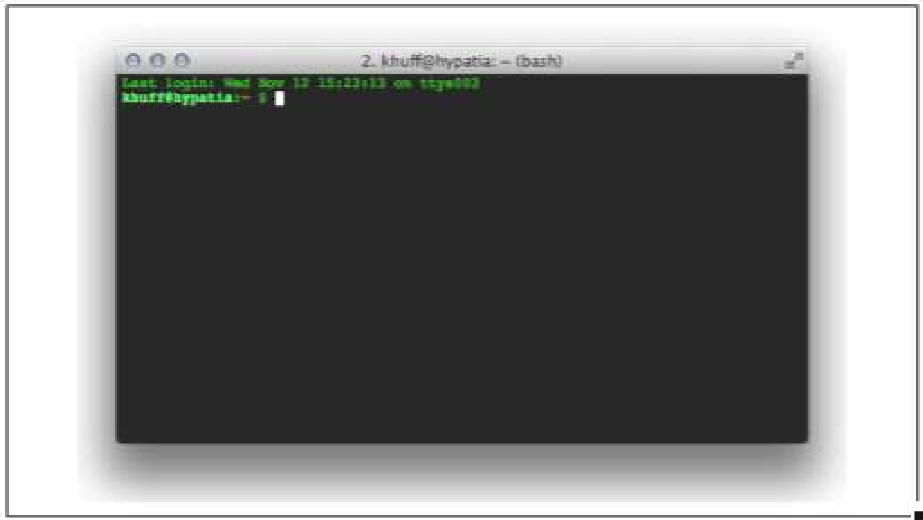
Linux command line (the shell)



```
Last login: Wed Nov 12 15:23:13 on ttys002
khuff@hypatia:~ $
```

- the shell is a programing language
- it has variables and states
- uses obtuse syntax including special characters

Terminal in Linux or OS/X

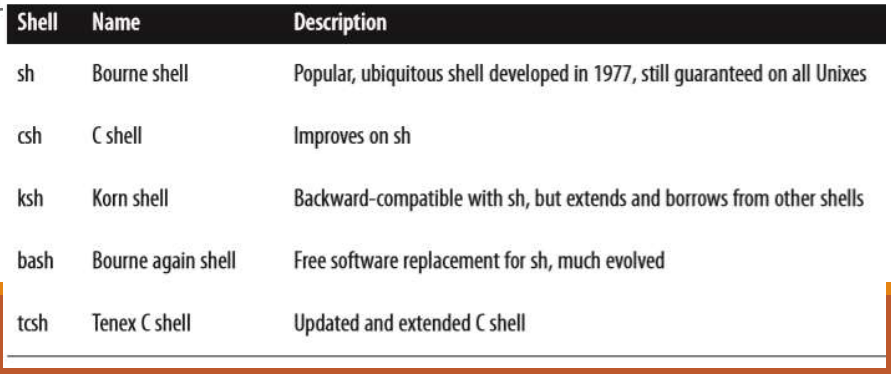
- For windows download Ubuntu

  (alternate bash shells)

  - MobaXterm (https://mobaxterm.mobatek.net)
  - Gitbash (https://git-scm.com/download/win)

if you don’t have bash, install it now.

different shells



Shell	Name	Description
sh	Bourne shell	Popular, ubiquitous shell developed in 1977, still guaranteed on all Unixes
csh	C shell	Improves on sh
ksh	Korn shell	Backward-compatible with sh, but extends and borrows from other shells
bash	Bourne again shell	Free software replacement for sh, much evolved
tcsh	Tenex C shell	Updated and extended C shell

Warning.

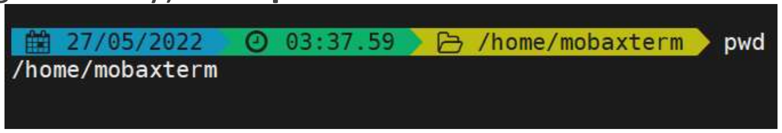
The following lecture is *super* dry.

It contains a bunch of information that you will not remember.

very basic shell commands (pwd, ls, and cd)

when opening a terminal you start in your “home directory” mine is /c/Users/damien, the shortcut is the tilde, ~

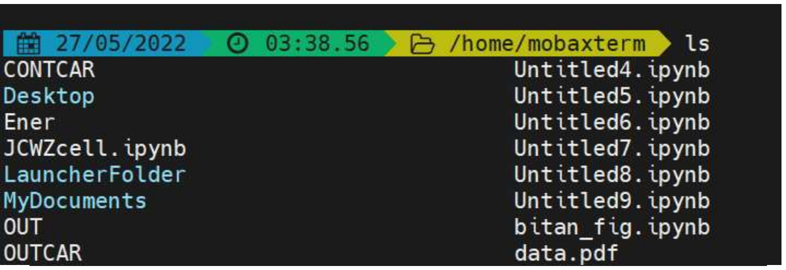
- (print working directory) : use **pwd** command to determine current directory.



```
 27/05/2022   03:37.59   /home/mobaxterm   pwd
/home/mobaxterm
```

to see output which has scrolled off-screen can typically use shift+pg up(down) to look through output buffer.

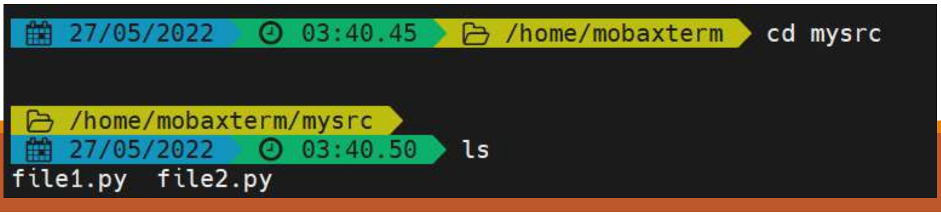
- (listing): use **ls** to show contents of current directory.



```
 27/05/2022   03:38.56   /home/mobaxterm   ls
CONTCAR                                  Untitled4.ipynb
Desktop                                  Untitled5.ipynb
Ener                                     Untitled6.ipynb
JCWZcell.ipynb                           Untitled7.ipynb
LauncherFolder                           Untitled8.ipynb
MyDocuments                              Untitled9.ipynb
OUT                                      bitan_fig.ipynb
OUTCAR                                   data.pdf
```

* is a wildcard.
$ **ls** Un* (would show any file/directory which begins with Un)

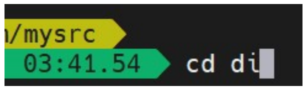
- (change directory): use **cd** to change the current directory.



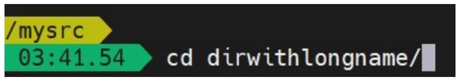
```
 27/05/2022   03:40.45   /home/mobaxterm   cd mysrc

 /home/mobaxterm/mysrc
 27/05/2022   03:40.50   ls
file1.py  file2.py
```

tab completion is nice!



```
/mysrc
03:41.54   cd di
```



```
/mysrc
03:41.54   cd dirwithlongname/
```

directory structure and shortcuts

Syntax	Meaning
/	The root, or top-level, directory of the filesystem (also used for separating the names of directories in paths)
~	The home directory
.	Current working directory
..	Parent of current working directory
../..	The parent of the parent of current directory

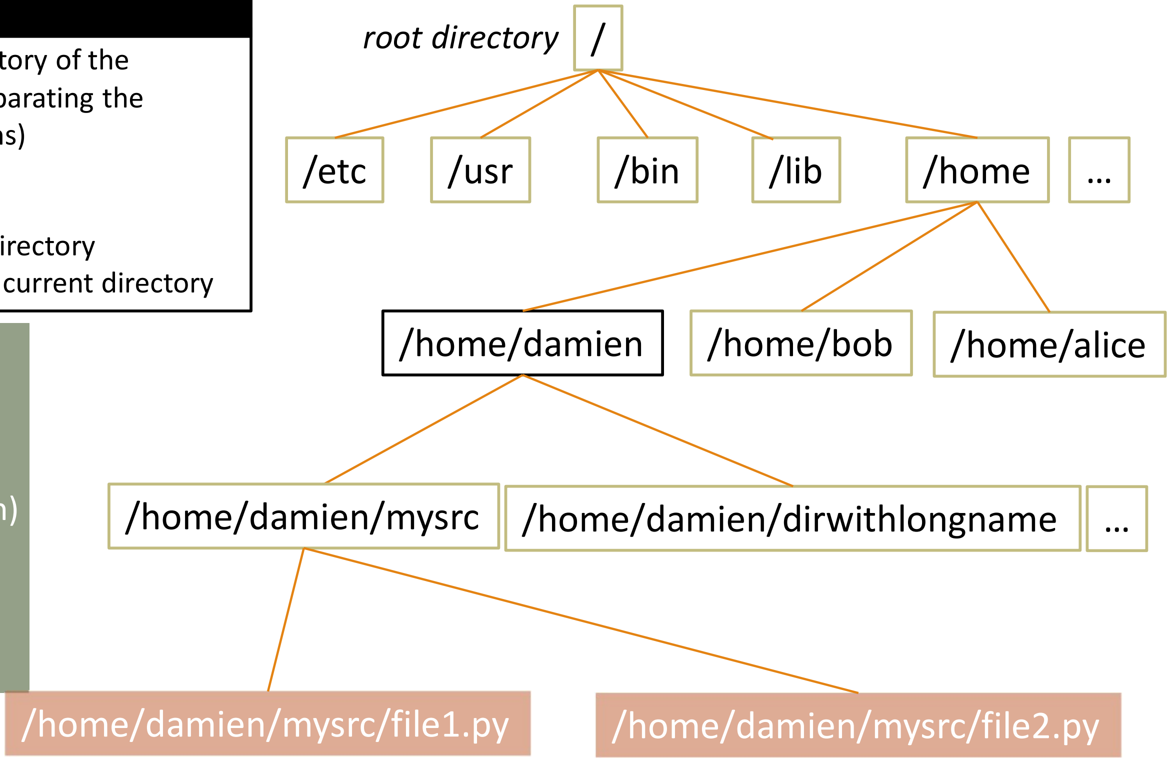
Absolute path:
/home/damien/mysrc/file1.py

Relative path:

(if current dir is /home/damien)

./mysrc/file1.py

or ../damien/mysrc/file1.py



continued

pwd	ls
cd	mkdir
cp	mv
man	less
echo	>

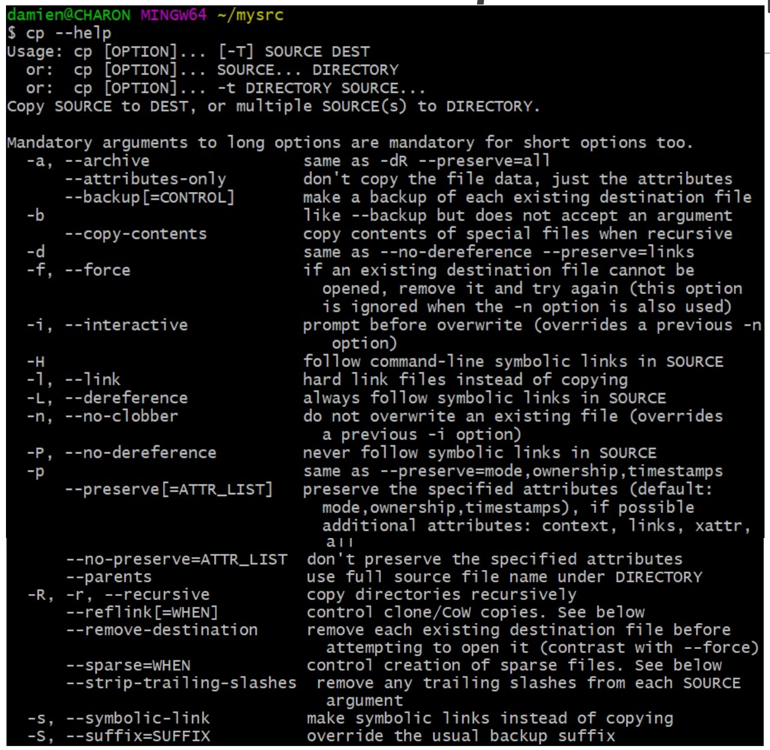
- **mkdir** *directoryname* (makes a new directory in the current directory)
- **cp** *filetocopy newfile* (makes a copy of “filetocopy” and names it “newfile”)
- **mv** *sourcefile newfile* (moves sourcefile to newfile, can be a used to rename file or put in a different directory)
- **man** *command* (brings up manual on how to use command)
- **apropos** *keyword* (equivalent to **man –k**, searches the manual for commands whose descriptions contain the keyword/s) e.g. $ apropos “text editor”
- **less** *filename* (displays text of filename to screen one page at a time, can use pg-up/down, search for expression with /*expr*, q to quit)
- **echo** *“type something”* (displays what you type to the screen)
- **>** (**redirects** the output which would go the screen and sends it somewhere else, e.g. a file or a device.)

  $ **echo** “hello world!” **>** hworld.txt

  (creates new file, “hworld.txt” which contains the text “hello world!”.)

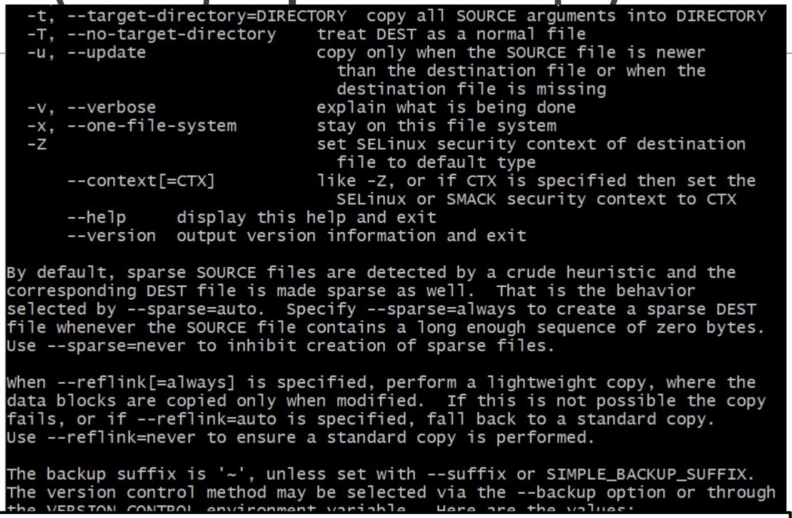
command --help (usually shows usage and command line arguments)

note I’m just giving the simplest use cases. $man cp (or $cp --help)



```
damien@CHARON MINGW64 ~/mysrc
$ cp --help
Usage: cp [OPTION]... [-T] SOURCE DEST
  or:  cp [OPTION]... SOURCE... DIRECTORY
  or:  cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
  -a, --archive                same as -dR --preserve=all
      --attributes-only        don't copy the file data, just the attributes
      --backup[=CONTROL]       make a backup of each existing destination file
  -b                           like --backup but does not accept an argument
      --copy-contents          copy contents of special files when recursive
  -d                           same as --no-dereference --preserve=links
  -f, --force                  if an existing destination file cannot be
                                 opened, remove it and try again (this option
                                 is ignored when the -n option is also used)
  -i, --interactive            prompt before overwrite (overrides a previous -n
                                  option)
  -H                           follow command-line symbolic links in SOURCE
  -l, --link                   hard link files instead of copying
  -L, --dereference            always follow symbolic links in SOURCE
  -n, --no-clobber             do not overwrite an existing file (overrides
                                 a previous -i option)
  -P, --no-dereference         never follow symbolic links in SOURCE
  -p                           same as --preserve=mode,ownership,timestamps
      --preserve[=ATTR_LIST]   preserve the specified attributes (default:
                                 mode,ownership,timestamps), if possible
                                 additional attributes: context, links, xattr,
                                 all
      --no-preserve=ATTR_LIST  don't preserve the specified attributes
      --parents                use full source file name under DIRECTORY
  -R, -r, --recursive          copy directories recursively
      --reflink[=WHEN]         control clone/CoW copies. See below
      --remove-destination     remove each existing destination file before
                                 attempting to open it (contrast with --force)
      --sparse=WHEN            control creation of sparse files. See below
      --strip-trailing-slashes  remove any trailing slashes from each SOURCE
                                 argument
  -s, --symbolic-link          make symbolic links instead of copying
  -S, --suffix=SUFFIX          override the usual backup suffix
```



```
  -t, --target-directory=DIRECTORY  copy all SOURCE arguments into DIRECTORY
  -T, --no-target-directory    treat DEST as a normal file
  -u, --update                 copy only when the SOURCE file is newer
                                 than the destination file or when the
                                 destination file is missing
  -v, --verbose                explain what is being done
  -x, --one-file-system        stay on this file system
  -Z                           set SELinux security context of destination
                                 file to default type
      --context[=CTX]          like -Z, or if CTX is specified then set the
                                 SELinux or SMACK security context to CTX
      --help     display this help and exit
      --version  output version information and exit

By default, sparse SOURCE files are detected by a crude heuristic and the
corresponding DEST file is made sparse as well.  That is the behavior
selected by --sparse=auto.  Specify --sparse=always to create a sparse DEST
file whenever the SOURCE file contains a long enough sequence of zero bytes.
Use --sparse=never to inhibit creation of sparse files.

When --reflink[=always] is specified, perform a lightweight copy, where the
data blocks are copied only when modified.  If this is not possible the copy
fails, or if --reflink=auto is specified, fall back to a standard copy.
Use --reflink=never to ensure a standard copy is performed.

The backup suffix is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.
The version control method may be selected via the --backup option or through
the VERSION_CONTROL environment variable.  Here are the values:
```

[] indicate something optional, … can be more than one

other common use cases

cp –ip *filetocopy newfile* (no auto-overwrite)

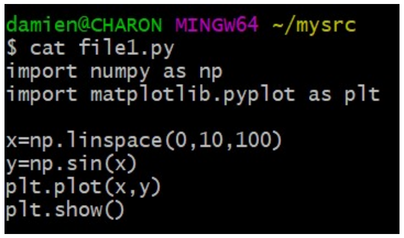
cp *file1 file2 file3 directory* (copy the three files to directory)

cp *f* directory* (copy all files that start with “f” to directory)

cp *-r sourcedir newdir* (copy entire source_dir to new_dir)

other useful commands

- **nano, vim, emacs** (terminal based text editors, nano is by far the simplest)
- **cat** (displays entire file to screen)



```
damien@CHARON MINGW64 ~/mysrc
$ cat file1.py
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(0,10,100)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```

can be used on multiple files, e.g.
$ **cat** file1.py file2.py
displays them one after another.

- **>** (redirect the output)

  $ **cat** file1.py file2.py **>** file3.py

>> will append the output to the end of a file instead of overwriting it.

- **grep** *expression filename* (outputs to screen lines from filename which contain expression)
- command1 **|** command2 (the ***pipe***. output of command1 is redirected to the input of command 2, extremely useful.)

◦ **chmod** code *filename (changes permissions on file, who can read/write/execute.)*

e.g, **chmod** *+x filename (make file executable)*

chmod *–w filename (remove write permissions)*

You can make a <u>bash script</u> by simply by making a text file containing bash commands executable (**chmod** +x). They will be executed as if typed in terminal. Bash even supports basic loops, e.g.

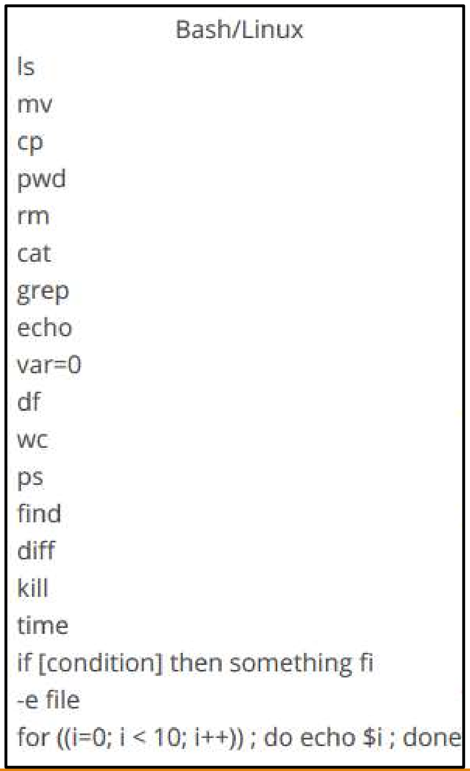
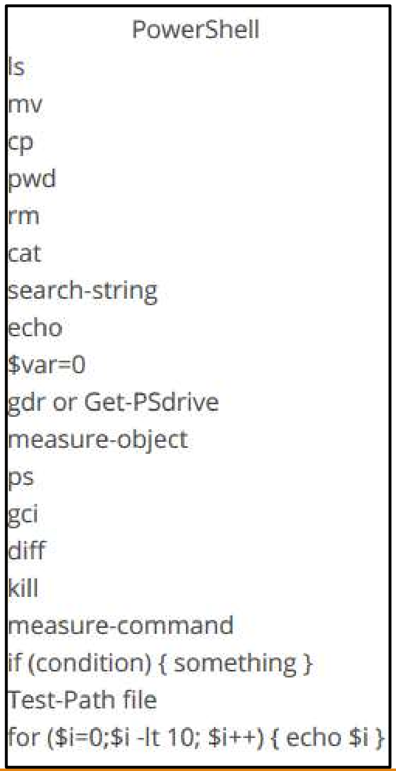
For loop in bash

```
for i in {1..100};do echo $i;done                (displays 1 to 100 to screen)
for i in {1..10};do cp file_temp file$i;done     (makes 10 copies of file_temp named:
                                                   file1, file2, file3,…,and file10)
for var in `cat test.txt`;do echo $var;done      (displays each space delimited string in file
                                                   test.txt to screen, each on its own line)
```

In the above examples both i and var are variables, they are dereferenced by using $. Variables can be created at anytime at the command line (e,g, myvar=5 or hrm="hello").

The shell is a full-fledged programming language, with arrays, conditionals, and loops

Windows powershell

Bash/Linux	PowerShell
ls	ls
mv	mv
cp	cp
pwd	pwd
rm	rm
cat	cat
grep	search-string
echo	echo
var=0	$var=0
df	gdr or Get-PSdrive
wc	measure-object
ps	ps
find	gci
diff	diff
kill	kill
time	measure-command
if [condition] then something fi	if (condition) { something }
-e file	Test-Path file
for ((i=0; i < 10; i++)) ; do echo $i ; done	for ($i=0;$i -lt 10; $i++) { echo $i }

- echo, >, |, mkdir, rm, mv, ls, pwd, cd, more (common linux command similar to less) all work in the same way as in bash shell.
- grep is missing (but sls has similar functionality)
- There are also alternatives to sed and (g)awk.

combining | with sed

- **sed** s/(stream editor, can perform text transformations on file or input from pipe. It is quite powerful, but steep learning curve)

  simple search and replace:

  sed ‘**s/**texttoreplace**/**replacement**/g**’

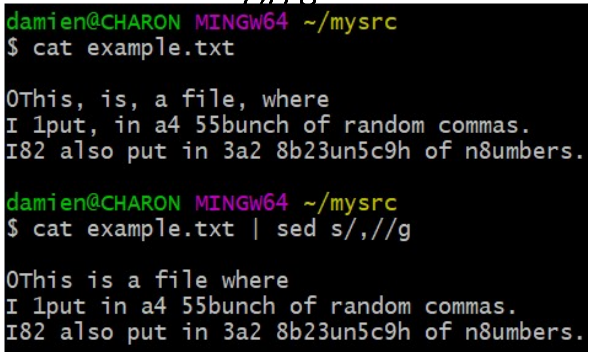
  replace every occurrence on each line

  replace a regular expression (regex):

  sed ‘**s/**[0-9]**/**replacement**/g**’

  matches every single-digit number

sed s/,//g file

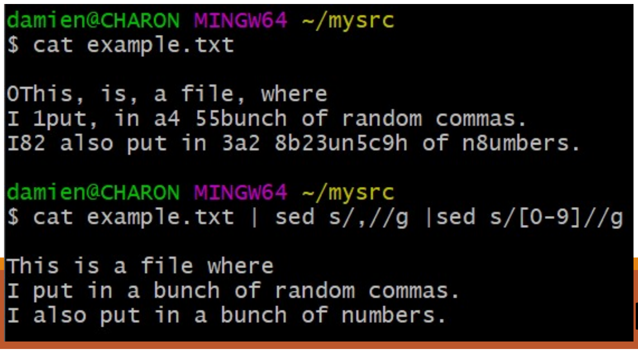


```
damien@CHARON MINGW64 ~/mysrc
$ cat example.txt

0This, is, a file, where
I 1put, in a4 55bunch of random commas.
I82 also put in 3a2 8b23un5c9h of n8umbers.

damien@CHARON MINGW64 ~/mysrc
$ cat example.txt | sed s/,//g

0This is a file where
I 1put in a4 55bunch of random commas.
I82 also put in 3a2 8b23un5c9h of n8umbers.
```



```
damien@CHARON MINGW64 ~/mysrc
$ cat example.txt

0This, is, a file, where
I 1put, in a4 55bunch of random commas.
I82 also put in 3a2 8b23un5c9h of n8umbers.

damien@CHARON MINGW64 ~/mysrc
$ cat example.txt | sed s/,//g |sed s/[0-9]//g

This is a file where
I put in a bunch of random commas.
I also put in a bunch of numbers.
```

some regex examples:

a.c	(will match any 3 chars that starts with a and ends with b, abc, aac, etc)
ab{2,4}	(will match abb, abbb, and abbbb)
ab*	(matches a,ab,abb,abbb,…)

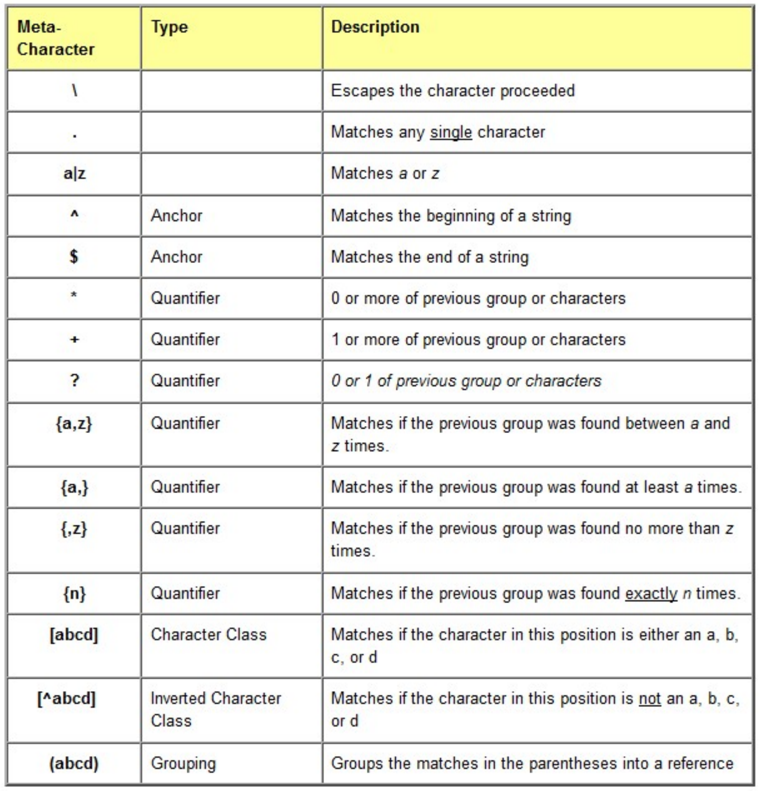
note: sed is greedy. will match largest it can.

http://www.grymoire.com/Unix/Sed.html

https://regexone.com (interactive tutorial on regular expressions)

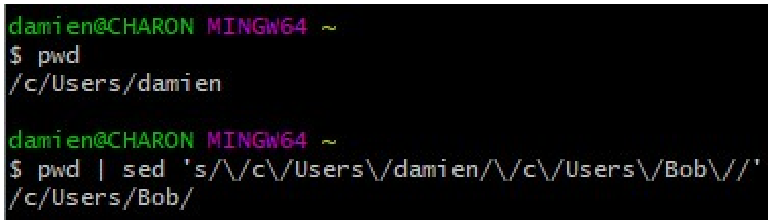
regular expressions

https://regexlearn.com/learn/regex101 (another tutorial)



Meta-Character	Type	Description
\		Escapes the character proceeded
.		Matches any single character
a|z		Matches *a* or *z*
^	Anchor	Matches the beginning of a string
$	Anchor	Matches the end of a string
*	Quantifier	0 or more of previous group or characters
+	Quantifier	1 or more of previous group or characters
?	Quantifier	*0 or 1 of previous group or characters*
{a,z}	Quantifier	Matches if the previous group was found between *a* and *z* times.
{a,}	Quantifier	Matches if the previous group was found at least *a* times.
{,z}	Quantifier	Matches if the previous group was found no more than *z* times.
{n}	Quantifier	Matches if the previous group was found exactly *n* times.
[abcd]	Character Class	Matches if the character in this position is either an a, b, c, or d
[^abcd]	Inverted Character Class	Matches if the character in this position is not an a, b, c, or d
(abcd)	Grouping	Groups the matches in the parentheses into a reference

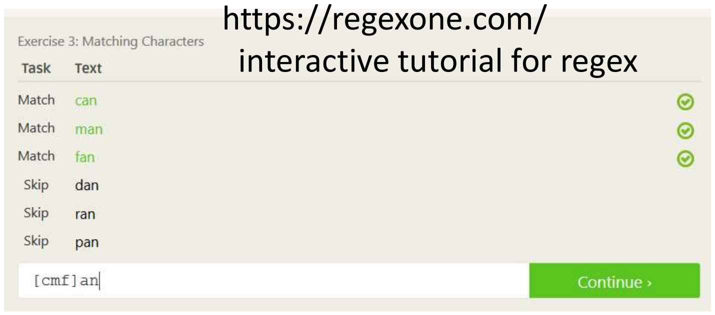
These are only some of the basic special chars associated with regex. They can get quite complicated pretty quickly.



```
damien@CHARON MINGW64 ~
$ pwd
/c/Users/damien

damien@CHARON MINGW64 ~
$ pwd | sed 's/\/c\/Users\/damien/\/c\/Users\/Bob\//'
/c/Users/Bob/
```

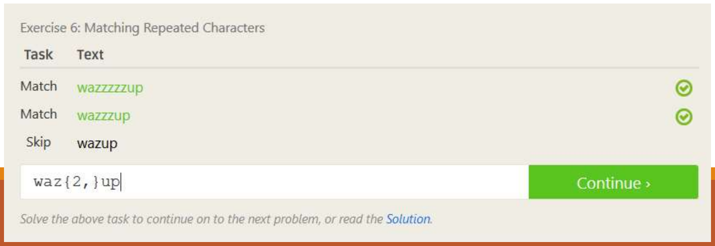
https://regexone.com/
interactive tutorial for regex



Exercise 3: Matching Characters

Task	Text
Match	can
Match	man
Match	fan
Skip	dan
Skip	ran
Skip	pan

```
[cmf]an
```



Exercise 6: Matching Repeated Characters

Task	Text
Match	wazzzzzup
Match	wazzzup
Skip	wazup

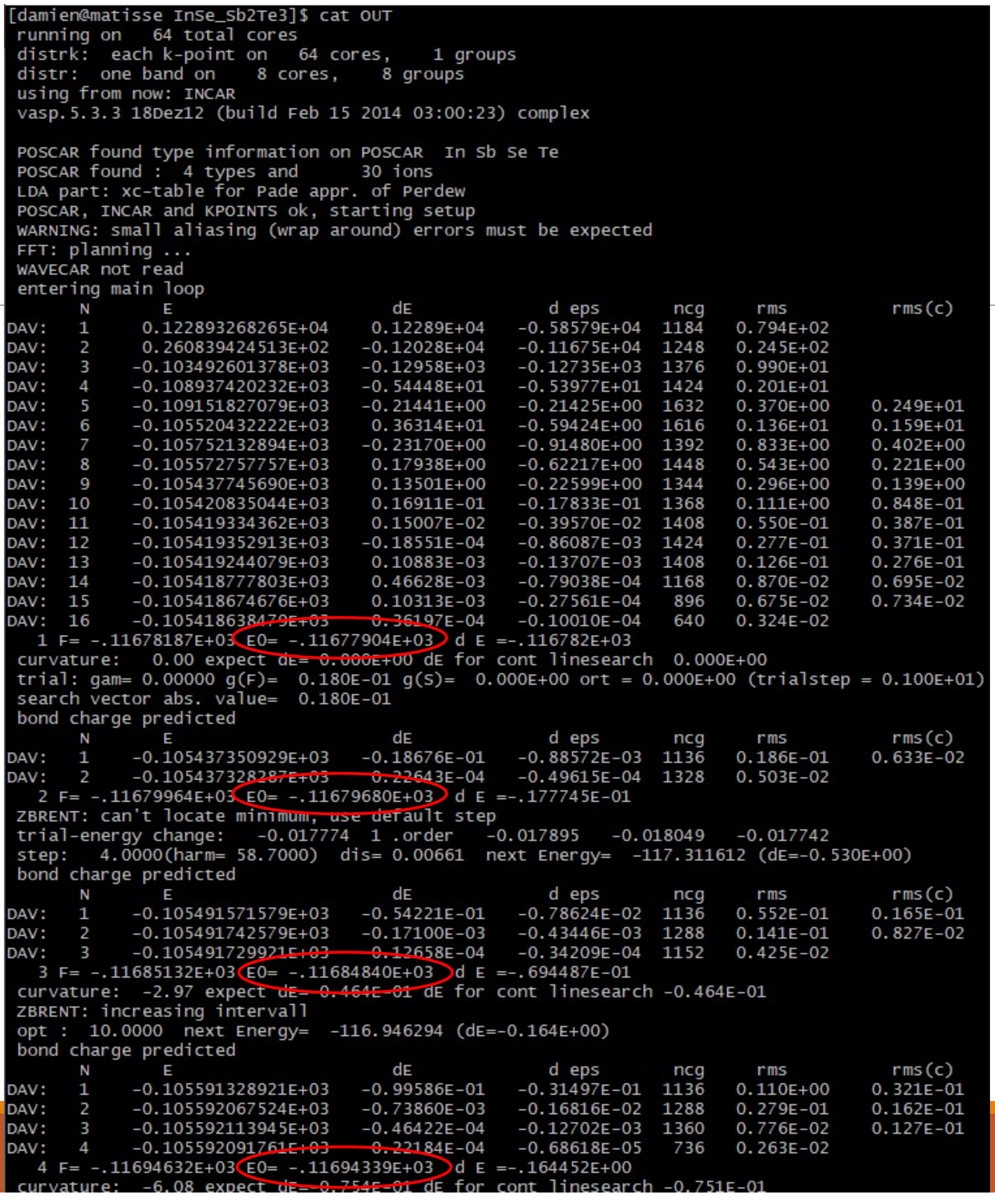
```
waz{2,}up
```

Solve the above task to continue on to the next problem, or read the Solution.

A fairly common example

I have a big output file which is not in any conventional format and I want to generate a plot of the value of “E0”.

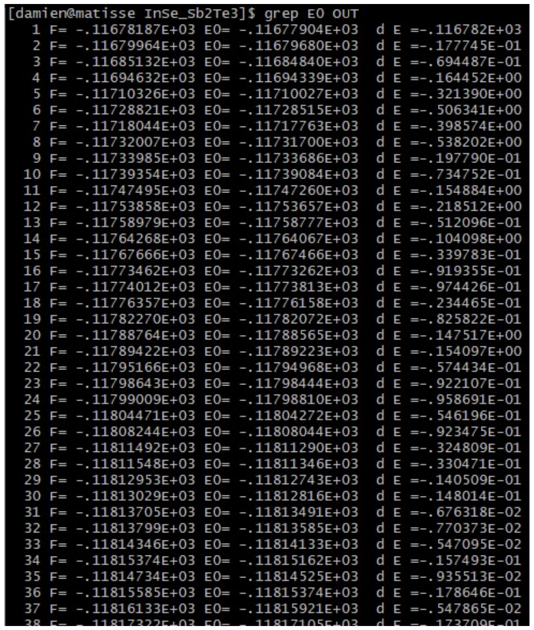
Things like this can be done very quickly by stringing together greps and sed or gawk.



```
[damien@matisse InSe_Sb2Te3]$ cat OUT
 running on   64 total cores
 distrk:  each k-point on   64 cores,    1 groups
 distr:  one band on    8 cores,    8 groups
 using from now: INCAR
 vasp.5.3.3 18Dez12 (build Feb 15 2014 03:00:23) complex

 POSCAR found type information on POSCAR  In Sb Se Te
 POSCAR found :  4 types and      30 ions
 LDA part: xc-table for Pade appr. of Perdew
 POSCAR, INCAR and KPOINTS ok, starting setup
 WARNING: small aliasing (wrap around) errors must be expected
 FFT: planning ...
 WAVECAR not read
 entering main loop
       N       E                     dE             d eps       ncg     rms          rms(c)
DAV:   1     0.122893268265E+04    0.12289E+04   -0.58579E+04  1184   0.794E+02
DAV:   2     0.260839424513E+02   -0.12028E+04   -0.11675E+04  1248   0.245E+02
DAV:   3    -0.103492601378E+03   -0.12958E+03   -0.12735E+03  1376   0.990E+01
DAV:   4    -0.108937420232E+03   -0.54448E+01   -0.53977E+01  1424   0.201E+01
DAV:   5    -0.109151827079E+03   -0.21441E+00   -0.21425E+00  1632   0.370E+00    0.249E+01
DAV:   6    -0.105520432222E+03    0.36314E+01   -0.59424E+00  1616   0.136E+01    0.159E+01
DAV:   7    -0.105752132894E+03   -0.23170E+00   -0.91480E+00  1392   0.833E+00    0.402E+00
DAV:   8    -0.105572757757E+03    0.17938E+00   -0.62217E+00  1448   0.543E+00    0.221E+00
DAV:   9    -0.105437745690E+03    0.13501E+00   -0.22599E+00  1344   0.296E+00    0.139E+00
DAV:  10    -0.105420835044E+03    0.16911E-01   -0.17833E-01  1368   0.111E+00    0.848E-01
DAV:  11    -0.105419334362E+03    0.15007E-02   -0.39570E-02  1408   0.550E-01    0.387E-01
DAV:  12    -0.105419352913E+03   -0.18551E-04   -0.86087E-03  1424   0.277E-01    0.371E-01
DAV:  13    -0.105419244079E+03    0.10883E-03   -0.13707E-03  1408   0.126E-01    0.276E-01
DAV:  14    -0.105418777803E+03    0.46628E-03   -0.79038E-04  1168   0.870E-02    0.695E-02
DAV:  15    -0.105418674676E+03    0.10313E-03   -0.27561E-04   896   0.675E-02    0.734E-02
DAV:  16    -0.105418638479E+03    0.36197E-04   -0.10010E-04   640   0.324E-02
   1 F= -.11678187E+03 E0= -.11677904E+03  d E =-.116782E+03
 curvature:   0.00 expect dE= 0.000E+00 dE for cont linesearch  0.000E+00
 trial: gam= 0.00000 g(F)=  0.180E-01 g(S)=  0.000E+00 ort = 0.000E+00 (trialstep = 0.100E+01)
 search vector abs. value=  0.180E-01
 bond charge predicted
       N       E                     dE             d eps       ncg     rms          rms(c)
DAV:   1    -0.105437350929E+03   -0.18676E-01   -0.88572E-03  1136   0.186E-01    0.633E-02
DAV:   2    -0.105437328287E+03    0.22643E-04   -0.49615E-04  1328   0.503E-02
   2 F= -.11679964E+03 E0= -.11679680E+03  d E =-.177745E-01
 ZBRENT: can't locate minimum, use default step
 trial-energy change:   -0.017774  1 .order   -0.017895   -0.018049   -0.017742
 step:   4.0000(harm= 58.7000)  dis= 0.00661  next Energy=  -117.311612 (dE=-0.530E+00)
 bond charge predicted
       N       E                     dE             d eps       ncg     rms          rms(c)
DAV:   1    -0.105491571579E+03   -0.54221E-01   -0.78624E-02  1136   0.552E-01    0.165E-01
DAV:   2    -0.105491742579E+03   -0.17100E-03   -0.43446E-03  1288   0.141E-01    0.827E-02
DAV:   3    -0.105491729921E+03    0.12658E-04   -0.34209E-04  1152   0.425E-02
   3 F= -.11685132E+03 E0= -.11684840E+03  d E =-.694487E-01
 curvature:  -2.97 expect dE=-0.464E-01 dE for cont linesearch -0.464E-01
 ZBRENT: increasing intervall
 opt :  10.0000  next Energy=  -116.946294 (dE=-0.164E+00)
 bond charge predicted
       N       E                     dE             d eps       ncg     rms          rms(c)
DAV:   1    -0.105591328921E+03   -0.99586E-01   -0.31497E-01  1136   0.110E+00    0.321E-01
DAV:   2    -0.105592067524E+03   -0.73860E-03   -0.16816E-02  1288   0.279E-01    0.162E-01
DAV:   3    -0.105592113945E+03   -0.46422E-04   -0.12702E-03  1360   0.776E-02    0.127E-01
DAV:   4    -0.105592091761E+03    0.22184E-04   -0.68618E-05   736   0.263E-02
   4 F= -.11694632E+03 E0= -.11694339E+03  d E =-.164452E+00
 curvature:  -6.08 expect dE=-0.754E-01 dE for cont linesearch -0.751E-01
```

$**grep** E0 OUT



```
[damien@matisse InSe_Sb2Te3]$ grep E0 OUT
   1 F= -.11678187E+03 E0= -.11677904E+03  d E =-.116782E+03
   2 F= -.11679964E+03 E0= -.11679680E+03  d E =-.177745E-01
   3 F= -.11685132E+03 E0= -.11684840E+03  d E =-.694487E-01
   4 F= -.11694632E+03 E0= -.11694339E+03  d E =-.164452E+00
   5 F= -.11710326E+03 E0= -.11710027E+03  d E =-.321390E+00
   6 F= -.11728821E+03 E0= -.11728515E+03  d E =-.506341E+00
   7 F= -.11718044E+03 E0= -.11717763E+03  d E =-.398574E+00
   8 F= -.11732007E+03 E0= -.11731700E+03  d E =-.538202E+00
   9 F= -.11733985E+03 E0= -.11733686E+03  d E =-.197790E-01
  10 F= -.11739354E+03 E0= -.11739084E+03  d E =-.734752E-01
  11 F= -.11747495E+03 E0= -.11747260E+03  d E =-.154884E+00
  12 F= -.11753858E+03 E0= -.11753657E+03  d E =-.218512E+00
  13 F= -.11758979E+03 E0= -.11758777E+03  d E =-.512096E-01
  14 F= -.11764268E+03 E0= -.11764067E+03  d E =-.104098E+00
  15 F= -.11767666E+03 E0= -.11767466E+03  d E =-.339783E-01
  16 F= -.11773462E+03 E0= -.11773262E+03  d E =-.919355E-01
  17 F= -.11774012E+03 E0= -.11773813E+03  d E =-.974426E-01
  18 F= -.11776357E+03 E0= -.11776158E+03  d E =-.234465E-01
  19 F= -.11782270E+03 E0= -.11782072E+03  d E =-.825822E-01
  20 F= -.11788764E+03 E0= -.11788565E+03  d E =-.147517E+00
  21 F= -.11789422E+03 E0= -.11789223E+03  d E =-.154097E+00
  22 F= -.11795166E+03 E0= -.11794968E+03  d E =-.574434E-01
  23 F= -.11798643E+03 E0= -.11798444E+03  d E =-.922107E-01
  24 F= -.11799009E+03 E0= -.11798810E+03  d E =-.958691E-01
  25 F= -.11804471E+03 E0= -.11804272E+03  d E =-.546196E-01
  26 F= -.11808244E+03 E0= -.11808044E+03  d E =-.923475E-01
  27 F= -.11811492E+03 E0= -.11811290E+03  d E =-.324809E-01
  28 F= -.11811548E+03 E0= -.11811346E+03  d E =-.330471E-01
  29 F= -.11812953E+03 E0= -.11812743E+03  d E =-.140509E-01
  30 F= -.11813029E+03 E0= -.11812816E+03  d E =-.148014E-01
  31 F= -.11813705E+03 E0= -.11813491E+03  d E =-.676318E-02
  32 F= -.11813799E+03 E0= -.11813585E+03  d E =-.770373E-02
  33 F= -.11814346E+03 E0= -.11814133E+03  d E =-.547095E-02
  34 F= -.11815374E+03 E0= -.11815162E+03  d E =-.157493E-01
  35 F= -.11814734E+03 E0= -.11814525E+03  d E =-.935513E-02
  36 F= -.11815585E+03 E0= -.11815374E+03  d E =-.178646E-01
  37 F= -.11816133E+03 E0= -.11815921E+03  d E =-.547865E-02
  38 F= -.11817322E+03 E0= -.11817105E+03  d E =-.173709E-01
```

We can us **sed**

.* is a wildcard for sed which will match anything

$ **grep** E0 OUT | **sed** s/.*E0=//|sed s/d.*//

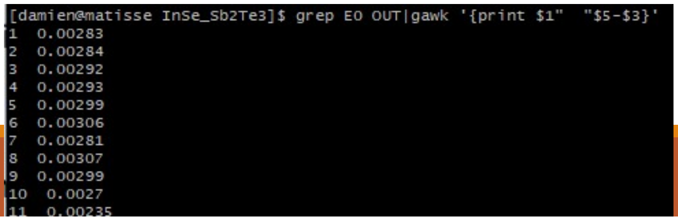
But **gawk** is the better tool here.

value is in 5th column

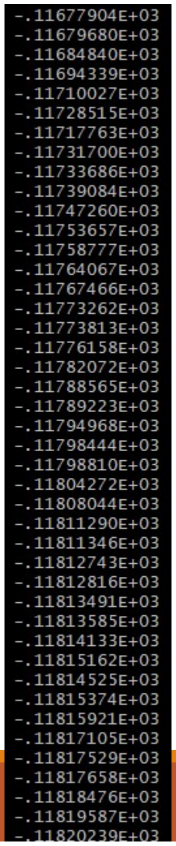
$ **grep** E0 OUT | **gawk** ‘{print $5}’

can also output multiple columns and perform some basic math on them. e.g,

$ **grep** E0 OUT | **gawk** ‘{print $1” “$3-$5}’



```
[damien@matisse InSe_Sb2Te3]$ grep E0 OUT|gawk '{print $1"  "$5-$3}'
1  0.00283
2  0.00284
3  0.00292
4  0.00293
5  0.00299
6  0.00306
7  0.00281
8  0.00307
9  0.00299
10  0.0027
11  0.00235
```



```
-.11677904E+03
-.11679680E+03
-.11684840E+03
-.11694339E+03
-.11710027E+03
-.11728515E+03
-.11717763E+03
-.11731700E+03
-.11733686E+03
-.11739084E+03
-.11747260E+03
-.11753657E+03
-.11758777E+03
-.11764067E+03
-.11767466E+03
-.11773262E+03
-.11773813E+03
-.11776158E+03
-.11782072E+03
-.11788565E+03
-.11789223E+03
-.11794968E+03
-.11798444E+03
-.11798810E+03
-.11804272E+03
-.11808044E+03
-.11811290E+03
-.11811346E+03
-.11812743E+03
-.11812816E+03
-.11813491E+03
-.11813585E+03
-.11814133E+03
-.11815162E+03
-.11814525E+03
-.11815374E+03
-.11815921E+03
-.11817105E+03
-.11817529E+03
-.11817658E+03
-.11818476E+03
-.11819587E+03
-.11820239E+03
```

connecting to other computers (ssh and scp)

Open a terminal on a **Remote Computer**

(secure shell): use **ssh username@remotehostname** to connect to a remote host. After password authentication you should have a remote shell.

Copy a file from a **Remote Computer** to your own computer

(secure copy): **scp [optional arguments]** *source_file destination_file*

- if one of the files is remote, it should be specified as username@remotehostname:/path/to/file

for example,

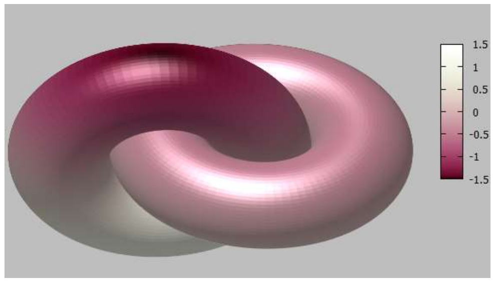
$$\textbf{scp}\ \underbrace{\textit{damien}}_{\text{username on remote computer}}@\underbrace{\textit{komodo.phys.rpi.edu:}}_{\text{remote computer name}}\underbrace{\textit{\textasciitilde/myfile.txt}}_{\text{full path to file on remote computer}}\ \ \underbrace{\textit{coolfile.txt}}_{\text{name you want to give copied file on your computer}}$$

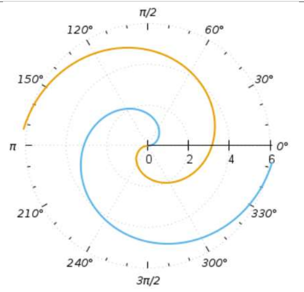
copies file “myfile.txt” from my home directory at komodo.phys.rpi.edu to my current working directory on my local computer and gives it the name “coolfile.txt”.

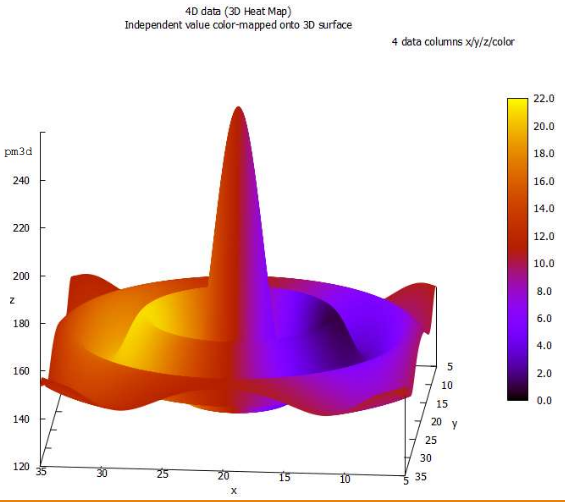
gnuplot

http://gnuplot.sourceforge.net/demo/

```
# set terminal pngcairo  transparent enhanced font "arial,10" fontscale 1.0 size 600, 400
# set output 'pm3d_lighting.2.png'
unset border
set style fill   solid 1.00 noborder
set dummy u, v
unset key
set style increment default
set object  1 rect from screen 0, 0 to screen 1, 1
set object  1 behind clip lw 1.0  dashtype solid fc  rgb "gray"  fillstyle   solid 1.00 border lt -1
set parametric
set view 236, 339, 1.245, 1
set isosamples 75, 75
unset xtics
unset ytics
unset ztics
set title "PM3D surfaces with specular highlighting"
set urange [ -3.14159 : 3.14159 ] noreverse nowriteback
set vrange [ -3.14159 : 3.14159 ] noreverse nowriteback
set xrange [ * : * ] noreverse writeback
set x2range [ * : * ] noreverse writeback
set yrange [ * : * ] noreverse writeback
set y2range [ * : * ] noreverse writeback
set zrange [ * : * ] noreverse writeback
set cbrange [ * : * ] noreverse writeback
set rrange [ * : * ] noreverse writeback
set pm3d depthorder
set pm3d lighting primary 0.5 specular 0.6
set palette rgbformulae 8, 9, 7
slice(x,y) = (x**2+y**2 < 10.0) ? 1.0 : (x**2+y**2 > 300.0) ? NaN : sin(abs(atan2(x,y)))
sinc2(x,y) = sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
flatten(x,y) = sqrt(x**2+y**2)/5.
F(x,y) =  sinc2(x,y) * slice(x,y) * flatten(x,y)
## Last datafile plotted: "++"
splot cos(u)+.5*cos(u)*cos(v),sin(u)+.5*sin(u)*cos(v),.5*sin(v) with pm3d,     1+cos(u)+.5*cos(u)*cos(v),.5*sin(v),sin(u)+.5*sin(u)*cos(v) with pm3d
```







gnuplot has its own scripting language for producing high quality plots.

widely used in physics.

we will just only touch on it here.

quick plot in gnuplot

help	**exit**
plot	**replot**
set	**unset**

can plot a function or data from file.

plot a function:

gnuplot> **plot** sin(x)

plot data:

gnuplot> **plot** “data.txt” (default is ugly)

gnuplot> **plot** “data.txt” pointtype 7 (better)

gnuplot> **plot** “data.txt” with lines

Things you can set

set xrange [-pi:pi]

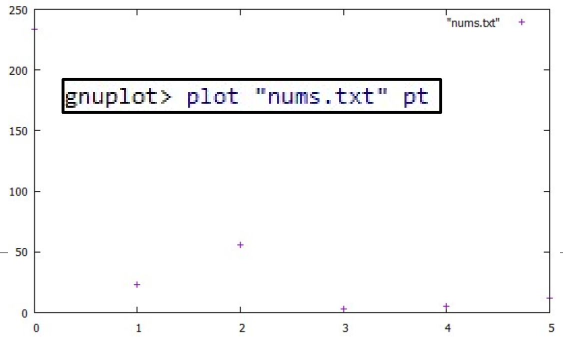
set yrange [-2:2.5]

set gridlines

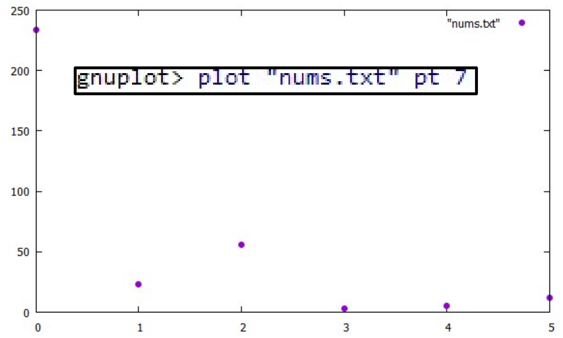
set xtics <start>,<incr>,<end>

set terminal … lots more

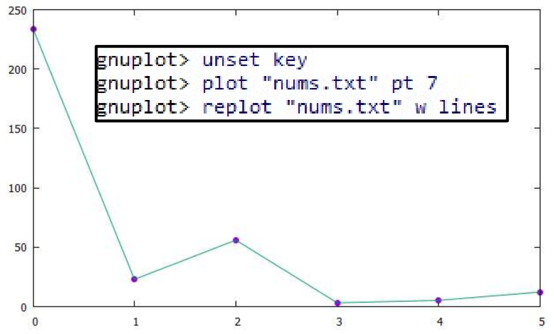
use **replot** to plot more than one thing, or after you have changed a setting.



```
gnuplot> plot "nums.txt" pt
```



```
gnuplot> plot "nums.txt" pt 7
```



```
gnuplot> unset key
gnuplot> plot "nums.txt" pt 7
gnuplot> replot "nums.txt" w lines
```

gnuplot> test terminal

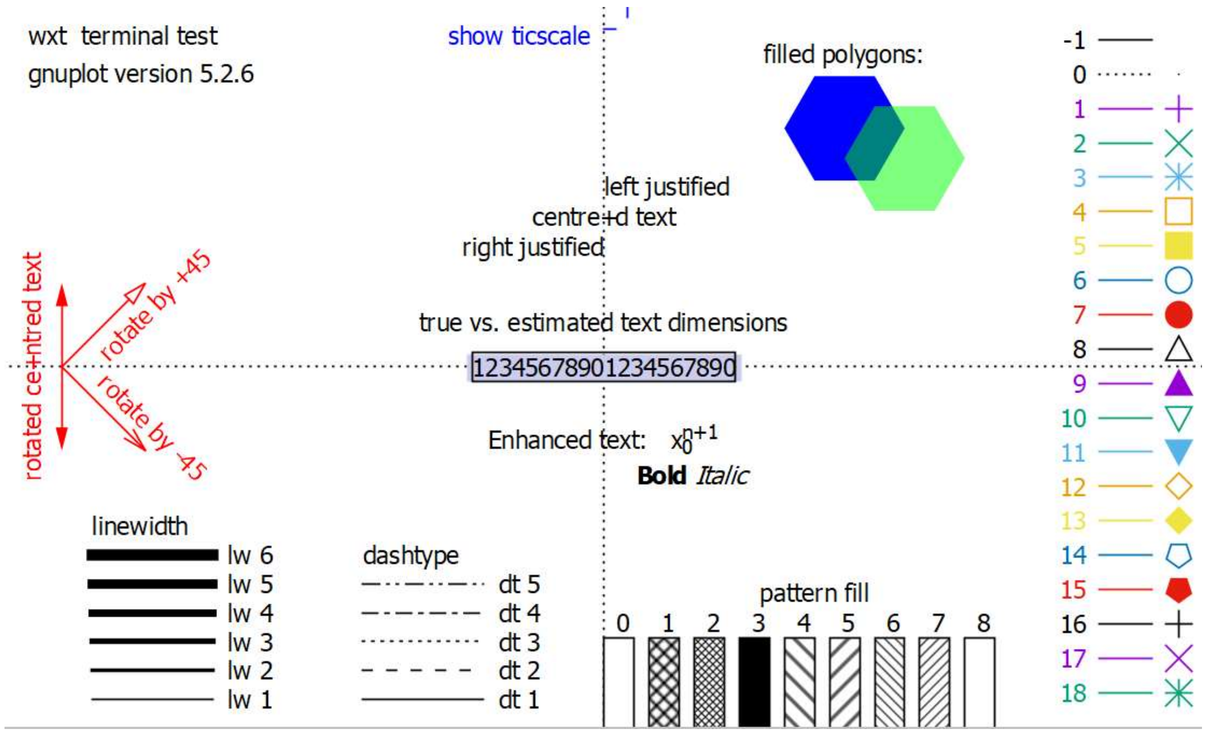
different terminal types have formatted output.

set term pdf

replot will display pdf code to screen.

To **output to file:**

```
set term pdf
set output “out.pdf”
replot
```



wxt terminal test
gnuplot version 5.2.6
show ticscale
filled polygons:
left justified
centre+d text
right justified
rotated ce+ntred text
rotate by +45
rotate by -45
true vs. estimated text dimensions
12345678901234567890
Enhanced text: x_0^{n+1}
Bold *Italic*
linewidth
lw 6
lw 5
lw 4
lw 3
lw 2
lw 1
dashtype
dt 5
dt 4
dt 3
dt 2
dt 1
pattern fill
0 1 2 3 4 5 6 7 8
-1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18