k(number) sorts lines based on a field

-n sorts numerically

-t(separator) default is a space “ “

-r sorts descending (high to low)

cut (option) [filename]

shows only specific fields within a dataset

-c specifies character positions 1-3 to show 1 through 3 or 1,3 to show 1 and 3

-d(separator) specifies separator, which is different from space

-f[field(s)] used to specify fields when -d is used to define a separator. Default is tab

paste [fileA] [fileB]

Concatenates horizontally using a space “ “ as a separator

-s concatenates vertically

-d “(separator)” designates a different separator for horizontal concatenation

sed ‘s/(word1)/(word2)/’ [filename]

replaces first instance of word1 with word2

sed ‘s/(word1)/(word2)/g’ [filename]

replaces all instances of word1 with word2

printf “%[width].[precision]type” argument

strings do not need . for spacing. Space to the right using a negative number in the first field

types:

%d decimal integer

%f floating point number

%s string

%e exponential/scientific notation

\n new line this can be included within the type to make a new line

stdout is the plain text output from a command

stderr is the plain text error output from commands

Redirect command >

Puts the stdout of a command into a file

Command (option) argument(s) > filename

If you put it into an existing file it will overwrite the existing content

>> command will append not overwrite

2> will redirect stderr into a file

/dev/null is a trash can type thing to dump errors you don’t want to keep

| is the pipeline command similar to the redirect command except it will format the command output

eg.

cat file1.dat | head -3

this will print the first 3 lines of file1.dat, you can use as many pipelines as you want. Do not repeat the argument (filename) in the pipeline

command substitution

if you enclose a command in back quotes “` `” the unshifted tilde, you can store an output into a variable

whatever is in quotes must be a unix command

this will only store one line (the last line) of output

eg

var1=`cat file1.dat`

you cannot store stderr into a variable

translate command

tr

will translate stdinput (keyboard input)

Eg

tr a-z A-Z

will translate lower case to upper case

or

tr ‘ao’ ‘tb’

this will translate any a to t and any o to b

you can also squeeze with -s

echo heeeeelloe | tr -s 'e'

will print helloe

to exit ctr+c

by using “<” as an argument you can translate file data, you can then put it back into a file using >

A screenshot of a computer program

Description automatically generated

You can use tr as a pipeline

is a programming language designed for text processing and used as a data extraction and reporting tool

awk ‘condition {print action}’ filename

you can extract lines based on conditions that you specify on fields

you can print specific fields {print action}

Fields are specified by $n

$1 is first field

$2 is second field

Default separator is white space(s) OR tab(s)

You can pipeline awk as normal

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

If you do xxx || xxx && xxx without parentheses this && will have priority so it would act like

xxx || ( xxx && xxx )

awk ‘condition {print}’ filename

will print the lines with certain conditions within a file

Print action

awk ‘condition {print action}’ filename

awk ‘condition {print $2, $6}’ filename

this will print 2nd and 6th field of lines with the condition

you can also preform arithmetic operations on fields

awk ‘condition {print $7 + $8}’ filename

will print the sum of field 7 and 8 within each line of a file

you can use printf to format in awk, the syntax is slightly different than in bash

awk ‘condition {printf “format”, $field}’ filename

the format notation is the same “%.2f” will do a float with 2 digit precision

you can also output text

awk ‘condition {print “Hello”, $1}’ filename

will print “Hello” then the first field in every line

you can even format this text with printf

awk ‘condition {printf “%s %.2f\n”, “Hello”, $1}’ filename

use -F to specify separator

awk -F: ‘condition {print action}’ filename

will use a colon to separate fields

you can use variable in awk with -v

var=FP2

awk -F, -v var="$var" '$2 == var {print}' eeg1.dat

for i in 2 5 8 do

sum=$(( sum + $i ))  
echo variable i is $i echo variable sum is $sum

done

s1=abc

s2=def

if [ $s1 != $s2 ] #Test if the strings s1 and s2 are not identical.

then

echo strings $s1 and $s2 are different

fi

if TEST-COMMAND1 then

statements1

elif TEST-COMMAND2

then

else

fi

Python

Run python files:

python3 filename

Import math #generic import, will import the entire module if you use generic import you must do math.sqrt

from math import sqrt,cos #will import just the sqrt and cos function

You can rename libraries

Import math as m

#so you can use

m.sqrt(100)

#data types

Have value type and identity

Value is their value

Type is the data type

type(x) will show

Identity is where on the memory it’s stored

id(x) will show the memory address

if you assign a variable as the value of another variable, it is simply referencing the same place in memory

x=45

y=x

#id(x) will return the same value as id(y)

a = b = c = 1

this will create one object pointing to the same place in memory

|  |
| --- |
| **Example Data type** |
| i = 20 int |
| f = 20.5 float |
| s = "Hello" str |
| L = ["apple", "banana", "cherry"] list |
| T= ("apple", "banana", "cherry") tuple |
| d= {"name" : "John", "age" : 36} dict |

printf in python

arg1=10000

arg2=schni

print(“Hello this is my result %.2e %s” %(arg1, arg2))

Input

number=input(“Enter a number: “)

by default all inputs will be a string

number=float(input(“enter a number: “))

this will convert the input to a float

a=10.34\*10\*\*8

a=10.34\*pow(10,8)

these will both make 10.34E8