IPPP Code Club

Toolkit I

Yannick Ulrich

IPPP, University of Durham

5TH NOVEMBER 2021

Code & slides:
`https://gitlab.com/yannickulrich/computing-toolkit`

disclaimer

- this will be entirely in the shell
- I will assume you have an Linux environment (macOS will probably work, WSL maybe as well)

$\Rightarrow$ use SSH if necessary (MobaXterm on Windows, `ip3-login` for IPPP members)

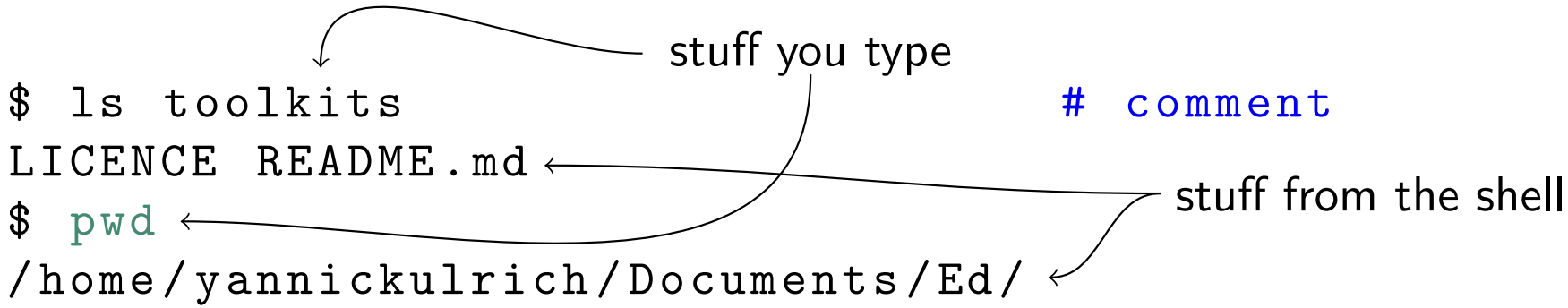
disclaimer

- this will be entirely in the shell
- I will assume you have an Linux environment (macOS will probably work, WSL maybe as well)

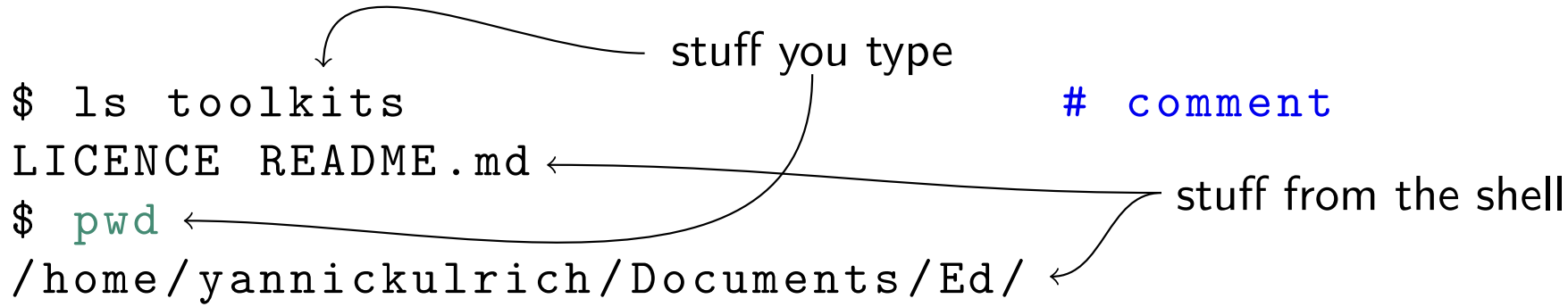
$\Rightarrow$ use SSH if necessary (MobaXterm on Windows, `ip3-login` for IPPP members)

- ... we probably need a part two

the command line



the command line

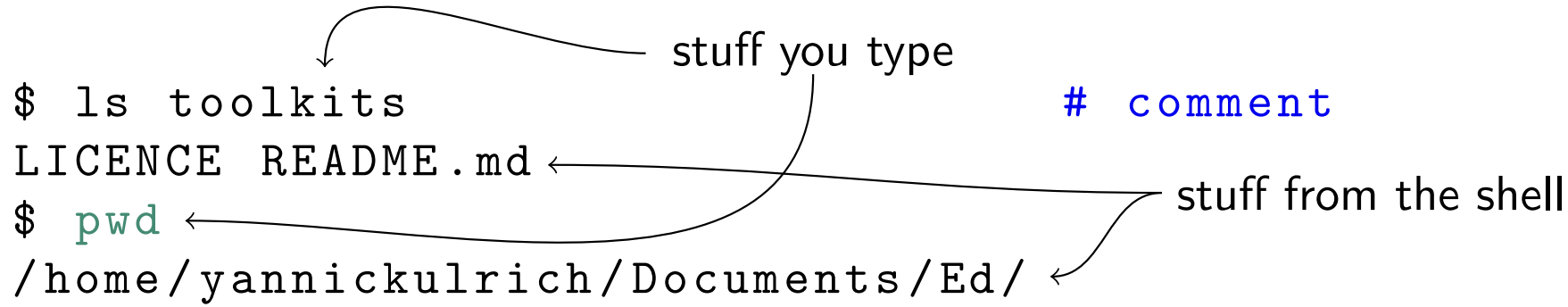


some commands

- `cd` change directory
- `ls` list current directory
- `cp` copy file
- `mv` move file
- `cat` output file to screen
- `pwd` where am I?
- `rm` remove file
  can't be undone!

things to try

the command line



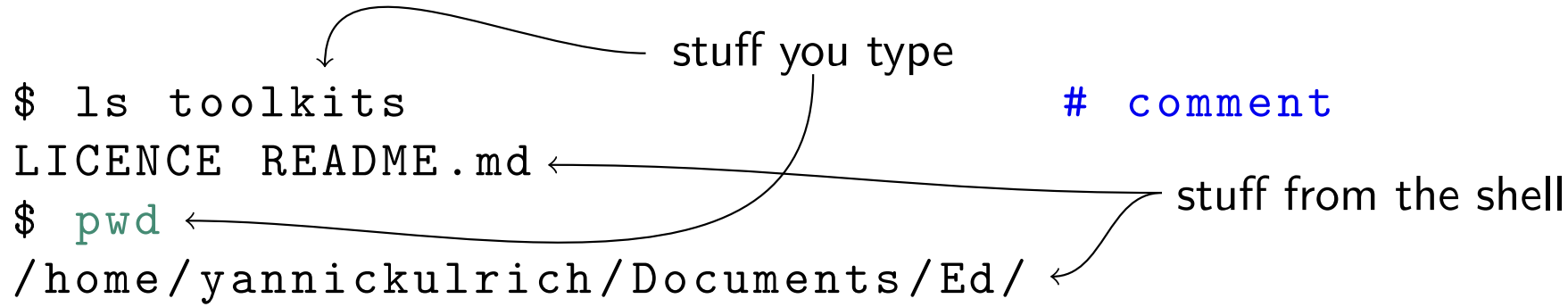
some commands

- `cd` change directory
- `ls` list current directory
- `cp` copy file
- `mv` move file
- `cat` output file to screen
- `pwd` where am I?
- `rm` remove file
  can't be undone!

things to try

- `$ ls sli<hit tab>`

the command line



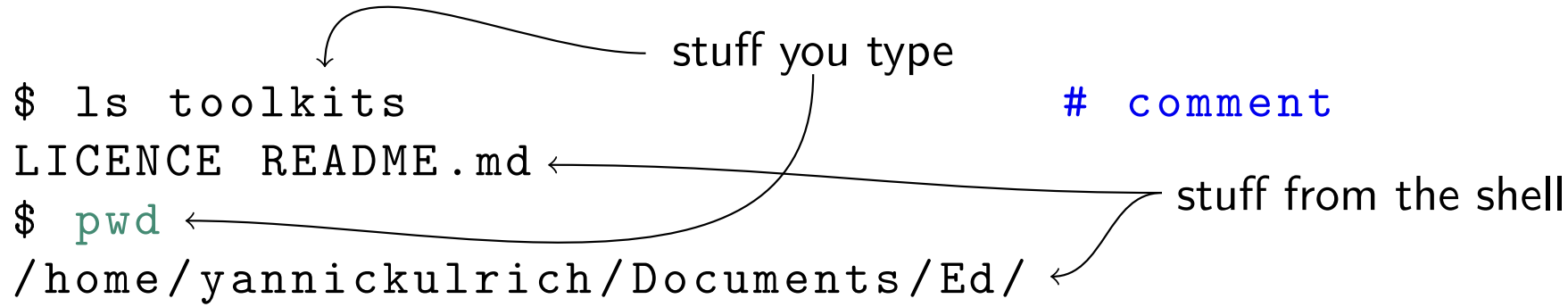
some commands

- `cd` change directory
- `ls` list current directory
- `cp` copy file
- `mv` move file
- `cat` output file to screen
- `pwd` where am I?
- `rm` remove file
  can't be undone!

things to try

- `$ ls sli<hit tab>`
- `$ <hit up arrow>`

the command line



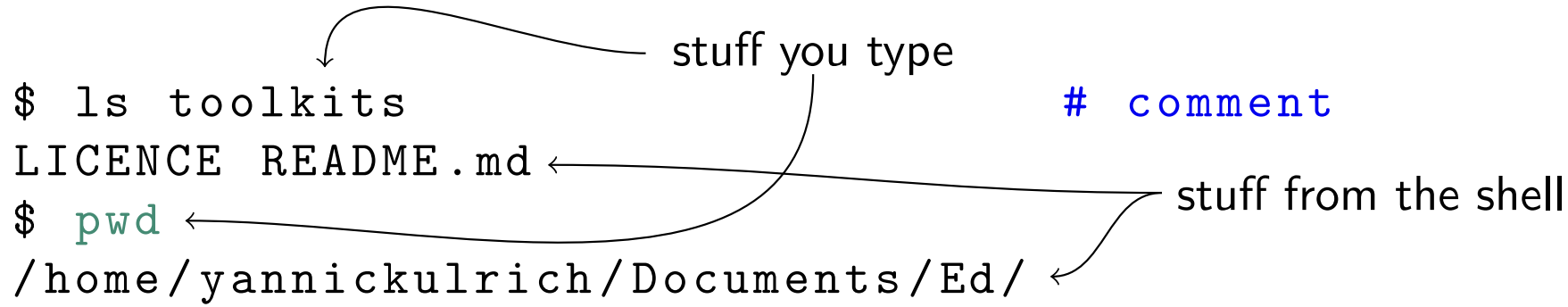
some commands

- `cd` change directory
- `ls` list current directory
- `cp` copy file
- `mv` move file
- `cat` output file to screen
- `pwd` where am I?
- `rm` remove file
  can't be undone!

things to try

- `$ ls sli<hit tab>`
- `$ <hit up arrow>`
- what are the directories `.` and `..`

the command line



some commands

- `cd` change directory
- `ls` list current directory
- `cp` copy file
- `mv` move file
- `cat` output file to screen
- `pwd` where am I?
- `rm` remove file
  can't be undone!

things to try

- `$ ls sli<hit tab>`
- `$ <hit up arrow>`
- what are the directories `.` and `..`
- `$ <hit ctrl-r>ls`

Demo time

how does the computer know what `ls` means?

the $PATH

how does the computer know what `ls` means?

- short answer: it doesn't but it knows where to look
- `$PATH` lists the search path

how does the computer know what ls means?

- short answer: it doesn't but it knows where to look
- $PATH lists the search path
- view yours

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:..
```

- to install stuff: just put it somewhere in $PATH

```
$ cp stuff /bin
cp: cannot create regular file '/bin/stuff
    ↪ ': Permission denied
```

the $PATH

how does the computer know what ls means?

- short answer: it doesn't but it knows where to look
- $PATH lists the search path
- view yours

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:..
```

- to install stuff: just put it somewhere in $PATH

```
$ cp stuff /bin
cp: cannot create regular file '/bin/stuff
    ↪ ': Permission denied
```

- temporary solution: add to $PATH

```
$ export PATH=/my/new/path/bin:$PATH
```

the $PATH

how does the computer know what ls means?

- short answer: it doesn't but it knows where to look
- $PATH lists the search path
- view yours

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:..
```

- to install stuff: just put it somewhere in $PATH

```
$ cp stuff /bin
cp: cannot create regular file '/bin/stuff
    ↪ ': Permission denied
```

- temporary solution: add to $PATH

```
$ export PATH=/my/new/path/bin:$PATH
```

- $LD_LIBRARY_PATH, $PYTHONPATH, ...

Demo time

git basics

- `git clone <url>` get a copy
- `git status`, `git diff`: changes
- `git log`: view history
- `git add -p`: staging
- `git commit -m "<msg>"`: commit
- `git push`: send changes (can't be undone)
- `git pull`: fetch changes
- see last year's talk for details

gitlab.com/yannickulrich/git-tutorial

xkcd.com/1597

Demo time

editing files: vim

learn to use `vim`

(it'll save time in the long run, yes, trust me, it will)

learn to use `vim`
(it'll save time in the long run, yes, trust me, it will)

- first things first: to exit `esc` and
  `:wq`

Stack Overflow: Helping One Million Developers Exit Vim

learn to use `vim`
(it'll save time in the long run, yes, trust me, it will)

- first things first: to exit `esc` and `:wq`
- open `vim folder/some-file`
- default text editor on most systems

Stack Overflow: Helping One Million Developers Exit Vim

learn to use `vim`
(it'll save time in the long run, yes, trust me, it will)

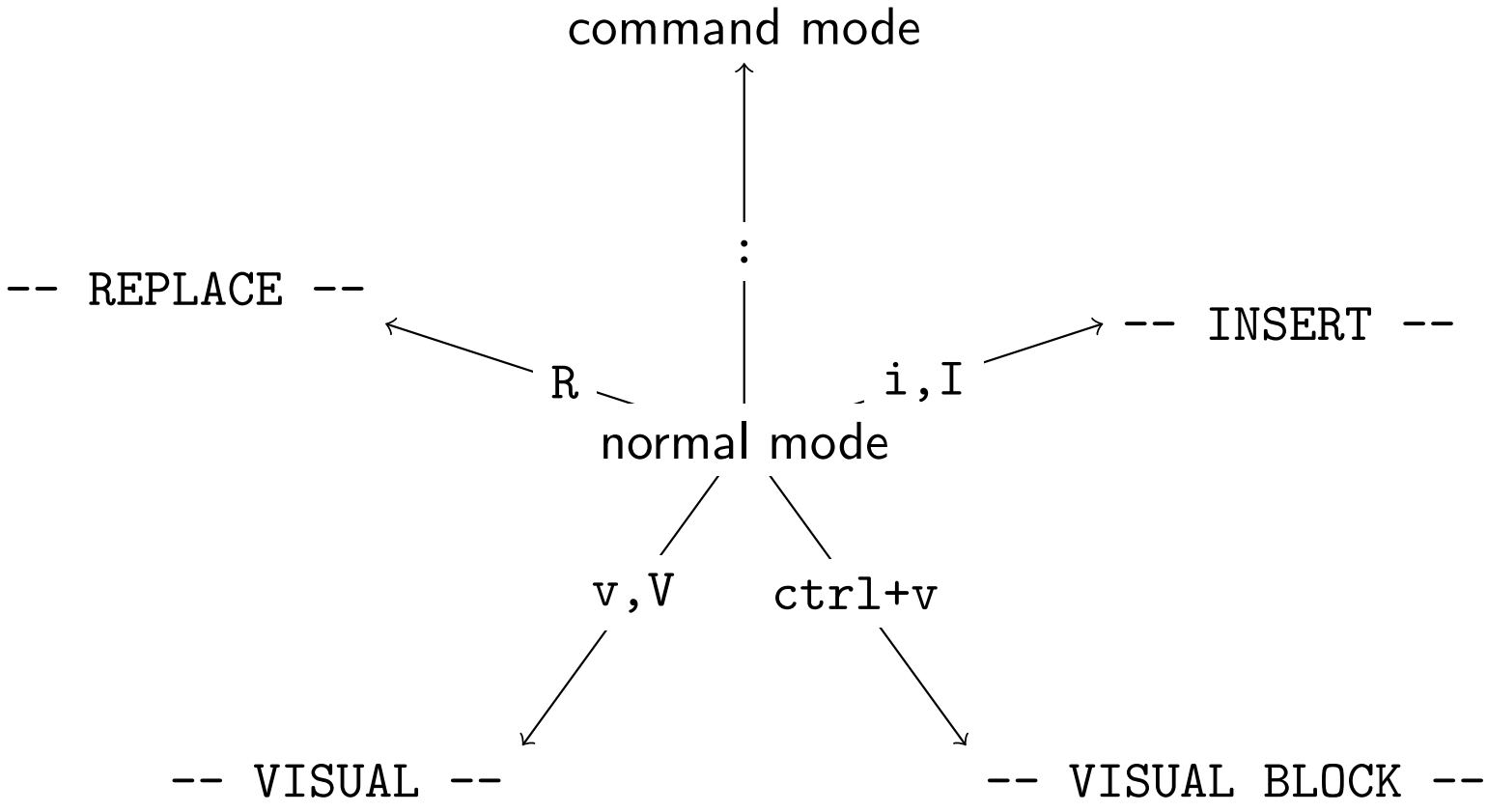
- first things first: to exit `esc` and `:wq`
- open `vim folder/some-file`
- default text editor on most systems
  (can be changed, won't tell you how)

Stack Overflow: Helping One Million Developers Exit Vim

learn to use `vim`
(it'll save time in the long run, yes, trust me, it will)

- first things first: to exit `esc` and `:wq`
- open `vim folder/some-file`
- default text editor on most systems
  (can be changed, won't tell you how)
- use it for a week for all your editing (incl. LaTeX)
- ... and make an effort to use it well
- you'll most likely be faster than before

Stack Overflow: Helping One Million Developers Exit Vim



- to return `esc`
- play with `vimtutor`
- consider `gvim`

Demo time

the static page

0. have a GitHub account and tell Ryan about it
1. install Hugo to ∼/.local/bin/
   https://github.com/gohugoio/hugo/releases
2. clone https://github.com/eidoom/computing-club-site

   note: you need to add the flag --recursive
3. start Hugo locally hugo server
4. find the manual
5. make an authors page using vim at
   authors/<yourname>_index.md
6. test things locally
7. pull any remote changes, stage, commit, push

further reading

- eleven part toolkit lecture `https://missing.csail.mit.edu/`
- fancy things you can do `https://youtu.be/sCZJblyT_XM`

bonus content

make a list of tags without 3rd party tools

make a list of tags without 3rd party tools

```
function findtags() {
  grep 'tags␣=' $1    \                  # find tags =
    | tr '[]' '|'|cut -f2 -d'|' \         # get [stuff]
    | tr ',' '\n' \                       # split lines
    | tr -d '␣"'
}
find . -name '*.md' \                     # find files
 | while read line ; \                    # loop over
   do \                                   # them
     findtags $line ; \                   #
   done \
 | sort | uniq                            # sort & uniq
```