The Unix Shell

Introduction to the shell

Files and the file system

Creating and deleting files

Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See http://software-carpentry.org/license.html for more information.

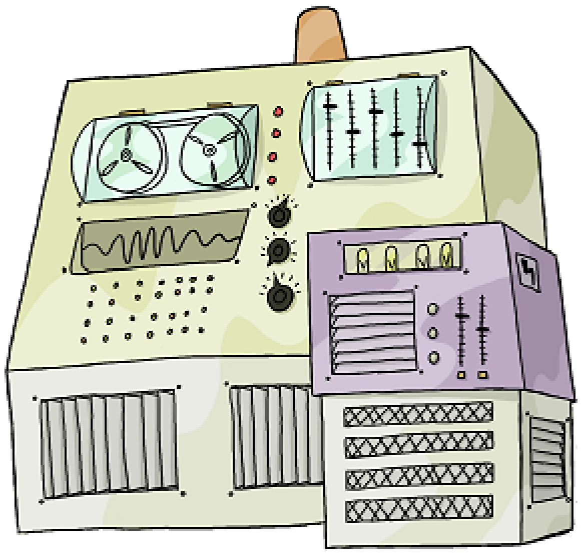
The Unix Shell

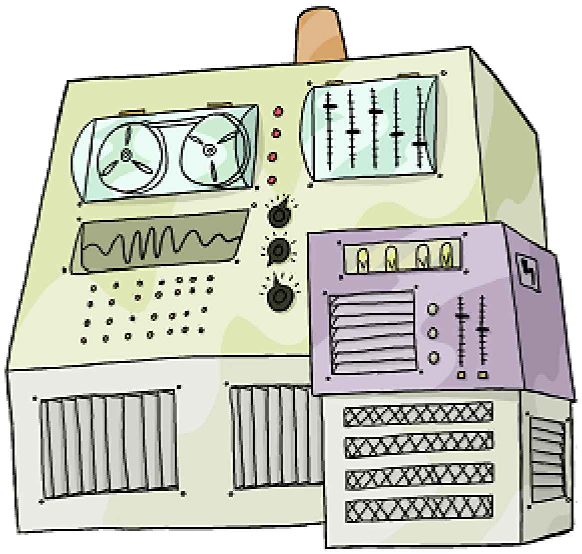
Introduction

Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

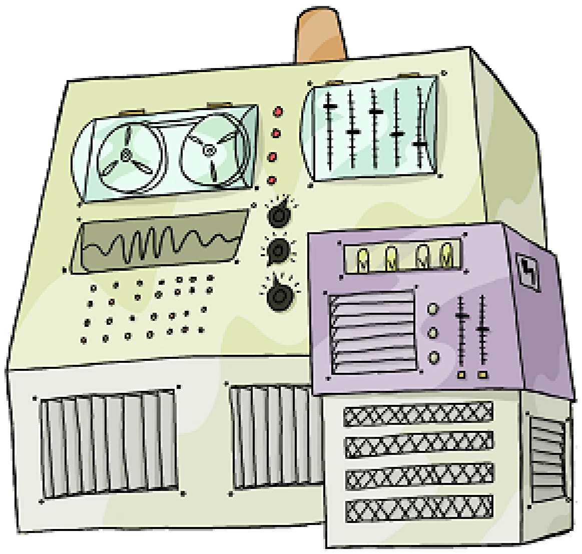
See http://software-carpentry.org/license.html for more information.





Run

Programs

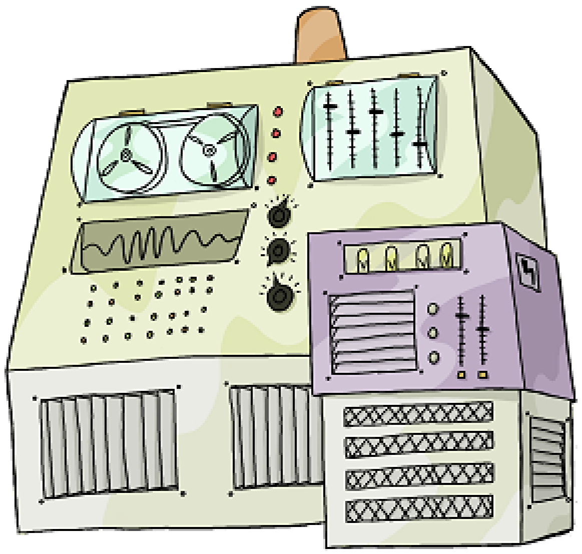


Run

Programs

Store

Data



Run

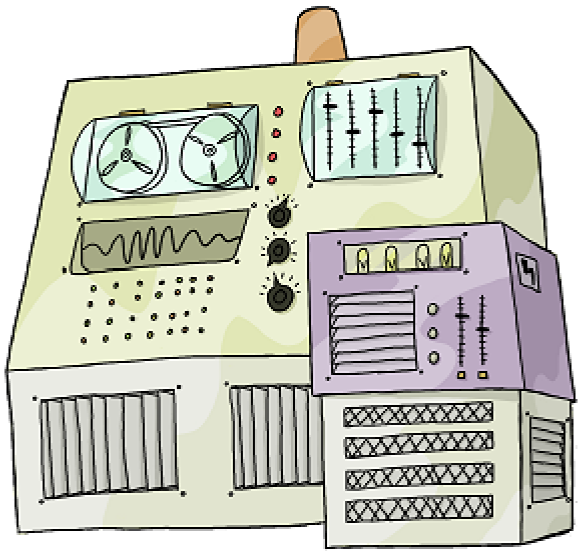
Programs

Store

Data

Communicate

with each other



Run

Programs

Store

Data

Communicate

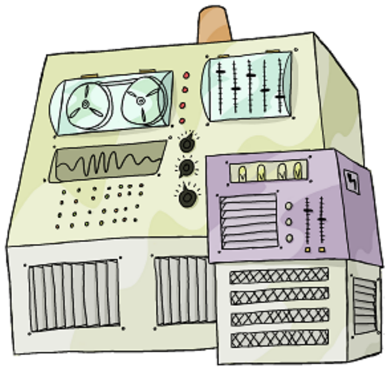
with each other

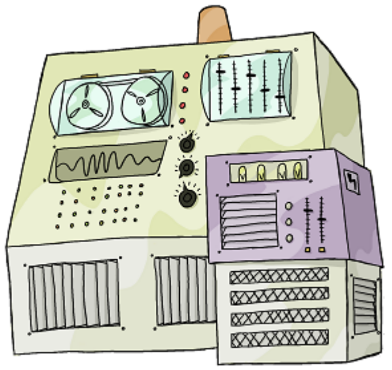
Interact

with us

Interact

with us

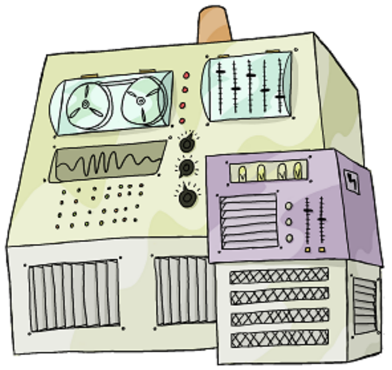




Interact

with us

Telepathy



Interact

with us

Telepathy

Speech

Interact

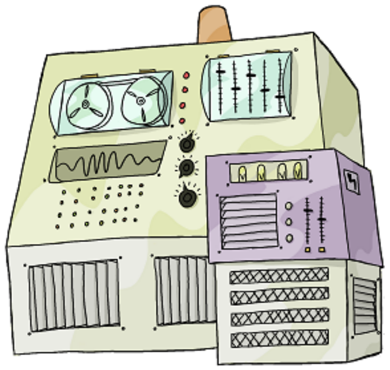
with us

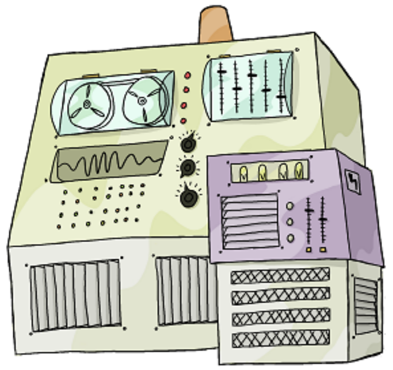
Telepathy

Speech

WIMP

(windows, icons, mice, pointers)





Interact

with us

Rewiring

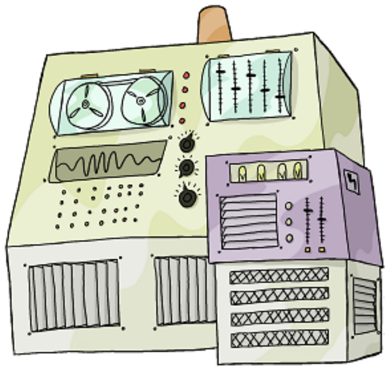
Telepathy

Speech

WIMP

Interact with us

$$\begin{tikzcd}
 & & \text{Interact with us} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{Rewiring} & \text{Typewriter} & \text{WIMP} & \text{Speech} & \text{Telepathy}
\end{tikzcd}$$



Typewriter

~~Typewriter~~

Line printer + keyboard

~~Typewriter~~

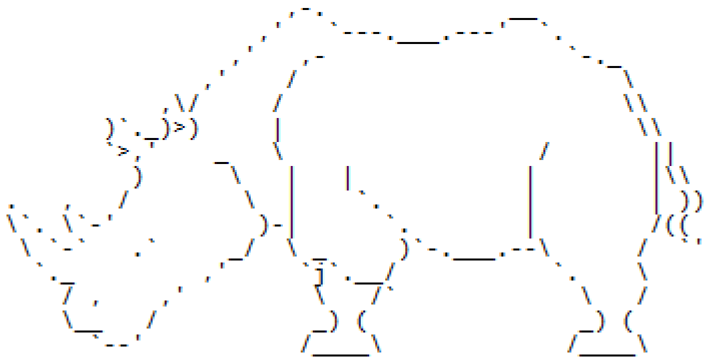
Line printer + keyboard

Text only

~~Typewriter~~

Line printer + keyboard

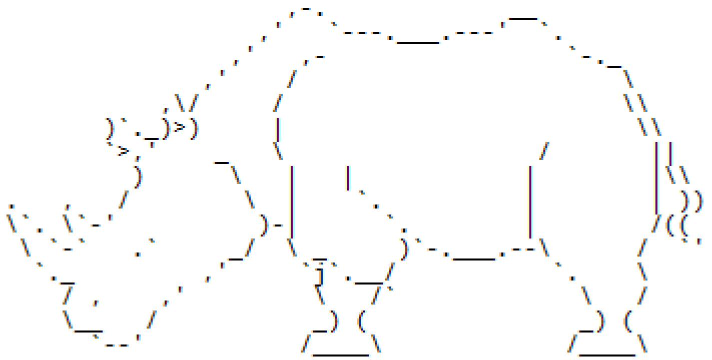
Text only



~~Typewriter~~

Line printer + keyboard

Text only



CLUI: command-line user interface

```
user logs in
```

```
user logs in
user types command
```

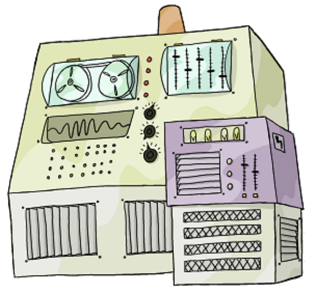
```
user logs in
user types command
computer executes command
  and prints output
```

```
user logs in
user types command
computer executes command
  and prints output
user types another command
```

```
user logs in
user types command
computer executes command
  and prints output
user types another command
computer executes command
  and prints output
```

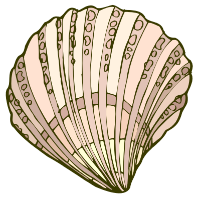
```
user logs in
user types command
computer executes command
  and prints output
user types another command
computer executes command
  and prints output
⋮
user logs off
```

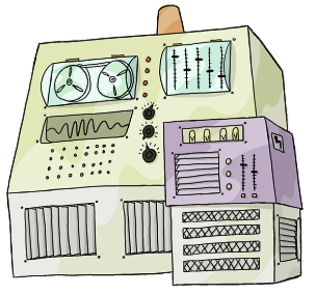
```
user logs in
user types command
computer executes command
  and prints output
user types another command
computer executes command
  and prints output
⋮
user logs off
```



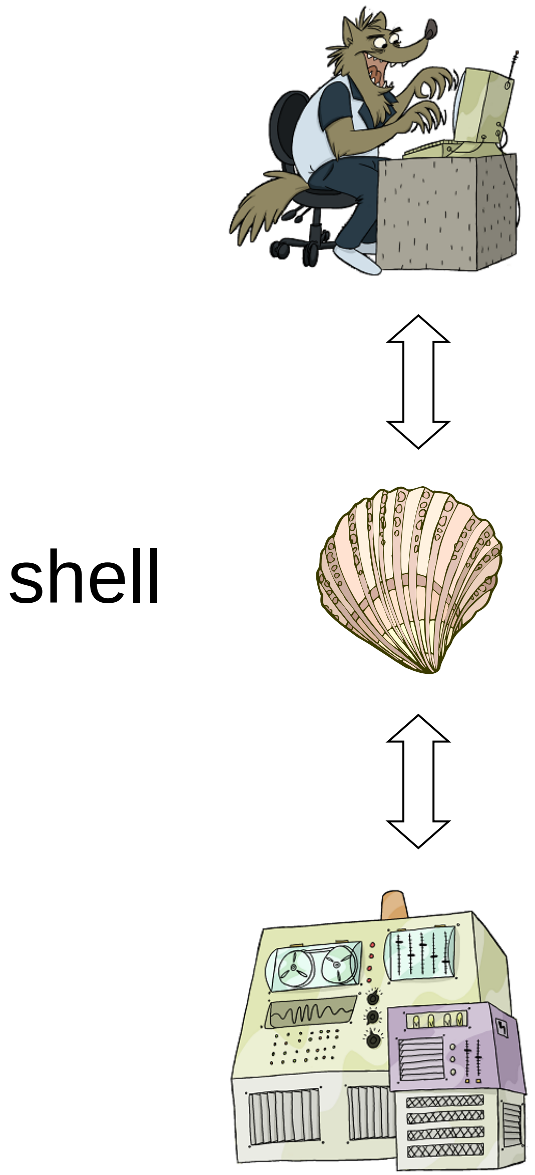
```
user logs in
user types command
computer executes command
  and prints output
user types another command
computer executes command
  and prints output
⋮
user logs off
```

shell





```
user logs in
user types command
computer executes command
  and prints output
user types another command
computer executes command
  and prints output
⋮
user logs off
```



A shell is just a program that runs other programs

A shell is just a program that runs other programs

Most popular is bash (the <u>B</u>ourne <u>a</u>gain <u>sh</u>ell)

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

Using it feels a lot more like programming

than using windows, a mouse, etc.

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

Using it feels a lot more like programming

than using windows, a mouse, etc.

Commands are terse and often cryptic

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

Using it feels a lot more like programming

than using windows, a mouse, etc.

Commands are terse and often cryptic

Use it because:

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

Using it feels a lot more like programming

than using windows, a mouse, etc.

Commands are terse and often cryptic

Use it because:

– many tools only have command-line interfaces

A shell is just a program that runs other programs

Most popular is bash (the Bourne again shell)

Using it feels a lot more like programming

than using windows, a mouse, etc.

Commands are terse and often cryptic

Use it because:

– many tools only have command-line interfaces

– allows you to combine tools in powerful new ways

created by

Greg Wilson

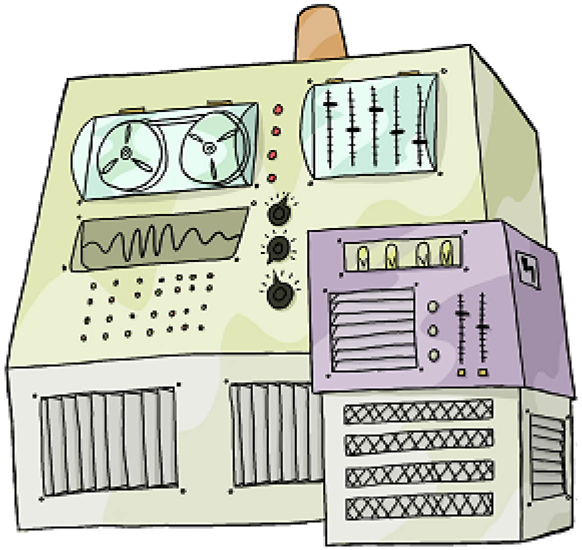
August 2010

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.

The Unix Shell

Files and Directories

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.



Run

Programs

Store

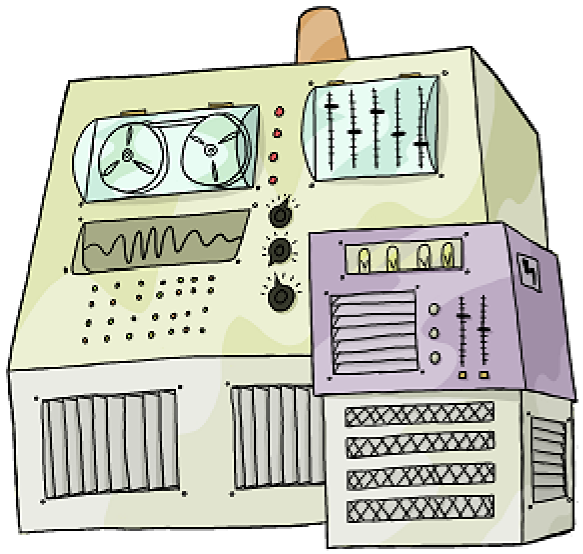
Data

Communicate

with each other

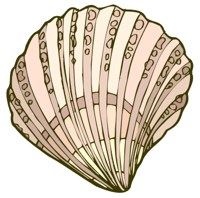
Interact

with us



Run

Programs



shell

Store

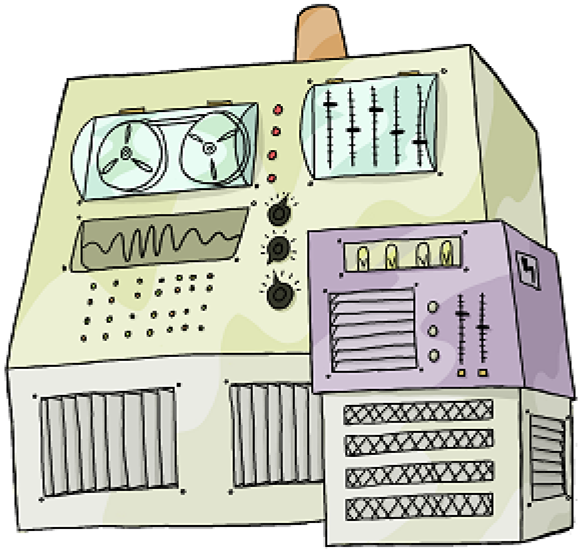
Data

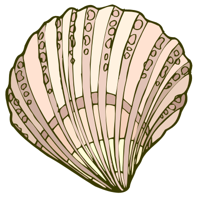
Communicate

with each other

Interact

with us

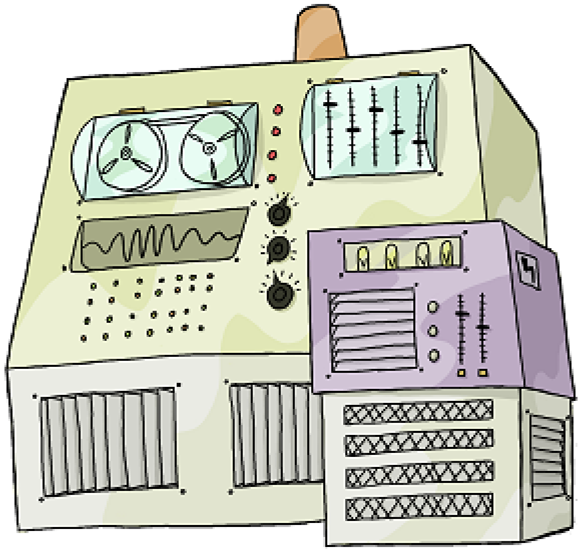


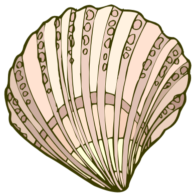


shell

Store

Data





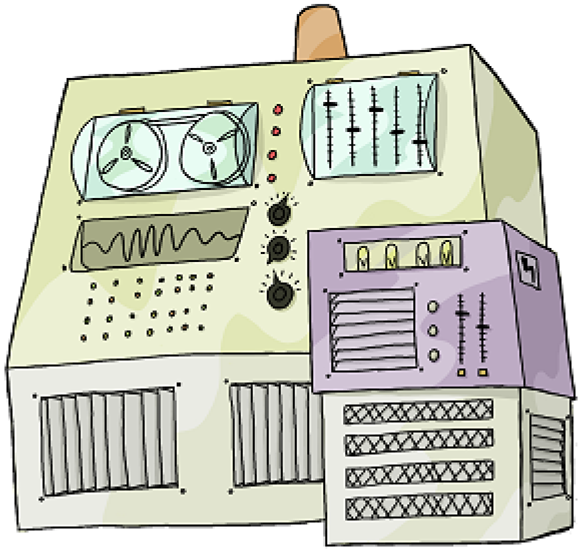
shell

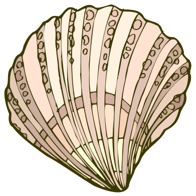
Store

Data

$\swarrow$

file system



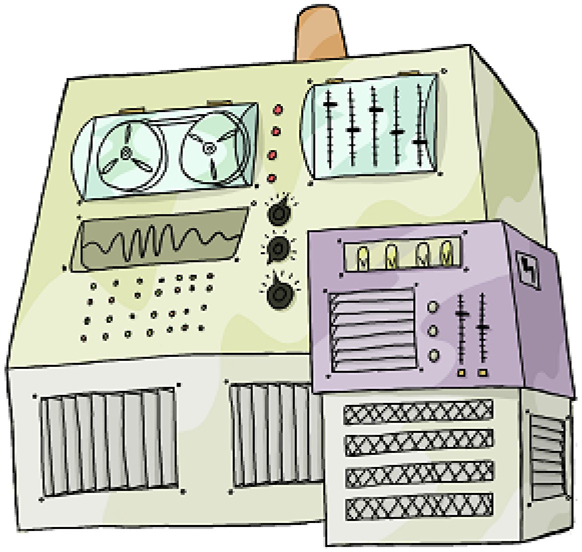


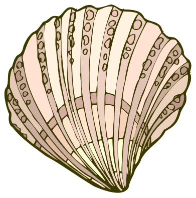
shell

Store

Data

$$\begin{tikzcd}
\text{Data} \arrow[d] \\
\text{file system} \arrow[d] \\
\text{files}
\end{tikzcd}$$



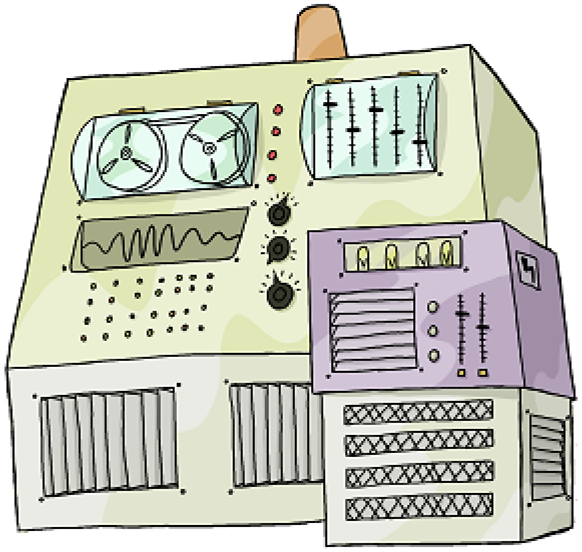


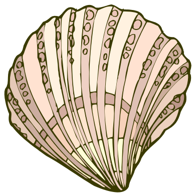
shell

Store

Data

$$\begin{tikzcd}
 & \text{Store Data} \arrow[d] & \\
 & \text{file system} \arrow[dl] \arrow[dr] & \\
\text{files} & & \text{directories}
\end{tikzcd}$$





shell

Store

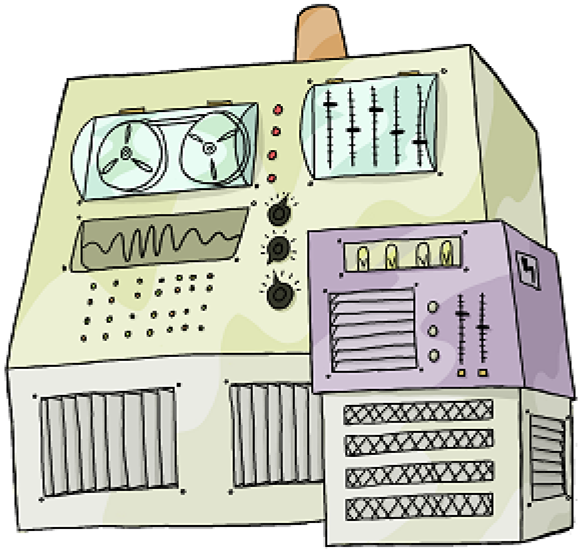
Data

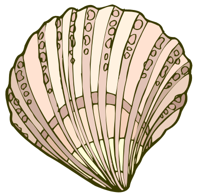
$$\begin{tikzcd}
& \text{file system} \arrow[dl] \arrow[dr] & \\
\text{files} & & \text{directories}
\end{tikzcd}$$

Use the shell

to view and change

the file system





shell

Use the shell

to run commands

to view what's in

the file system

$$\begin{tikzcd}
\text{Store Data} \arrow[d] \\
\text{file system} \arrow[d] \arrow[rd] \\
\text{files} & \text{directories}
\end{tikzcd}$$

```
login:
```

```
login:
```

$\xleftarrow{\hspace{3cm}}$ computer prompt in **bold**

```
login:
```

computer prompt in **bold**

explanatory text in blue

```
login: vlad
```

$\longleftarrow$ user input in green

```
login: vlad
password: *********
```

$\xleftarrow{\hspace{3em}}$ password

```
login: vlad
password: *********
$
```

$\xleftarrow{\hspace{4cm}}$ shell prompt

```
login: vlad
password: *********
$
```

$\leftarrow$ shell prompt

like Python's >>> and …

```
login: vlad
password: *********
$ whoami
```

check user ID

```
login: vlad
password: *********
$ whoami
```

$\leftarrow$ check user ID

shell finds the `whoami` program

```
login: vlad
password: *********
$ whoami
```

check user ID

shell finds the `whoami` program

runs it

```
login: vlad
password: *********
$ whoami
vlad
```

check user ID

shell finds the `whoami` program

runs it

prints its output

```
login: vlad
password: *********
$ whoami
vlad
$
```

check user ID

shell finds the `whoami` program

runs it

prints its output

displays a new prompt

```
login: vlad
password: *********
$ whoami
vlad
$ pwd
```

$\longleftarrow$ what is the *working directory*

```
login: vlad
password: *********
$ whoami
vlad
$ pwd
```

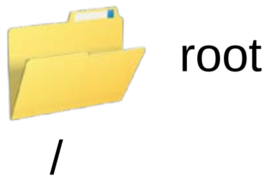
what is the *working directory*

the directory used when no other directory is explicitly specified

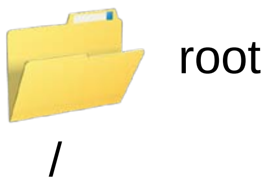
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```

```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```

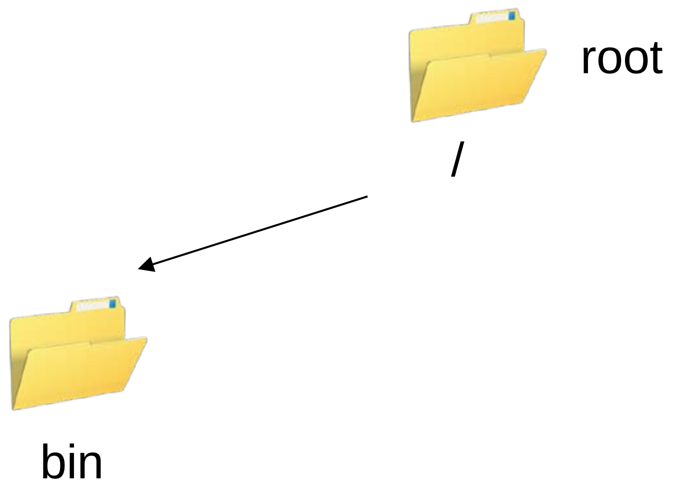
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



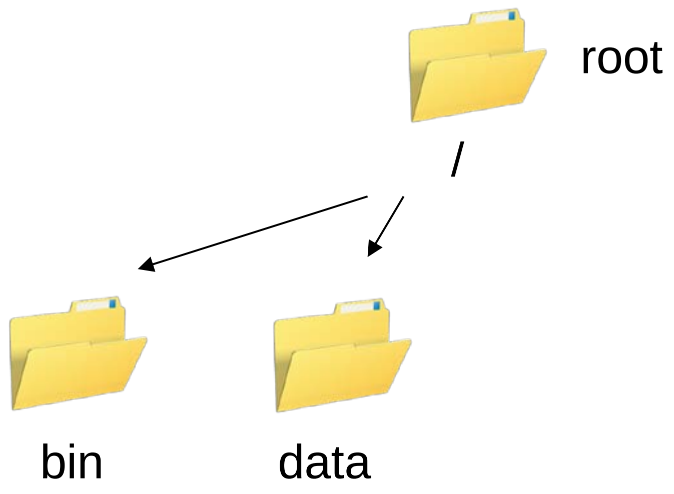
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



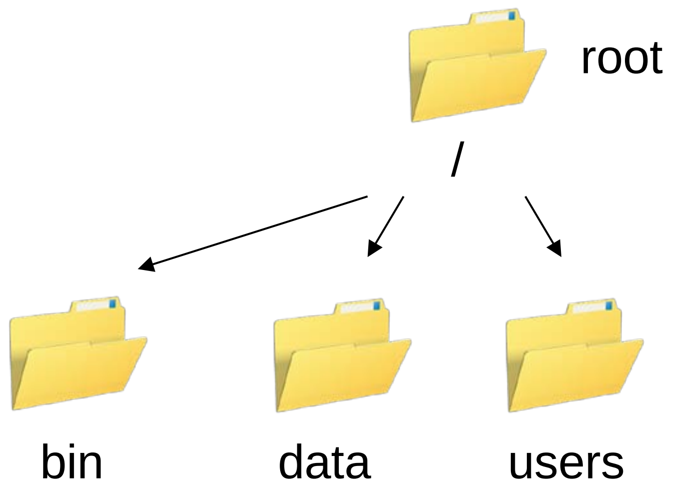
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



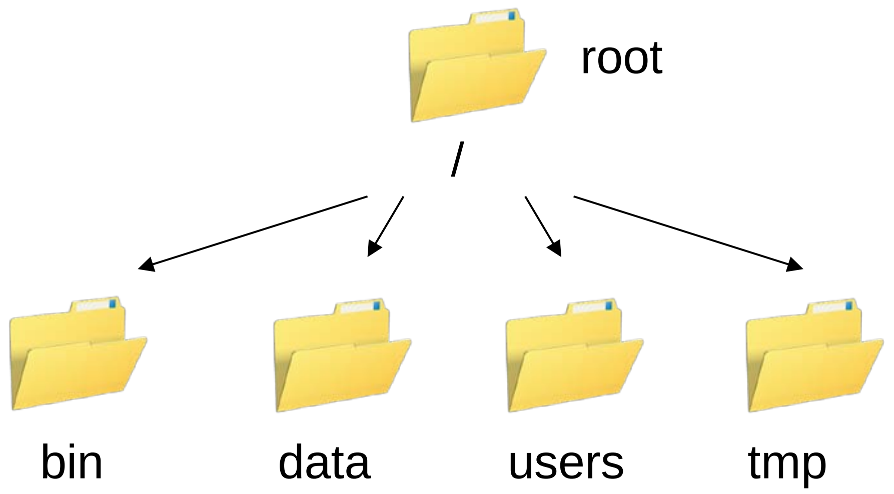
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



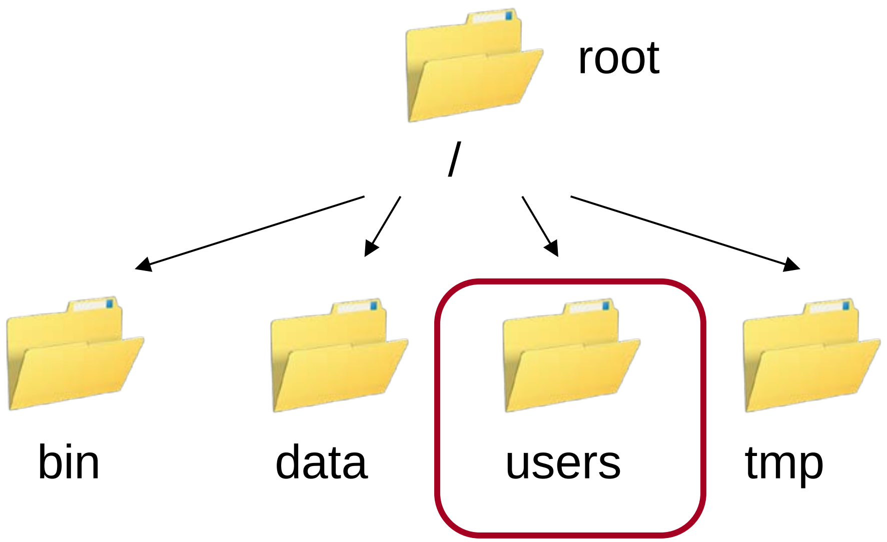
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



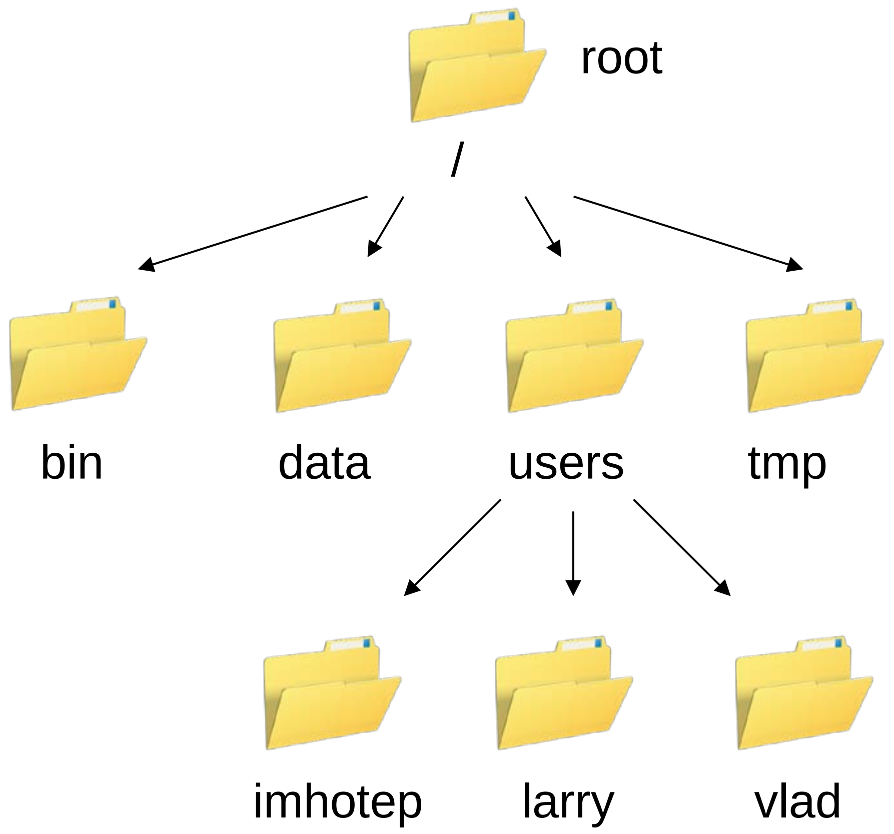
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



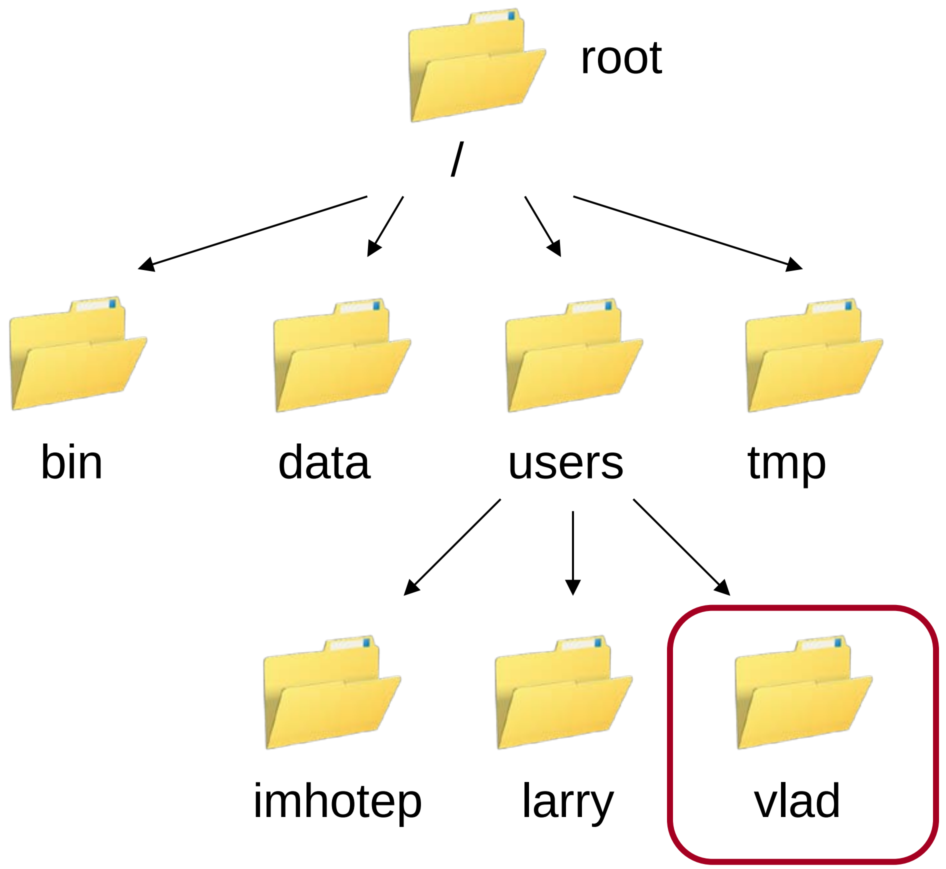
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



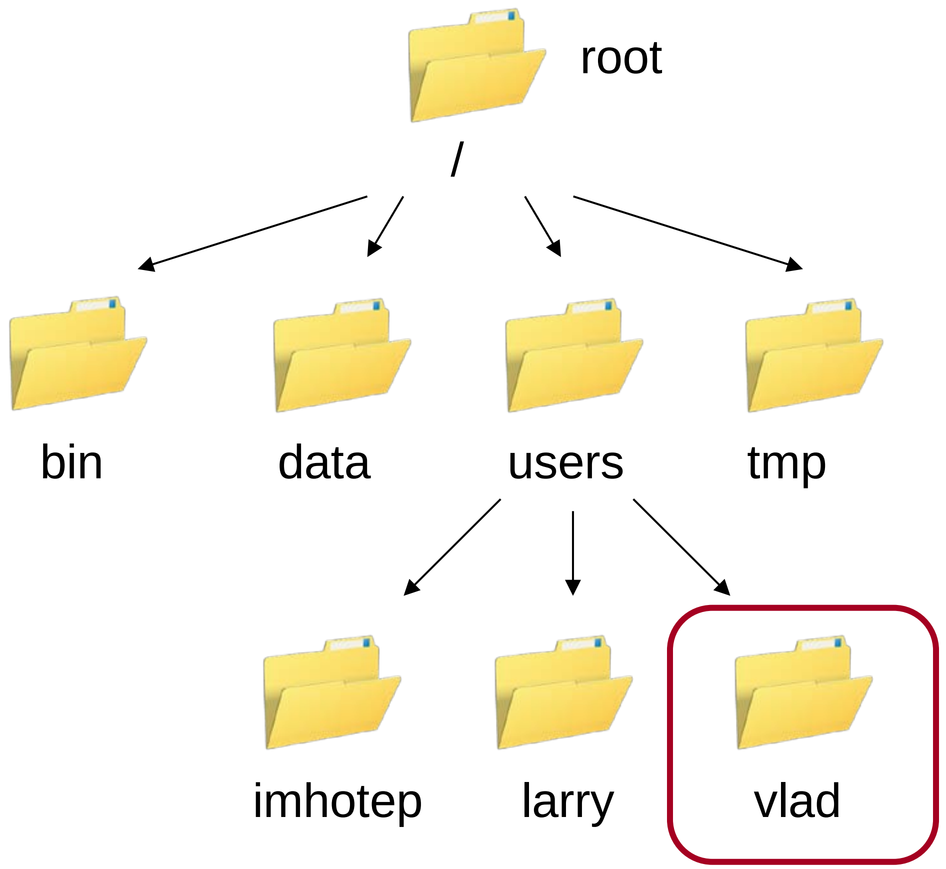
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



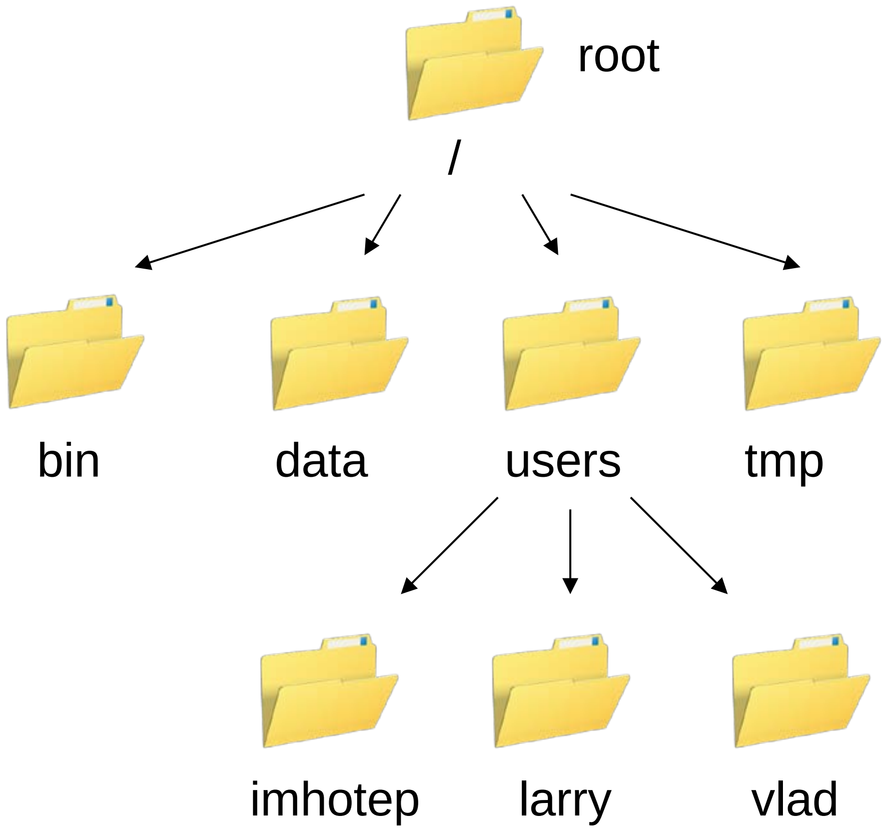
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$
```



```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$ ls
```

$\longleftarrow$ stands for "listing"

```
login: vlad
password: ********
$ whoami
vlad
$ pwd
/users/vlad
$ ls
```

stands for "listing"
sadly more memorable than
most command names

```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$ ls
bin          data      mail       music
notes.txt    papers    pizza.cfg  solar
solar.pdf    swc
$
```

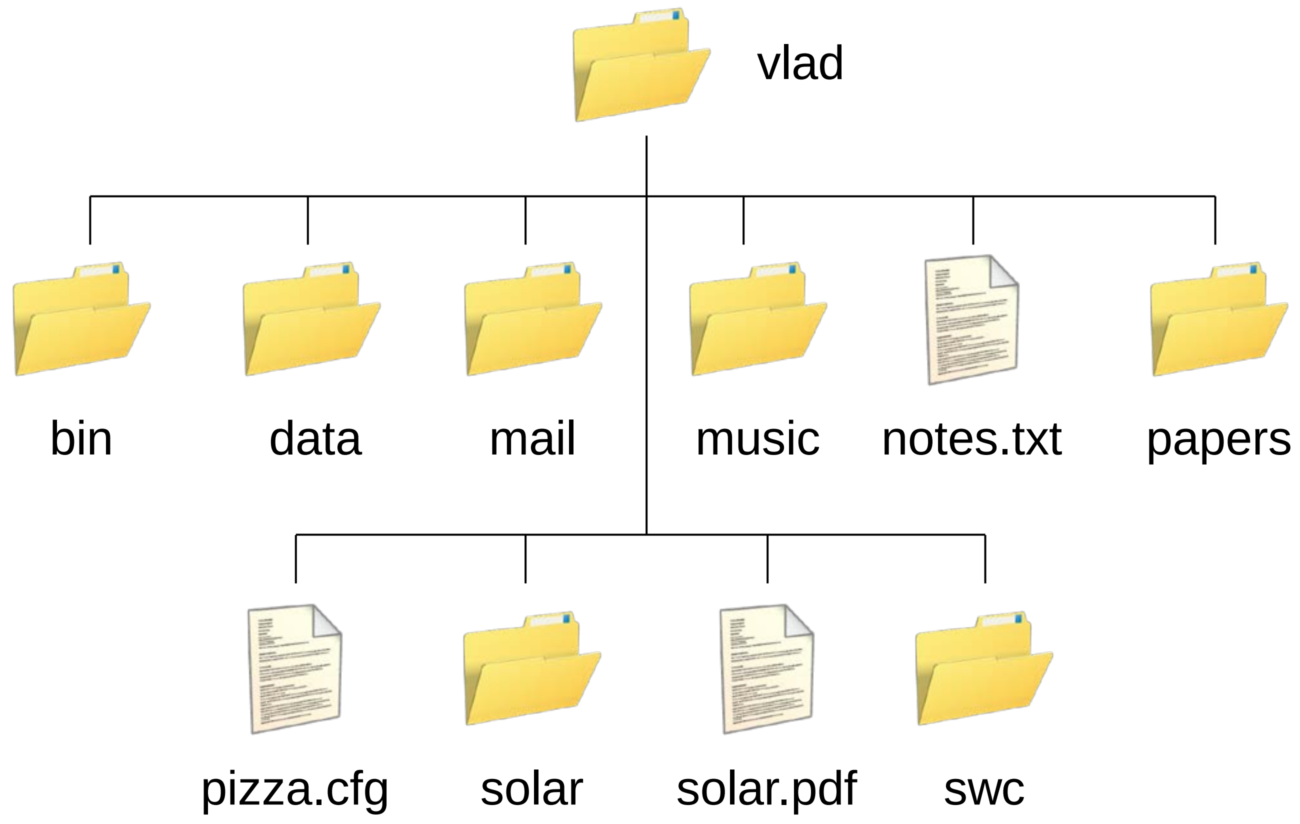
```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$
```

an *argument* or *flag* modifying the command's behavior

```
login: vlad
password: *********
$ whoami
vlad
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$
```

adds a trailing '/' to directory names

```
$ ls -F
bin/           data/     mail/       music/
notes.txt      papers/   pizza.cfg   solar/
solar.pdf      swc/
```



```
$ ls -F
bin/           data/     mail/      music/
notes.txt      papers/   pizza.cfg  solar/
solar.pdf      swc/
```

By convention, use *filename extension* to indicate file type

```
$ ls -F
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
```

By convention, use *filename extension* to indicate file type

.txt for text, .pdf for PDF, .cfg for configuration file, etc.

```
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
```

By convention, use *filename extension* to indicate file type

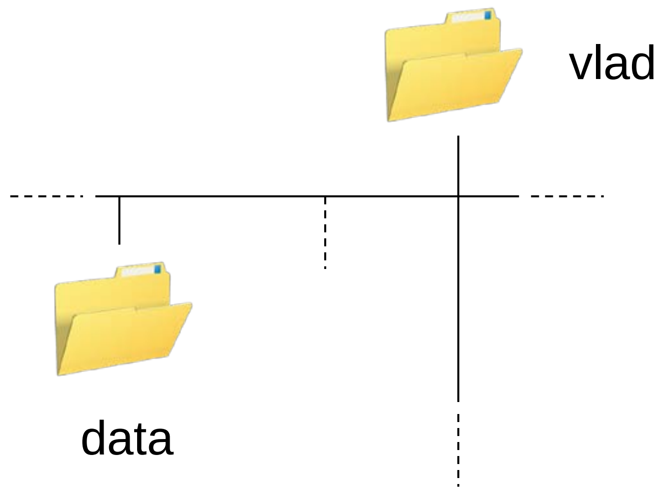
.txt for text, .pdf for PDF, .cfg for configuration file, etc.

But this is only a convention, not a guarantee

```
$ ls -F data
$ ls -F data
amino_acids.txt   elements/     morse.txt
pdb/              planets.txt   sunspot.txt
$
```

```
$ ls -F data
amino_acids.txt   elements/     morse.txt
pdb/              planets.txt   sunspot.txt
$
```

a *relative path*

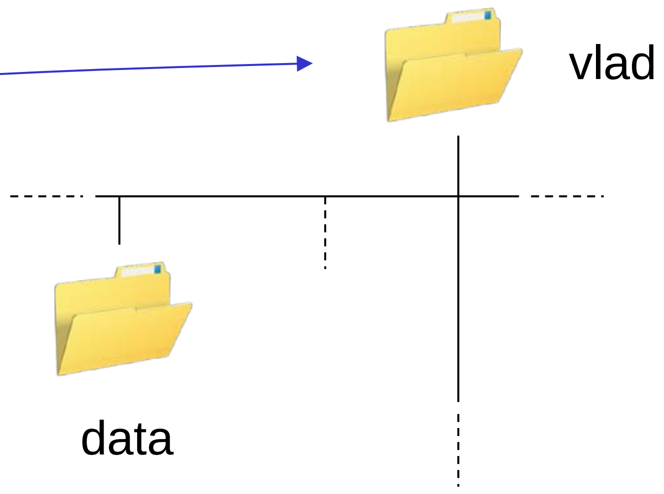


```
$ ls -F data
amino_acids.txt   elements/     morse.txt
pdb/              planets.txt   sunspot.txt
$
```

a *relative path*

relative to

current working directory



```
$ ls -F /data
access.log    backup/    hardware.cfg
network.cfg
$
```

```
$ ls -F /data
access.log    backup/    hardware.cfg
network.cfg
$
```

an *absolute path*

```
$ ls -F /data
access.log    backup/    hardware.cfg
network.cfg
$
```

an *absolute path*

leading '/' means "from root"

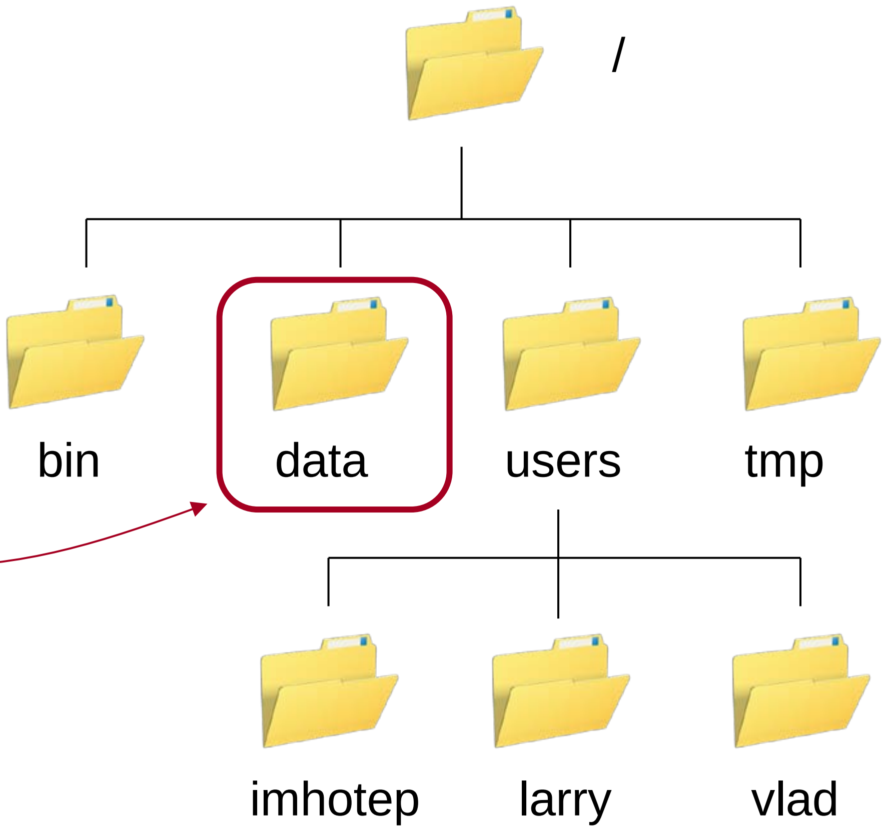
```
$ ls -F /data
access.log    backup/    hardware.cfg
network.cfg
$
```

an *absolute path*

leading '/' means "from root"

so it always refers to

this directory



```
$ pwd
/users/vlad
$
```

```
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$
```

```
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ cd data
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ cd data
```

$\leftarrow$ change directory

```
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ cd data
```

change directory

actually doesn't change the directory

```
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ cd data
```

change directory

actually doesn't change the directory

changes the shell's idea of

which directory we are in

```
$ pwd
/users/vlad
$ ls
bin/          data/      mail/       music/
notes.txt     papers/    pizza.cfg   solar/
solar.pdf     swc/
$ cd data
$ pwd
/users/vlad/data
$
```

```
$ pwd
/users/vlad
$ ls
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ cd data
$ pwd
/users/vlad/data
$ ls
amino_acids.txt   elements/     morse.txt
pdb/              planets.txt   sunspot.txt
$
```

```
$ pwd
/users/vlad
$ ls
bin/          data/       mail/       music/
notes.txt     papers/     pizza.cfg   solar/
solar.pdf     swc/
$ cd data
$ pwd
/users/vlad/data
$ ls
amino_acids.txt   elements/      morse.txt
pdb/              planets.txt    sunspot.txt
$
```

because we're now "in" this directory

```
$ pwd
/users/vlad/data
$
```

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad/data
$ cd ..
```

the directory above the current one

```
$ pwd
/users/vlad/data
$ cd ..
```

$\xleftarrow{}$ the directory above the current one

its *parent directory*

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$
```

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/         data/     mail/      music/
notes.txt    papers/   pizza.cfg  solar/
solar.pdf    swc/
$
```

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ ls -F -a
./            ../       bin/       data/
mail/         music/    notes.txt  papers/
pizza.cfg     solar/    solar.pdf    swc/
```

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ ls -F -a
./            ../       bin/       data/
mail/         music/    notes.txt  papers/
pizza.cfg     solar/    solar.pdf    swc/
```

"show all"

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/           data/      mail/      music/
notes.txt      papers/    pizza.cfg  solar/
solar.pdf      swc/
$ ls -F -a
./             ../        bin/       data/
mail/          music/     notes.txt  papers/
pizza.cfg      solar/     solar.pdf    swc/
```

parent directory

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ ls -F -a
./            ../       bin/        data/
mail/         music/    notes.txt   papers/
pizza.cfg     solar/    solar.pdf     swc/
```

parent directory

/users

```
$ pwd
/users/vlad/data
$ cd ..
$ pwd
/users/vlad
$ ls
bin/          data/      mail/      music/
notes.txt     papers/    pizza.cfg  solar/
solar.pdf     swc/
$ ls -F -a
./            ../        bin/       data/
mail/         music/     notes.txt  papers/
pizza.cfg     solar/     solar.pdf  swc/
```

this directory itself

Things are different on Windows

Things are different on Windows

```
C:\Users\vlad
```

Things are different on Windows

$$\boxed{\texttt{C:}}\texttt{\textbackslash Users\textbackslash vlad}$$

Drive letter

Things are different on Windows

```
C:\Users\vlad
```

Drive letter

Each drive is a separate file system

Things are different on Windows

```
C:\Users\vlad
```

Backslash \ as separator

Things are different on Windows

```
C:\Users\vlad
```

Backslash \ as separator

Unix uses \ to escape special characters

in names like `my\ files.txt`

Things are different on Windows

$$\underbrace{\texttt{C:\textbackslash Users\textbackslash}}_{\text{Case insensitive}}\texttt{vlad}$$

Things are different on Windows

$$\begin{tikzcd} & \underbrace{\texttt{C:\textbackslash Users\textbackslash vlad}}_{\text{Case insensitive}} \arrow[dl] \arrow[d] \arrow[dr] & \\ \texttt{c:\textbackslash users\textbackslash vlad} & \texttt{C:\textbackslash USERS\textbackslash VLAD} & \texttt{C:\textbackslash uSeRs\textbackslash Vl} \end{tikzcd}$$

Things are different on Windows

`C:\Users\vlad`

Cygwin: `/cygdrive/c/Users/vlad`

Map drive letters to "directories"

Things are different on Windows

`C:\Users\vlad`

Cygwin: `/cygdrive/c/Users/vlad`

Map drive letters to "directories"

And use / instead of \

Things are different on Windows

`C:\Users\vlad`

Cygwin: `/cygdrive/c/Users/vlad`

Map drive letters to "directories"

And use / instead of \

But still case insensitive

Things are different on Windows

`C:\Users\vlad`

Cygwin: $\underbrace{\texttt{/cygdrive/c}}\texttt{/Users/vlad}$

Map drive letters to "directories"

And use / instead of \

But still case insensitive

Can't put `backup.txt` and `Backup.txt` in a directory

pwd	print working directory
cd	change working directory
ls	listing
.	current directory
..	parent directory

created by

Greg Wilson

August 2010

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.

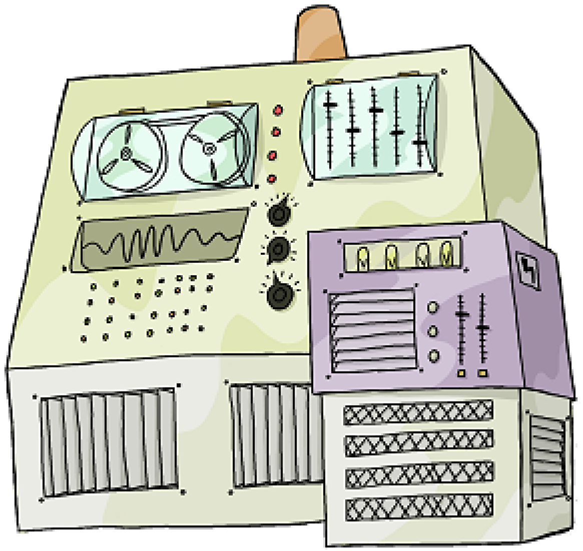
The Unix Shell

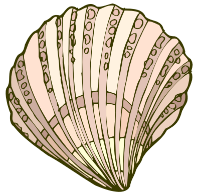
Creating and Deleting

Copyright © Software Carpentry 2010

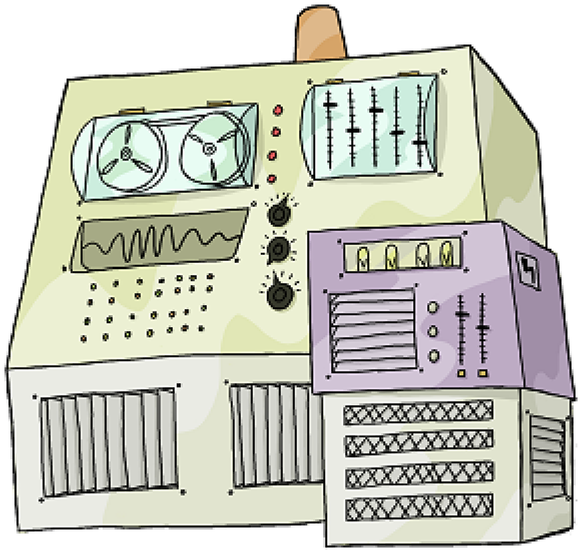
This work is licensed under the Creative Commons Attribution License

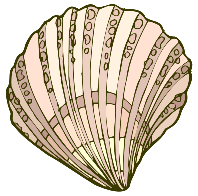
See http://software-carpentry.org/license.html for more information.





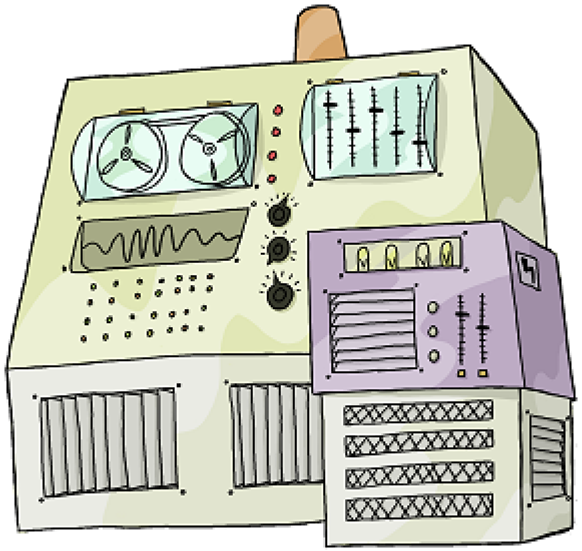
shell

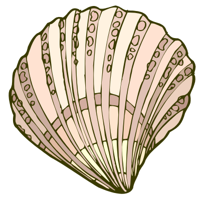




shell

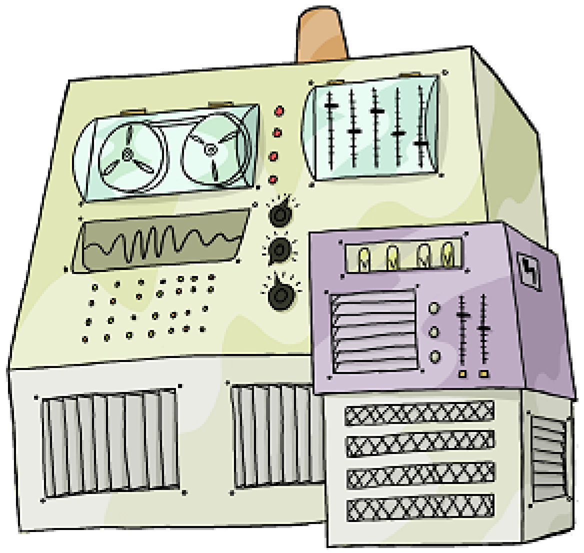
pwd	print working directory
cd	change working directory
ls	listing
.	current directory
..	parent directory





shell

pwd	print working directory
cd	change working directory
ls	listing
.	current directory
.	parent directory



But how do we create things

in the first place?

```
$ pwd
/users/vlad
$
```

```
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$
```

```
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/      music/
notes.txt     papers/   pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ mkdir tmp
```

make directory

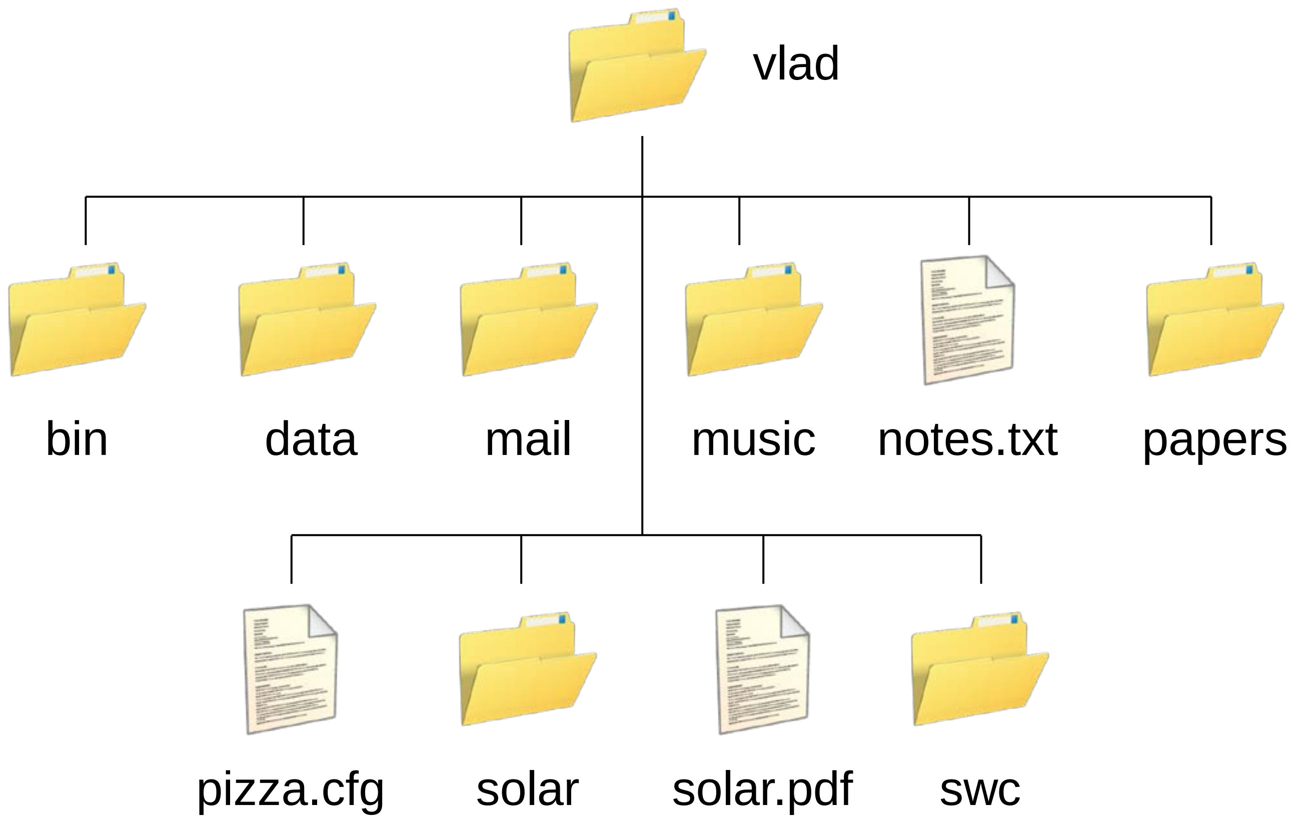
```
$ pwd
/users/vlad
$ ls -F
bin/          data/     mail/       music/
notes.txt     papers/   pizza.cfg   solar/
solar.pdf     swc/
$ mkdir tmp
```

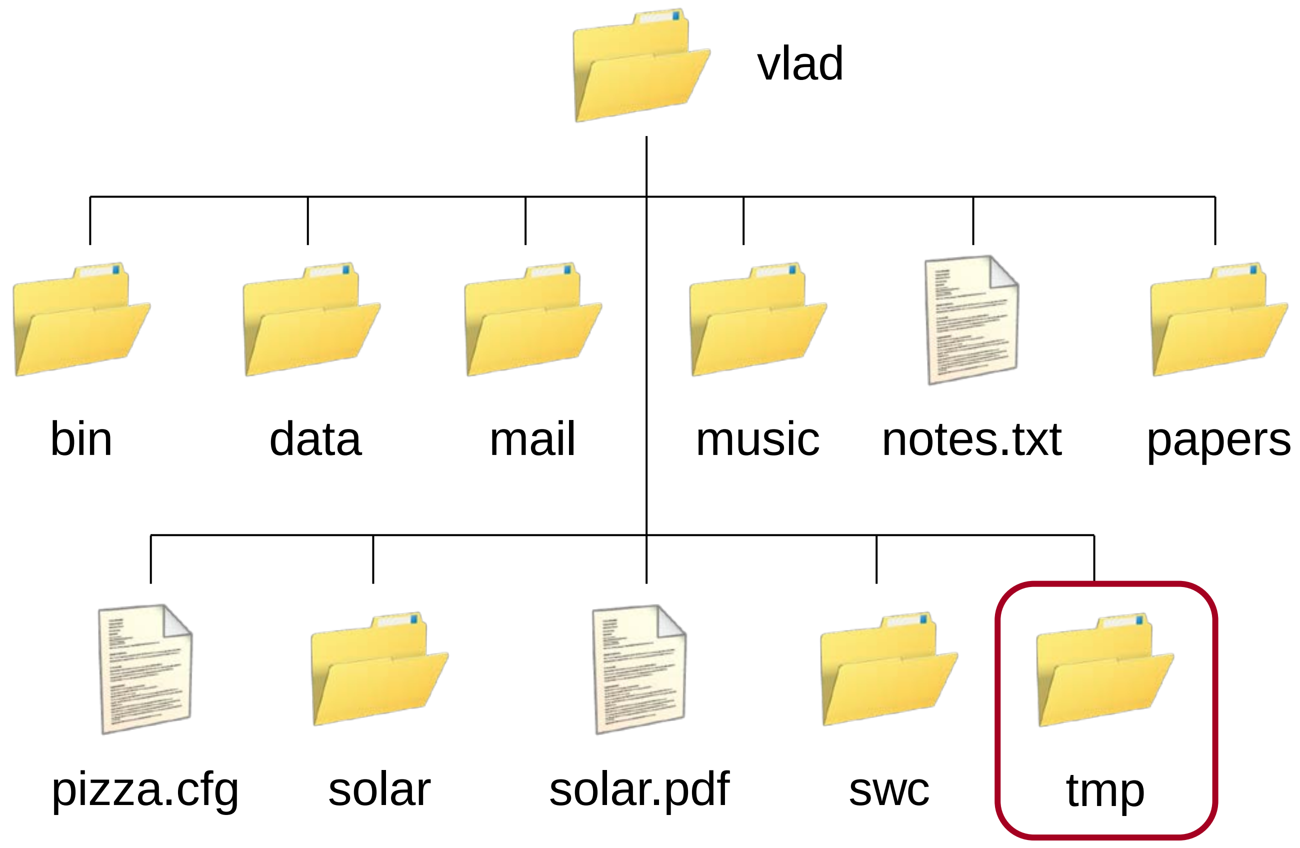
make directory

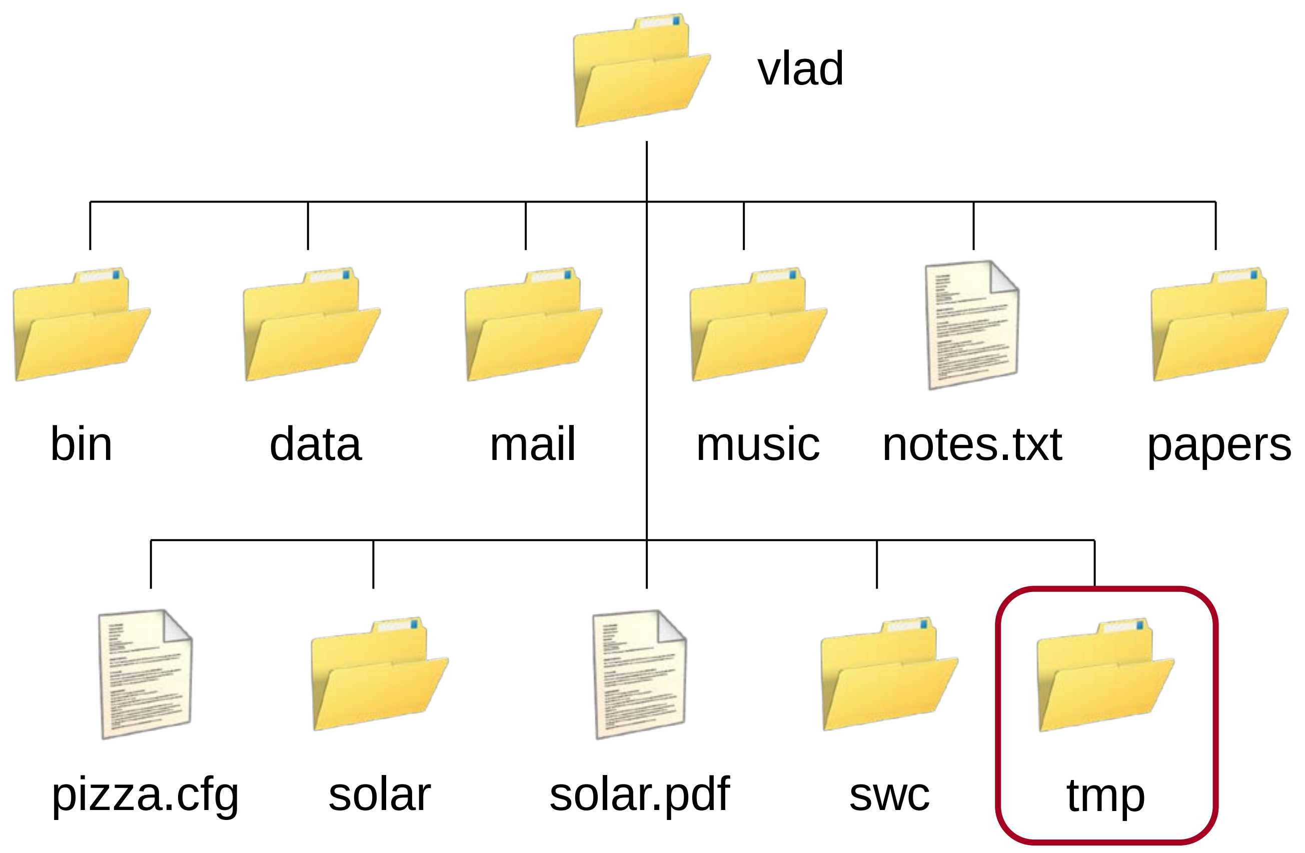
a relative path, so the new directory is made below the current one

```
$ pwd
/users/vlad
$ ls -F
bin/         data/     mail/      music/
notes.txt    papers/   pizza.cfg  solar/
solar.pdf    swc/
$ mkdir tmp
$ ls –F
bin/         data/     mail/      music/
notes.txt    papers/   pizza.cfg  solar/
solar.pdf    swc/      tmp/
$
```

```
$ pwd
/users/vlad
$ ls -F
bin/          data/      mail/      music/
notes.txt     papers/    pizza.cfg  solar/
solar.pdf     swc/
$ mkdir tmp
$ ls –F
bin/          data/      mail/      music/
notes.txt     papers/    pizza.cfg  solar/
solar.pdf     swc/       tmp/
$
```


vlad
bin
data
mail
music
notes.txt
papers
pizza.cfg
solar
solar.pdf
swc


vlad
bin
data
mail
music
notes.txt
papers
pizza.cfg
solar
solar.pdf
swc
tmp



nothing below it yet

```
$ pwd
/users/vlad
$
```

```
$ pwd
/users/vlad
$ ls tmp
$
```

```
$ pwd
/users/vlad
$ ls tmp
$
```

$\xleftarrow{\hspace{4em}}$ no output

```
$ pwd
/users/vlad
$ ls tmp
$ ls -a tmp
.             ..
$
```

```
$ pwd
/users/vlad
$ ls tmp
$ ls -a tmp
.             ..
```

$\uparrow$

/users/vlad/tmp

```
$ pwd
/users/vlad
$ ls tmp
$ ls -a tmp
.             ..
```

/users/vlad

```
$ cd tmp
$ nano junk
$ cd tmp
$ nano junk
```

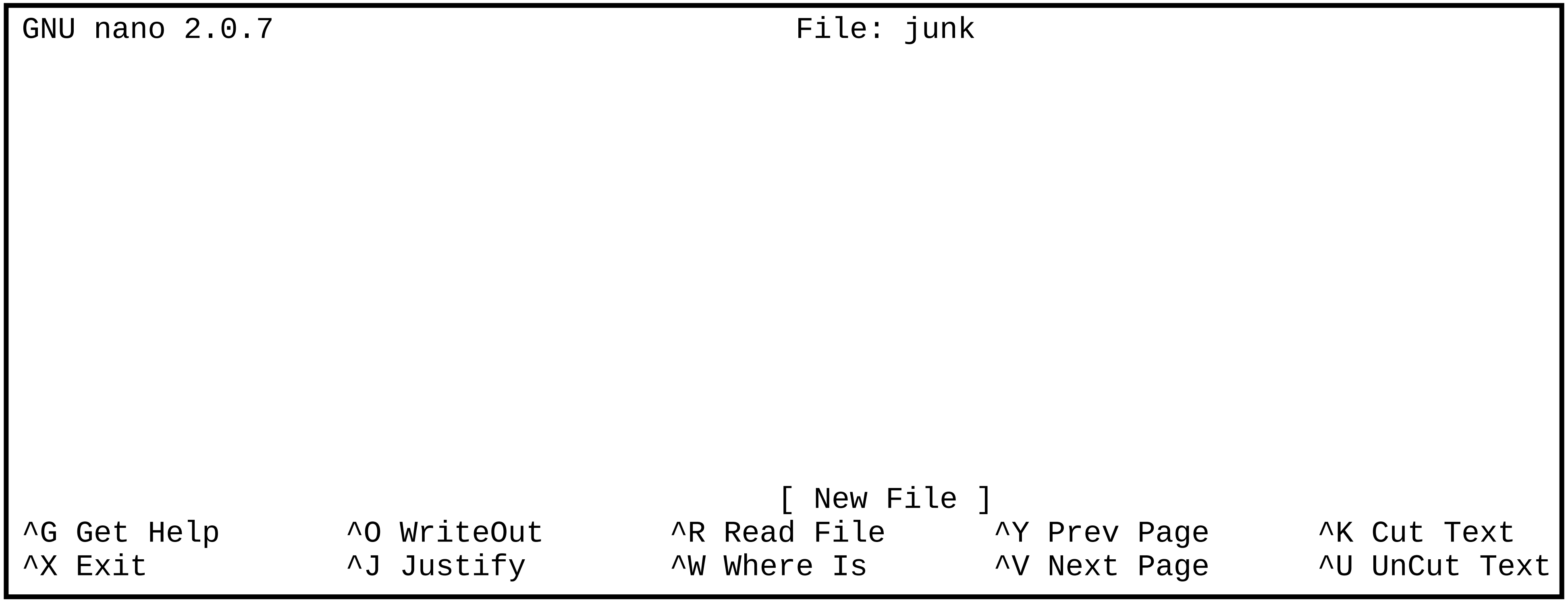
a text editor only a programmer could love

```
$ cd tmp
$ nano junk
```

a text editor only a programmer could love

really do mean "text"…

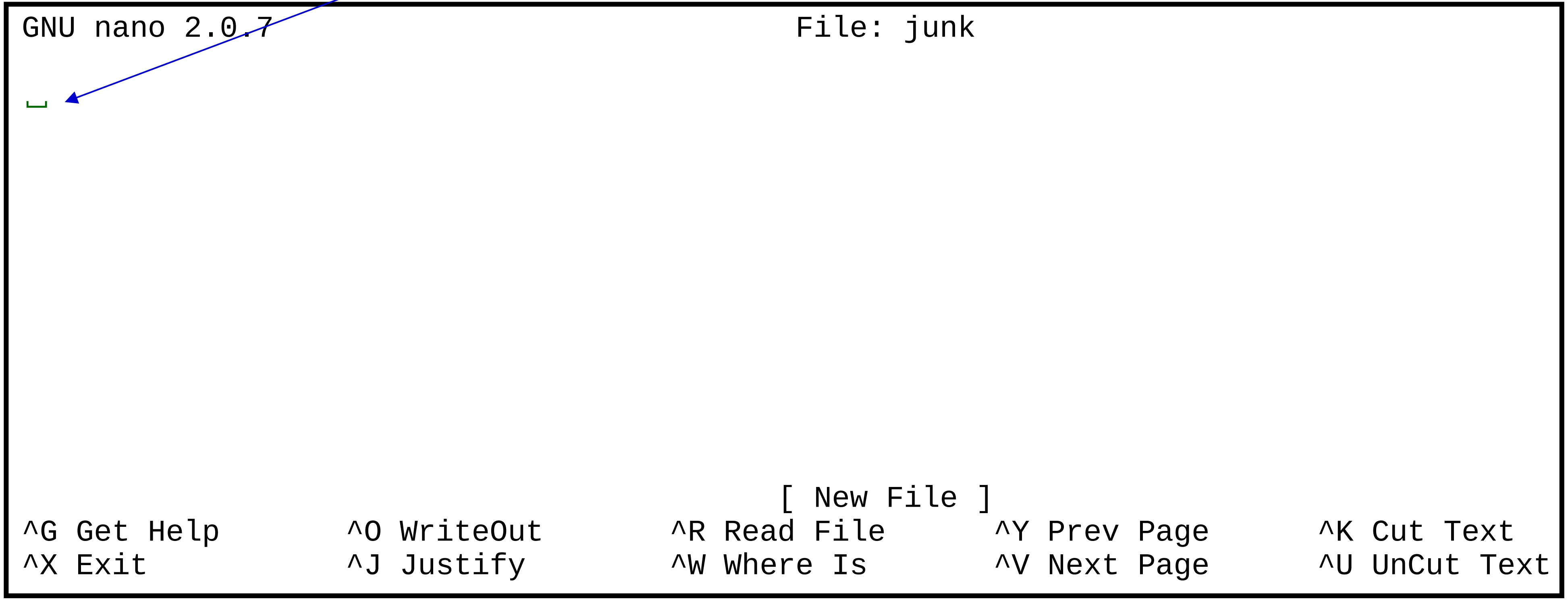
```
$ cd tmp
$ nano junk
```



```
GNU nano 2.0.7                              File: junk











                                           [ New File ]
^G Get Help        ^O WriteOut        ^R Read File       ^Y Prev Page       ^K Cut Text
^X Exit            ^J Justify         ^W Where Is        ^V Next Page       ^U UnCut Text
```

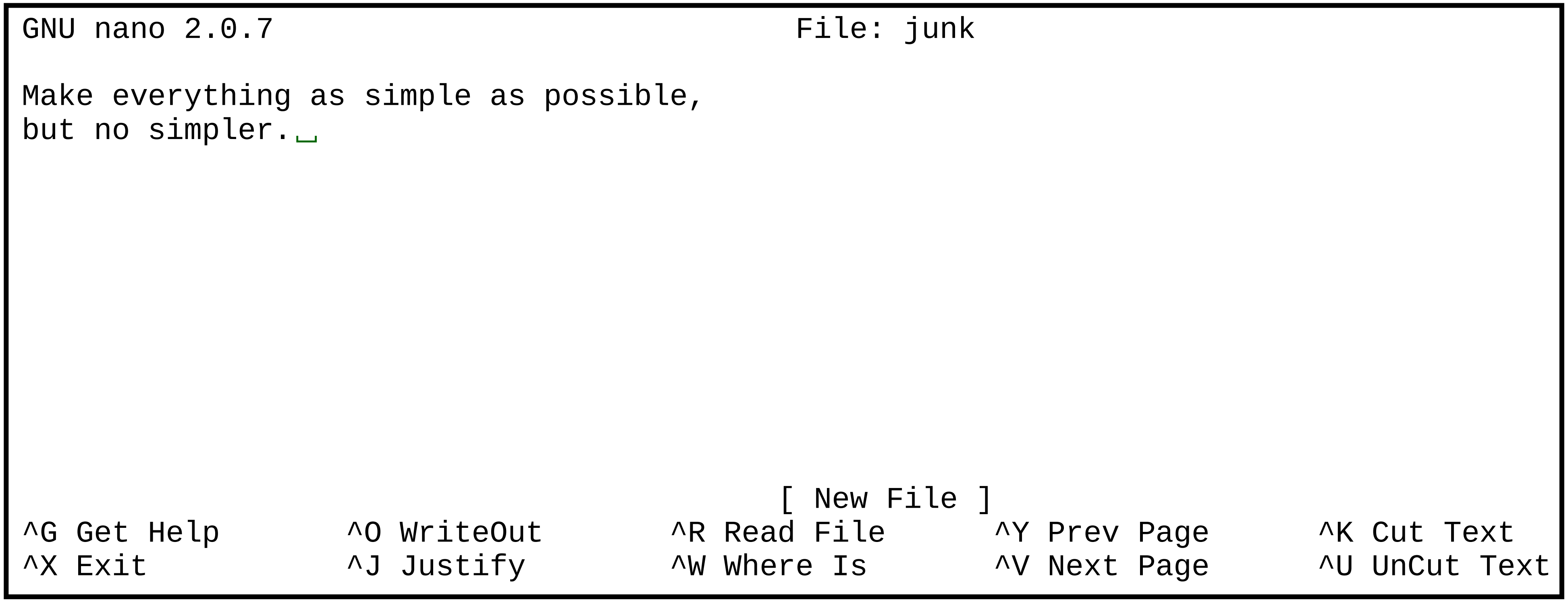
```
$ cd tmp
$ nano junk
```

That's your cursor



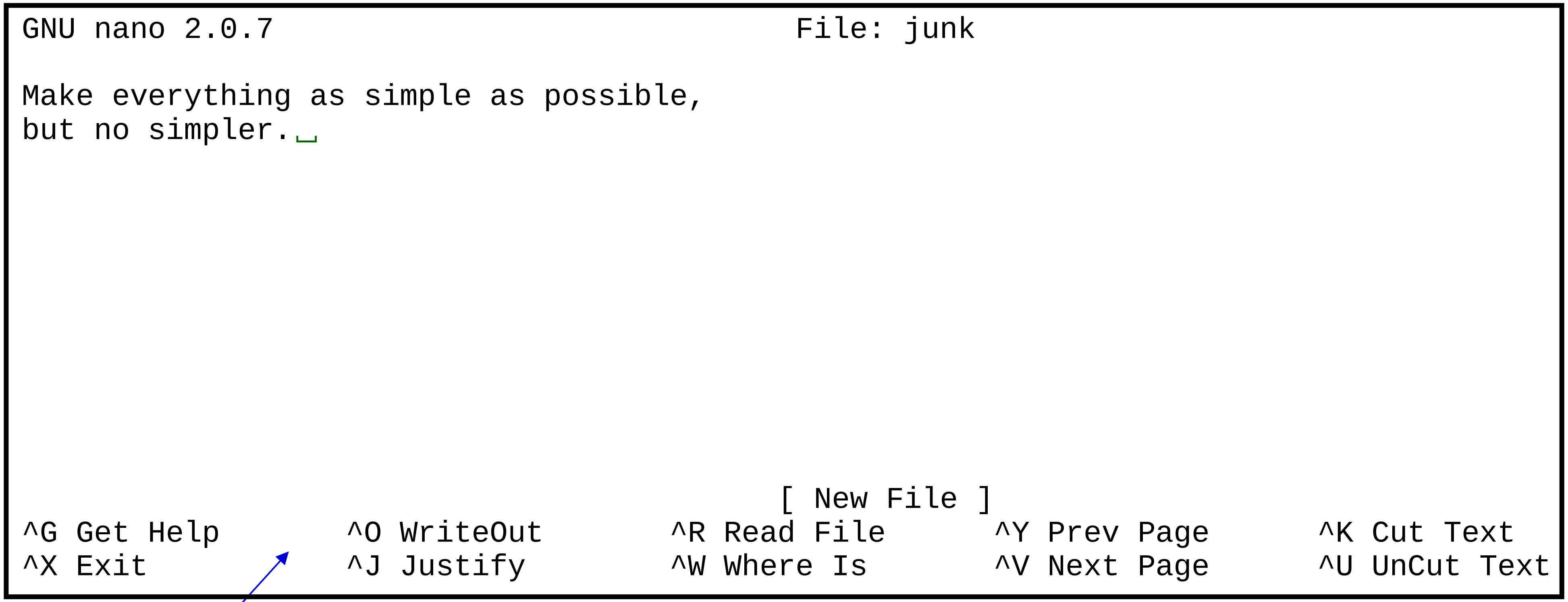
```
GNU nano 2.0.7                                File: junk

␣









                                            [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text
```

```
$ cd tmp
$ nano junk
```



```
GNU nano 2.0.7                                File: junk

Make everything as simple as possible,
but no simpler.␣










                                             [ New File ]
^G Get Help        ^O WriteOut        ^R Read File        ^Y Prev Page        ^K Cut Text
^X Exit            ^J Justify         ^W Where Is         ^V Next Page        ^U UnCut Text
```

```
$ cd tmp
$ nano junk
```



```
GNU nano 2.0.7                              File: junk

Make everything as simple as possible,
but no simpler.␣











                                            [ New File ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text
```

^O means "Control + O" (to save changes)

```
$ cd tmp
$ nano junk
```

```
GNU nano 2.0.7                           File: junk

Make everything as simple as possible,
but no simpler.␣











                                        [ New File ]
^G Get Help       ^O WriteOut       ^R Read File       ^Y Prev Page       ^K Cut Text
^X Exit           ^J Justify        ^W Where Is        ^V Next Page       ^U UnCut Text
```

^X to exit back to the shell

```
$ cd tmp
$ nano junk
$
```

nano doesn't leave any output on the screen after it exits

```
$ cd tmp
$ nano junk
$ ls
junk
$
```

← but it has created the file

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

$\leftarrow$ use -s to show sizes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

use -s to show sizes

reported in disk blocks

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$
```

use -s to show sizes

reported in disk blocks

a less helpful default

may have been possible…

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

← use -h for human-friendly output

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

use -h for human-friendly output

number of bytes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$
```

use -h for human-friendly output

number of bytes

rounded up because computer stores things on disk using blocks of 512 bytes

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$
```

← remove (delete) file

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$
```

remove (delete) file

there is no (easy) un-delete!

```
$ cd tmp
$ nano junk
$ ls
junk
$ ls -s
   1  junk
$ ls -s -h
 512  junk
$ rm junk
$ ls
$
```

← check that it's gone

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$
```

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$
```

change working directory to `/users/vlad`

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$
```

`rm` only works on files

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$ rmdir tmp
```

use `rmdir` to remove directories

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove `tmp': Directory not empty
$
```

but it only works when the directory is empty

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove `tmp': Directory not empty
$
```

but it only works when the directory is empty

(safety feature)

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove `tmp': Directory not empty
$ rm tmp/junk
$
```

so get rid of the directory's contents…

```
$ pwd
/users/vlad/tmp
$ nano junk
$ ls
junk
$ cd ..
$ rm tmp
rm: cannot remove `tmp': Is a directory
$ rmdir tmp
rmdir: failed to remove `tmp': Directory not empty
$ rm tmp/junk
$ rmdir tmp
$
```

…then get rid of the directory

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$
```

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

from here…

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

from here…

…to here

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$
```

move a file (or directory)

from here…

…to here

renames the file!

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$
```

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

current working directory

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$
```

move /users/vlad/tmp/quotes.txt
to /users/vlad/quotes.txt

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$
```

nothing left in `tmp`

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$ ls quotes.txt
quotes.txt
```

$\leftarrow$ `quotes.txt` now in this directory

```
$ pwd
/users/vlad/tmp
$ mkdir tmp
$ nano tmp/junk
$ ls tmp
junk
$ mv tmp/junk tmp/quotes.txt
$ ls tmp
quotes.txt
$ mv tmp/quotes.txt .
$ ls tmp
$ ls quotes.txt
quotes.txt
```

ls with a file or directory argument lists that file or directory

```
$ cp quotes.txt tmp/quotations.txt
$
```

copy a file

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or direct
tmp/quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or direct
tmp/quotations.txt
$ cp tmp/quotations.txt .
$ ls quotations.txt
quotations.txt
$
```

```
$ cp quotes.txt tmp/quotations.txt
$ ls quotes.txt tmp/quotations.txt
quotes.txt   tmp/quotations.txt
$ rm quotes.txt
$ ls quotes.txt tmp/quotations.txt
ls: cannot access quotes.txt: No such file or direct
tmp/quotations.txt
$ cp tmp/quotations.txt .
$ ls quotations.txt
quotations.txt
$
```

this is a directory, so the copy has the same name as the original file

pwd	print working directory
cd	change working directory
ls	listing
.	current directory
..	parent directory
mkdir	make a directory
nano	text editor
rm	remove (delete) a file
rmdir	remove (delete) a directory
mv	move (rename) a file or directory
cp	copy a file

created by

Greg Wilson

August 2010

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.