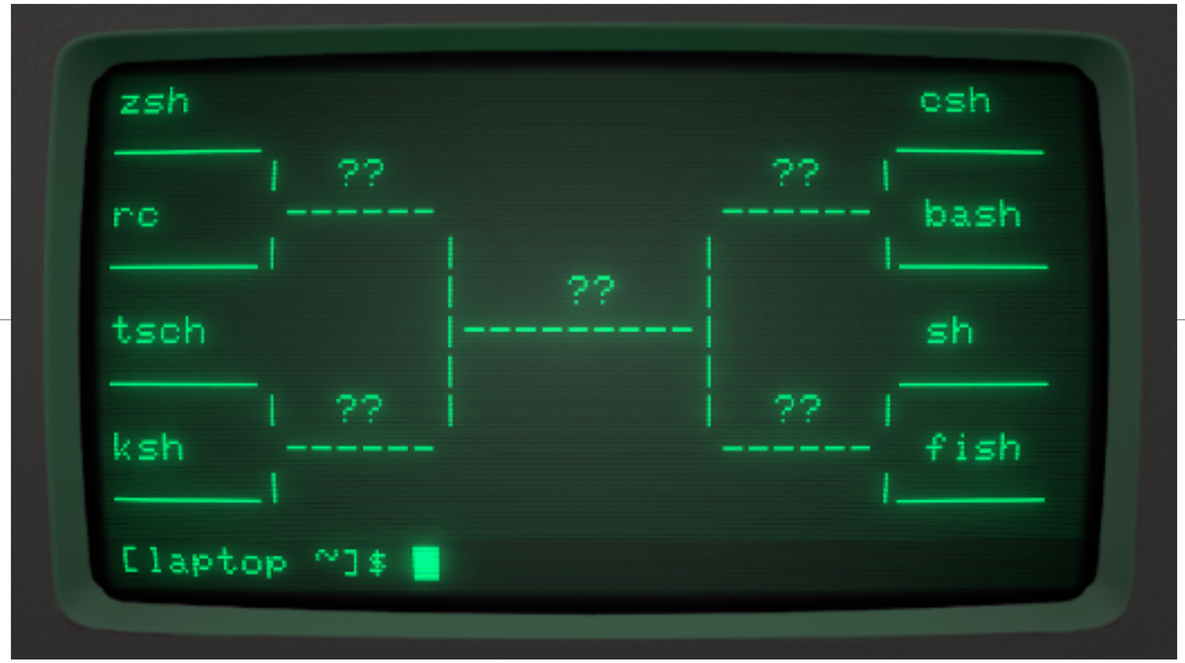
The Unix Shell

Tiago M. D. Pereira



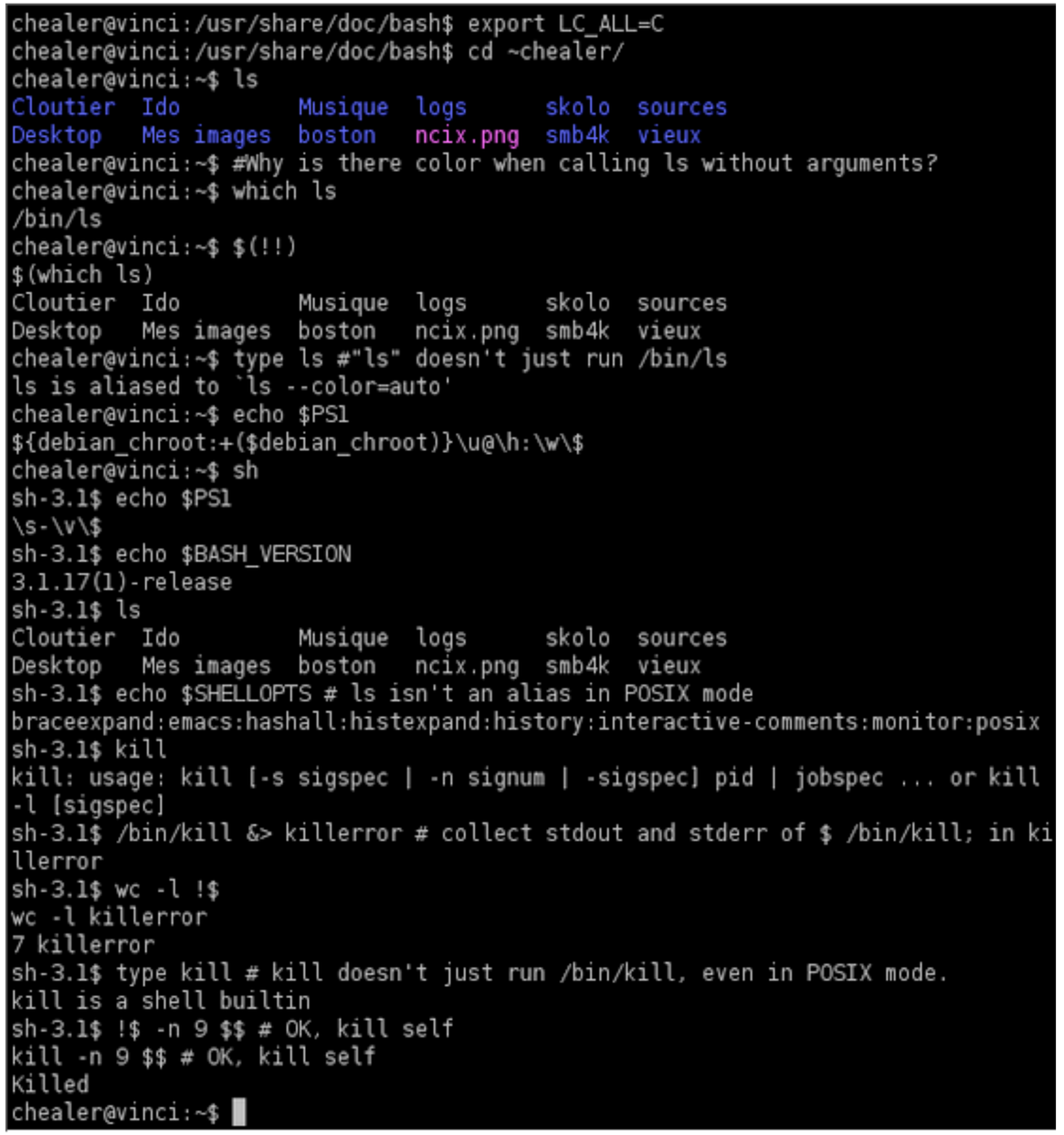
Slides: https://folk.uio.no/tiago/teaching/unix2017

UiO : **Institute of Theoretical Astrophysics**

What is the shell?

- Command line interface to operative system
- Interpreter to direct the operation of the computer

Don’t confuse the shell with the terminal program (XTerm, Konsole, iTerm)



Why should YOU learn about shell?

- Become faster at using computers
- Become a power user
- Save a lot of time in repetitive tasks
- Have more control over the machine
- Pad your CV

Today’s lecture in a nutshell

1. Different shells and shell scripting
2. Basic concepts, text editors and tools
3. Processing output
4. Shell scripting basics
5. Hands-on tutorials

1. Different Unix Shells

- bash: Bourne Again Shell (1989)
- tcsh: TENEX C Shell (1981) (from C shell, 1978)
- zsh: Z Shell, improved bash (1990)
- ksh: Korn Shell (1983)
- Many others...
- *Choose one and stick to that!*

What shell do I have?

Enter: **echo $0**
in your terminal

2. Basic concepts, text editors and tools

```
[tiago@beehive ~]$ env
HOSTNAME=beehive.uio.no
TERM=xterm
SHELL=/local/gnu/bin/bash
SSH_TTY=/dev/pts/25
LC_ALL=en_GB.UTF-8
USER=tiago
PATH=/astro/local/anaconda3/
bin:/astro/local/bin:/usr/
local/bin:/usr/bin
PWD=/mn/stornext/u3/tiago
_=/usr/bin/env
```

```
$ echo $HOSTNAME
beehive.uio.no

Bash:
$ export MYVAR='Testing...'
$ echo $MYVAR
Testing...

Tcsh:
$ setenv MYVAR 'Testing...'
$ echo $MYVAR
Testing...
```

$PATH: a special variable

```
[tiago@beehive ~]$ echo $PATH
PATH=/astro/local/bin:/usr/local/bin:/usr/bin
```

- $PATH defines where the shell looks for programmes, and in which order
- Can include several locations separated by :

The shell prompt

```
[tiago@beehive ~]$
```


```
[tiago@beehive ~]$ echo $PS1
[\u@\h \W]\$
[tiago@beehive ~]$ export PS1='What is your command? '
What is your command?
```

Tab completion

```
[tiago@beehive ~]$ ls zo<TAB>
```

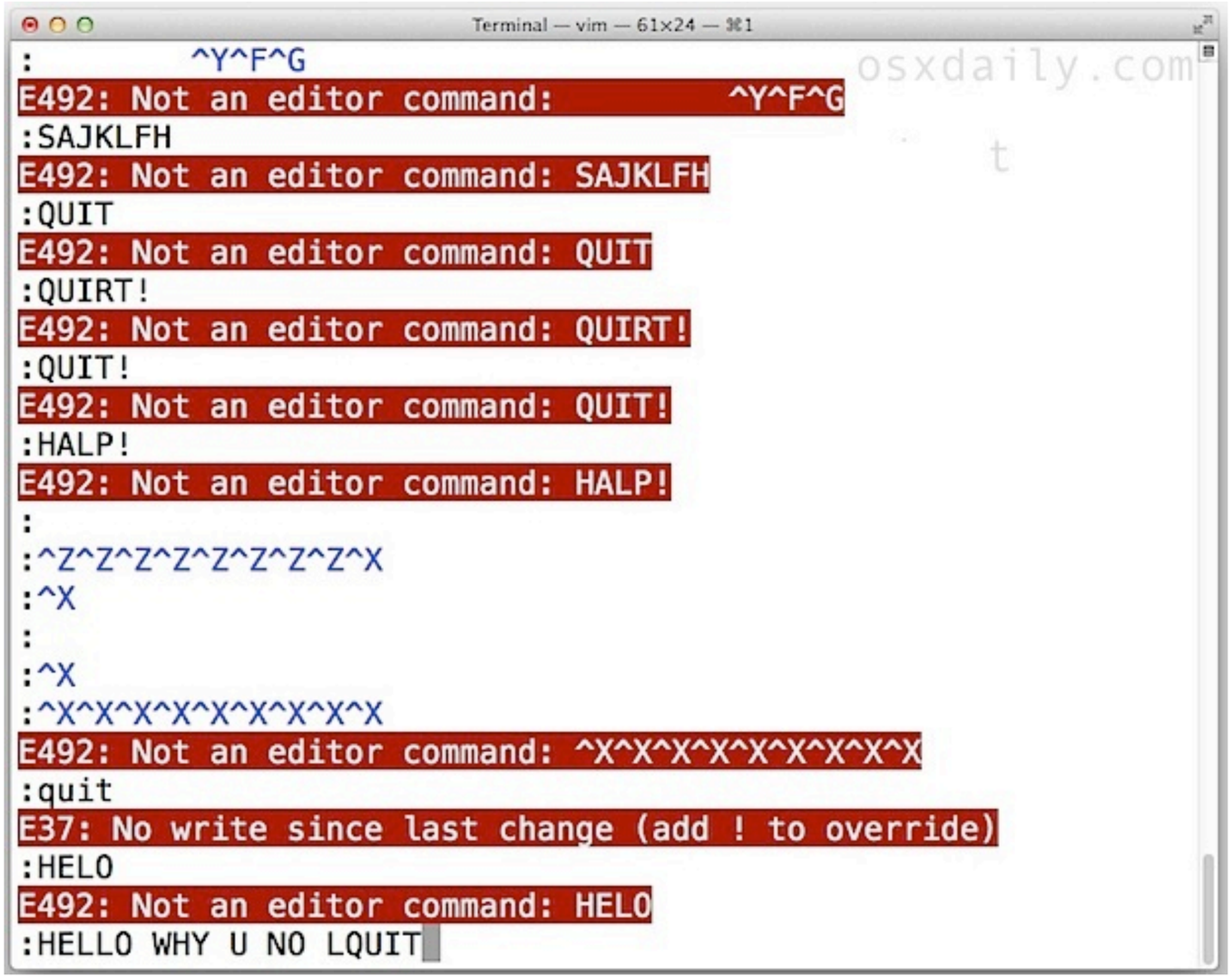
Tab completion

```
[tiago@beehive ~]$ ls zoom_Aug2004_1024x1024_noticks.mov
```

Tab completion

Text editors, from easiest to most complex:

- **pico** or **nano**
- **jed**
- **emacs**
- **vi** or **vim**
- **ed**



Text editors, from easiest to most complex:

- **pico** or **nano**
- **jed**
- **emacs**
- **vi** or **vim**
- **ed**

```
$ ed
help
?
h
Invalid command suffix
?
?
^C
?
exit
?
quit
?
^Z
$ killall ed
```

Too many shell commands? Get help!

- Can’t remember what a command is used for, or what its options are? Just use **man** <command>
- Most Unix commands have a “manual page”, a reference page for description and options. Often helpful!
- When you enter man, it starts your pager (default: **less**). Press “q” to quit, arrows to move around the page, “/” to search. E.g. “/read” will search for “read” string

Now we’re getting serious: useful commands

- The **tar** command: create and open archives
- *Job interview question: write down a working tar command on your first try*
- Unpack an archive to the current directory:

```
$ tar zxvf my_archive.tar.gz
```

- Create an archive from a directory:

```
$ tar zcvf my_archive.tar.gz directory/
```

Now we’re getting serious: useful commands

- **wget** or **curl** (MacOS): download files from the shell
- Using **wget** to download a file:

```
$ wget http://folk.uio.no/tiago/cfitsio3420.tar.gz
```

- Using **curl** to download a file:

```
$ curl -O http://folk.uio.no/tiago/cfitsio3420.tar.gz
```

Now we’re getting serious: useful commands

- Use **grep** to find patterns in files

```
$ grep <pattern> <file(s)>
```

```
$ grep National License.txt
of the National Aeronautics and Space Administration.
```

- Case insensitive:

```
$ grep -i
```

Show line in file:

```
$ grep -n
```

Recursive:

```
$ grep -r
```

- Use **zgrep** for compressed files

Now we’re getting serious: useful commands

- Use **find** to find files

```
$ find . -name myfile.txt
$ find . -name '*.c'
$ find . -name mydir -type d
$ find /mydir -type f -not name '*.h'
$ find . -mtime 1   # last 24h
```

Now we’re getting serious: useful commands

- Use **find -exec** to find files and do an action

```
$ find . -name '*.tmp' -delete
$ find . -name '*.c' -exec ls -lah {} \;
$ find . -type d -exec chmod ugo+x {} \;
$ find . -name '*.gz' -exec zgrep -rni national {} \;
$ find . -type f -exec ls -lsh {} \; | sort -n -r | head -5
```

Now we’re getting serious: useful commands

- Quickly parsing files: **less** is **more**, **tail**, **head**

```
$ less myfile.txt  # press q to quit
$ head wcsutil.c
#include <math.h>
#include "fitsio2.h"
#define D2R 0.01745329252
#define TWOPI 6.28318530717959
(...)
```

Now we’re getting serious: useful commands

- **tail**: show last lines of file

```
$ tail -n 5 wcsutil.c
    /*      convert to pixels  */
    *xpix = dx / xinc + xrefpix;
    *ypix = dy / yinc + yrefpix;
    return(*status);
}
```

- Most useful with **-f**, follow option:

```
$ tail -f changing_file.log
```

Now we’re getting serious: useful commands

- **wc**: count words, lines, and characters

```
$ wc /etc/passwd
  36   76 1830 /etc/passwd
```

```
$ wc -l *.c
    1377 buffers.c
    7456 cfileio.c
     508 checksum.c
(...)
  136182 total
```

A summary of shell commands we've learned

Today:

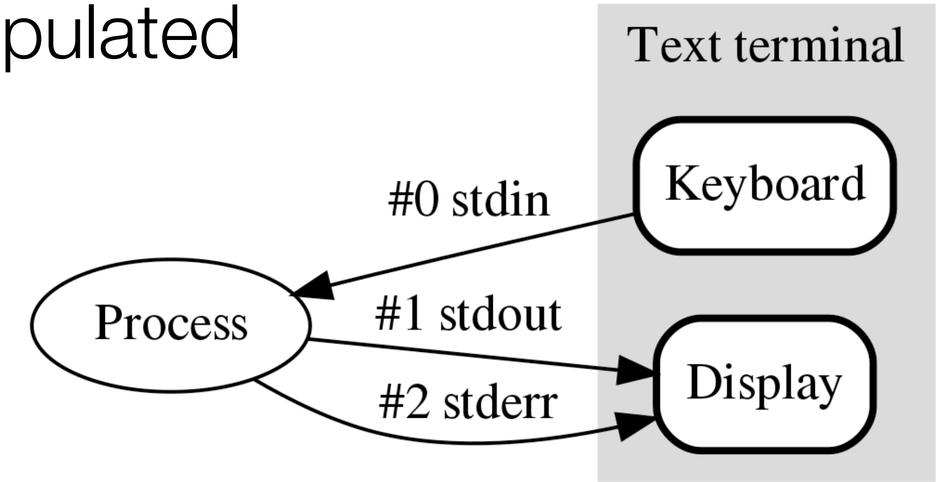
du	Display disk usage statistics
find	Find files and do something with them
grep	Search for pattern
man	Read manual page
tail / head	Print last or first lines of file
tar	Create and unpack archives
wc	Word and line count
wget	Download files

Last course:

chmod	Change permissions
chown	Change ownership
df	See disks and free space
kill	Kill or send signals to processes
nice / renice	Set process priority
ps	See process list
top	See processes and resources used
su / sudo	Change user / run as super user
uname	See operative system name
w	See who's logged in and doing what

3. Processing Output

- Output and input in terminal can be manipulated
- Standard streams: *stdin*, *stdout*, *stderr*
- Redirection: **>** and **<**
- Pipes: **|**
- *Other operators:* **&&**, **||**



Redirection: stdout and stderr

- The **>** operator redirects *stdout* to a new file

```
$ ls -lah > test.txt
```

- The **>>** operator redirects stdout to an existing file (appends):

```
$ echo ‘The END’ >> test.txt
```

- To redirect *stderr* to a different file, use **2>** (or **2>>**):

```
$ ./myscript.py 1> output.txt 2> error.txt
```

- To redirect both *stderr* and *stdout* to the same file, use **&>**:

```
$ du -hs * &> result.txt
```

Pipe: send output to a programme

- The | operator sends *stdout* to a programme:

```
$ ls -lah | grep old
$ ls *.c | wc -l
$ du | sort -n -r
$ grep -i print testprog.c |grep HDU |grep -v Total
$ ps aux | grep tiago | bzip2 > processes.bz2
```

- It can be used multiple times!

Other operators

- Operator **;** always runs second command after the first:

```
$ tar zxvf archive.tar.gz ; ls -lah
```

- Operator **&&** will only run second command if first is successful:

```
$ tar zxvf archive.tar.gz && rm -rf archive.tar.gz
```

- Operator **||** will only run second command if first fails:

```
$ tar zxvf archive.tar.gz || wget http://archive.gz
```

4. Shell Scripting Basics

- Shell is a basic programming language
- A shell script is a file with a bunch of instructions
- Used in place of a *normal* programme for quick tasks
- Always decide: do it by hand / shell script / normal programme? The tradeoff depends on the task!

When to use shell scripts

- When you have a lot of simple, repetitive tasks
  (e.g. rename thousands of files)
- When you want to combine different commands and use the output of some as input to others
- Always a personal preference.
  My rule of thumb: when more than 10 lines, use Python.

When NOT to use shell scripts

- Need high performance (number crunching)
- Need to use arrays or complex data structures
- To manipulate graphics or GUIs
- Need direct access to hardware
- Anything that takes long to programme!

Components of script

Shebang

Commands

```
#!/bin/bash
echo Hello from $HOSTNAME
```

You can use many different interpreters by changing the shebang:

```
#!/bin/sh
#!/bin/csh
#!/usr/bin/perl
#!/usr/bin/env python
```

Simple shell script

```
#!/bin/bash
mkdir -p $HOME/files
cd $HOME/files
wget -c https://folk.uio.no/tiago/cfitsio3420.tar.gz
tar zxvf cfitsio3420.tar.gz
echo ‘Script finished!!!’
```

Tutorials and hands-on part

- These slides online: https://folk.uio.no/tiago/teaching/unix2017
- Find tutorial questions on same page