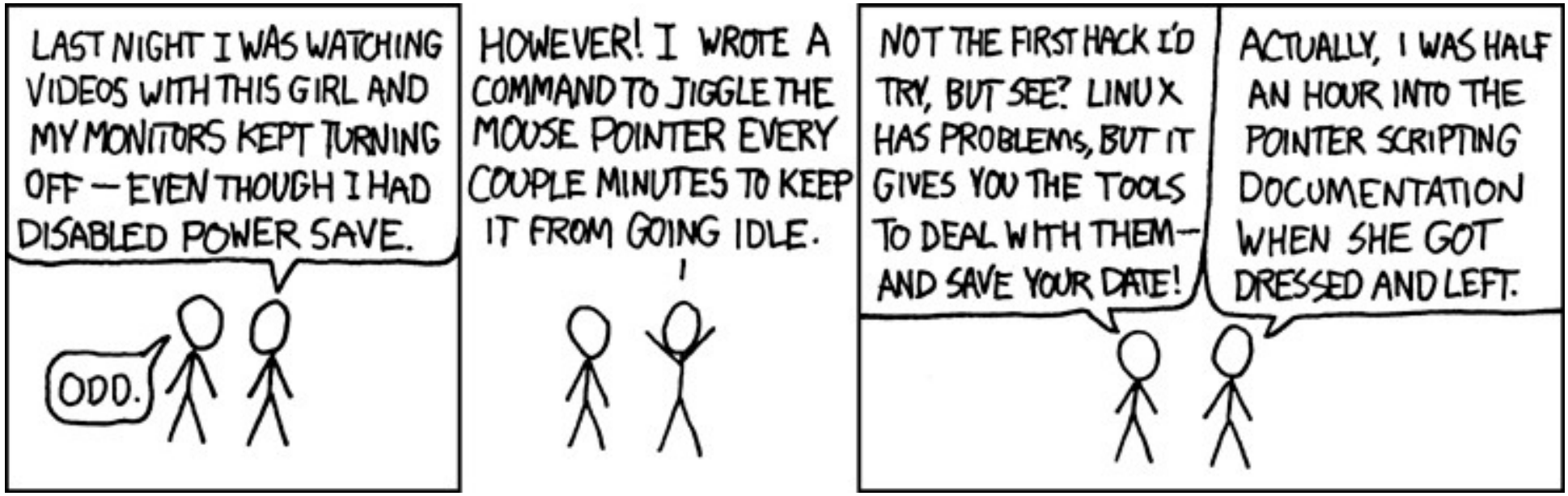
Pablo Alcain
pabloalcain@gmail.com

Herramientas GNU o: cómo aprendí a dejar de preocuparme y amar la línea de comandos

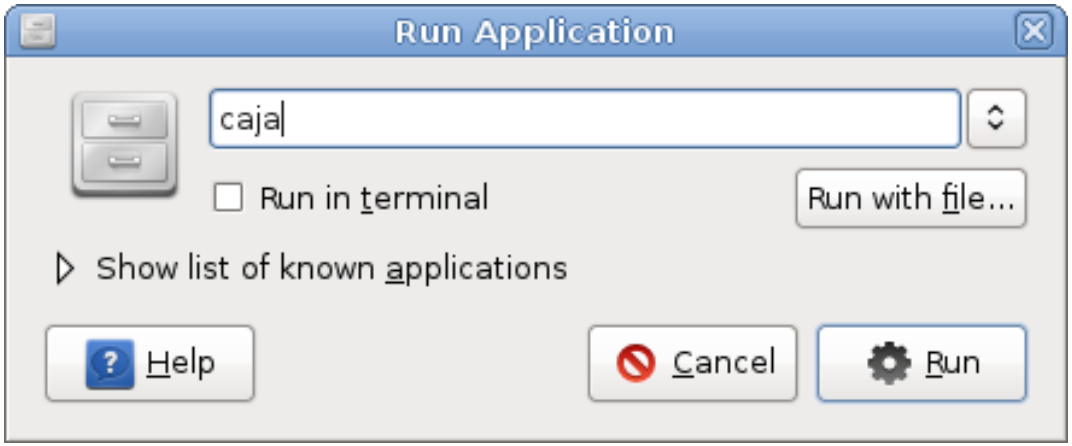
Command Line Fu



WTPC
Workshop en técnicas de programación científica

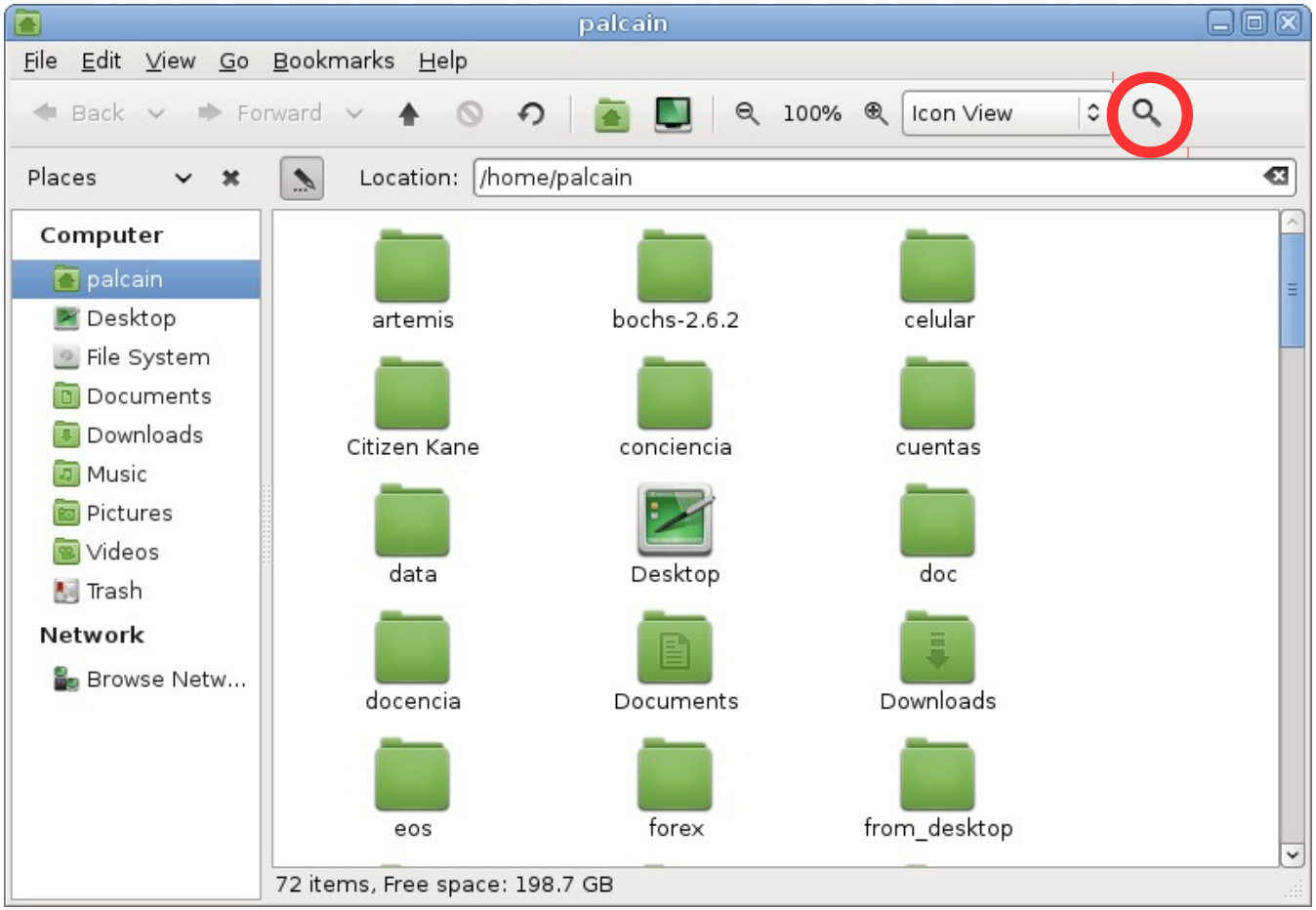
Vale la pena? Busquemos un archivo...

alt+F2: caja [en mi caso, cualquier otro file browser]



Vale la pena? Busquemos un archivo...

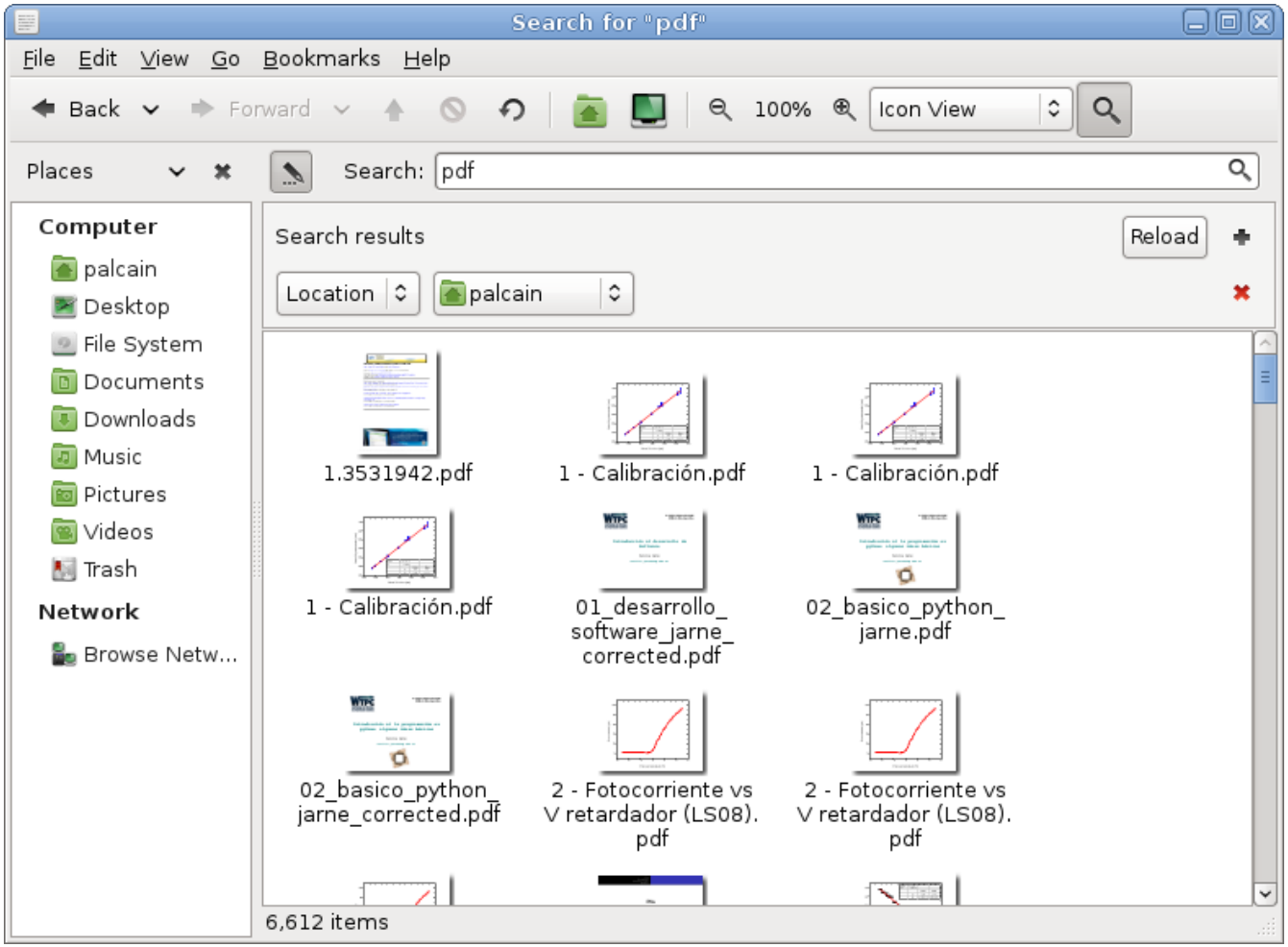
Buscar (será la lupita?)



WTPC
Workshop en técnicas de programación científica

Vale la pena? Busquemos un archivo...

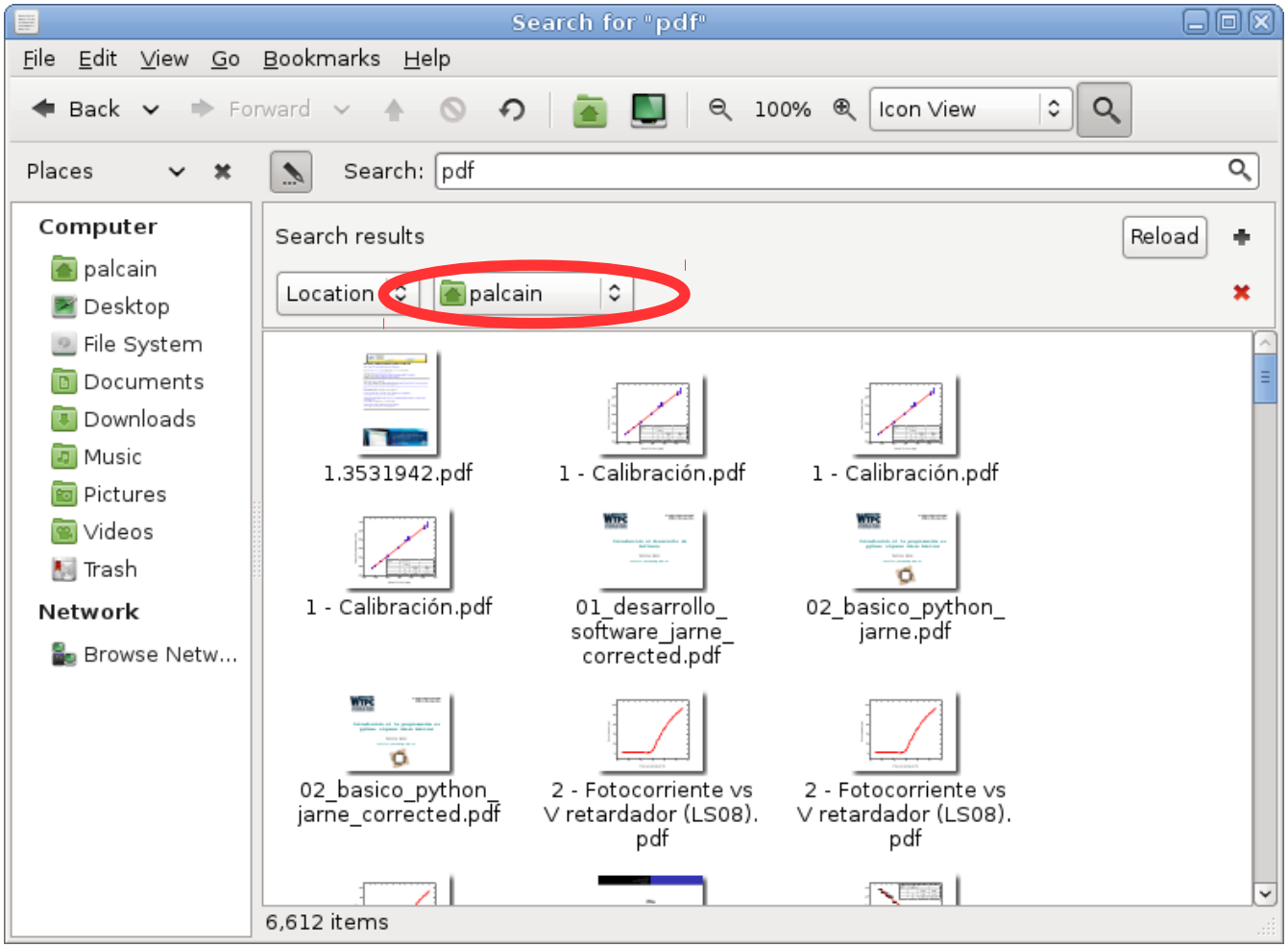
Buscar (será la lupita?)



WTPC
Workshop en técnicas de programación científica

Uh, yo lo quería en una carpeta

¿La tengo que elegir acá? ~/wtpc



WTPC
Workshop en técnicas de programación científica

Tarde o temprano nos aburrimos...

```
$ find ~/wtpc -name “*.pdf”
/home/palcain/wtpc/cuader/cuadernillo.pdf
/home/palcain/wtpc/charlas/architecture.pdf
/home/palcain/wtpc/charlas/optimization.pdf
/home/palcain/wtpc/charlas/linking_compiled.pdf
/home/palcain/wtpc/charlas/linking_python.pdf
/home/palcain/wtpc/charlas/documentation.pdf
/home/palcain/wtpc/charlas/licenses.pdf
/home/palcain/wtpc/charlas/debug_profile.pdf
```

Ni hablemos en una notebook, con el maldito touch pad

¿Por qué usamos la command line?

¿Por qué usamos la command line?

Porque no tenemos entorno gráfico

Porque queremos información rápida

Porque no necesitamos información gŕafica

Porque queremos hacer muchas cosas a la vez

Porque nos gusta

¿Por qué usamos la command line?

Porque no tenemos entorno gráfico

Porque queremos información rápida

Porque no necesitamos información gŕafica

Porque queremos hacer muchas cosas a la vez

Porque nos gusta

Porque la interfaz gráfica la usa cualquiera!
Nosotros somos mejores (?)

¿Qué hace a linux *LINUX*?

La “terminal”: herramientas **GNU**. Si no, pregúntenle a Stallman.

¿Qué hace a linux *LINUX*?

La “terminal”: herramientas **GNU**. Si no, pregúntenle a Stallman.

Algunas características de la terminal:

- Autocompleta con tab hasta que se repita algo
- Mantiene una historia de comandos usados: con la flecha para arriba los recorren; con ctrl-R pueden *buscar* comandos anteriores
- Muy fácil cambiar de usuario (no sólo a *root*, a cualquier otro)
- $man <comando> [mucho mejor que apretar F1 sin querer]
- Podemos redirigir el output y el input

Terminal: ver texto

Archivo de texto con inscriptos

```
$ cat alumnos.txt
Apellido, Nombre
Alcain, Pablo
Dolina, Alejandro
Singer, Paul
```

Terminal: ver texto

Archivo de texto con inscriptos (esto lo tengo que mostrar)

```
$ more cuadernillo.tex
\documentclass[10pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[inner=0.5in, outer=1in]{geometry}
\renewcommand{\familydefault}{\sfdefault}
\begin{document}
\section*{Lista de Participantes}
\begin{tabular}{|l | c | c | r|}
--More--(17%)
```

WTPC
Workshop en técnicas de programación científica

Terminal: ver texto

Archivo de texto con inscriptos (esto lo tengo que mostrar)

```
$ less cuadernillo.tex
\documentclass[10pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[inner=0.5in, outer=1in]{geometry}
\renewcommand{\familydefault}{\sfdefault}
\begin{document}
\section*{Lista de Participantes}
\begin{tabular}{|l | c | c | r|}
:
```

Terminal: ver texto

cat: simplemente pone todo el archivo en la terminal

less y more: más complejos, se puede buscar. less es más potente

```
$ less cuadernillo.tex
\documentclass[10pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[inner=0.5in, outer=1in]{geometry}
\renewcommand{\familydefault}{\sfdefault}
\begin{document}
\section*{Lista de Participantes}
\begin{tabular}{|l | c | c | r|}
:
```

WTPC
Workshop en técnicas de programación científica

Terminal: crear texto

Guardar la lista que sale de find en un archivo

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
```

Terminal: crear texto

Guardar la lista que sale de find en un archivo

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
$ ls
lista_archivos
```

Terminal: crear texto

Guardar la lista que sale de find en un archivo

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
$ ls
lista_archivos
$ cat lista_archivos
wtpc/cuader/cuadernillo.pdf
wtpc/charlas/architecture.pdf
wtpc/charlas/optimization.pdf
wtpc/charlas/linking_compiled.pdf
wtpc/charlas/linking_python.pdf
wtpc/charlas/documentation.pdf
wtpc/charlas/licenses.pdf
wtpc/charlas/debug_profile.pdf
```

WTPC
Workshop en técnicas de programación científica

Terminal: buscar texto

Busquemos en la lista de archivos las charlas de linking

```
$ grep linking lista_archivos
/home/palcain/wtpc/charlas/linking_compiled.pdf
/home/palcain/wtpc/charlas/linking_python.pdf
```

Terminal: buscar texto

Busquemos en la lista de archivos las charlas de linking

```
$ grep linking lista_archivos
/home/palcain/wtpc/charlas/linking_compiled.pdf
/home/palcain/wtpc/charlas/linking_python.pdf
```

```
$ grep -e <expresion regular>
$ grep -r recursivo: en un directorio
```

Terminal: editar texto

Cambiemos en la lista de archivos los palcain por pablohe

```
$ sed 's/palcain/pablohe/g' lista_archivos
/home/pablohe/wtpc/cuader/cuadernillo.pdf
/home/pablohe/wtpc/charlas/architecture.pdf
/home/pablohe/wtpc/charlas/optimization.pdf
/home/pablohe/wtpc/charlas/linking_compiled.pdf
/home/pablohe/wtpc/charlas/linking_python.pdf
/home/pablohe/wtpc/charlas/documentation.pdf
/home/pablohe/wtpc/charlas/licenses.pdf
/home/pablohe/wtpc/charlas/debug_profile.pdf
```

Terminal: editar texto

Cambiemos en la lista de archivos los palcain por pablohe

```
$ sed 's/palcain/pablohe/g' lista_archivos
/home/pablohe/wtpc/cuader/cuadernillo.pdf
/home/pablohe/wtpc/charlas/architecture.pdf
/home/pablohe/wtpc/charlas/optimization.pdf
/home/pablohe/wtpc/charlas/linking_compiled.pdf
/home/pablohe/wtpc/charlas/linking_python.pdf
/home/pablohe/wtpc/charlas/documentation.pdf
/home/pablohe/wtpc/charlas/licenses.pdf
/home/pablohe/wtpc/charlas/debug_profile.pdf
```

```
$ sed -i modifica el archivo “in place”
```

WTPC
Workshop en técnicas de programación científica

Claves: pipe

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
redirige el standard output a un archivo
```

Claves: pipe

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
redirige el standard output a un archivo
```

```
$ ./test.x < archivo_input
redirige el archivo al standard input
```

Claves: pipe

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
redirige el standard output a un archivo
```

```
$ ./test.x < archivo_input
redirige el archivo al standard input
```

```
$ find ~/wtpc -name “.pdf” | grep linking
la salida de la izq de | es el arg de la der
```

WTPC
Workshop en técnicas de programación científica

Claves: pipe

```
$ find ~/wtpc -name “*.pdf” > lista_archivos
redirige el standard output a un archivo
```

```
$ ./test.x < archivo_input
redirige el archivo al standard input
```

```
$ find ~/wtpc -name “.pdf” | grep linking
la salida de la izq de | es el arg de la der
```

```
$ ./un_programa 2>&1 output_y_errores
redirige el standard output y error a un archivo
```

WTPC
Workshop en técnicas de programación científica

Variables de entorno

Generalmente en mayúscula, se accede con $

```
$ echo $PATH
/home/palcain/lib
$ export PATH=$PATH:/home/palcain/proyecto/lib
$ echo $PATH
/home/palcain/bin:/home/palcain/proyecto/lib
```

si editan ~/.bashrc pueden setear con export las variables por defecto para su usuario

Bash: es un lenguaje de programación

Bash Shell Scripting

```
$ cat hello_world
echo 'Hello, world!'
$ bash hello_world
Hello, world!
```

Bash: es un lenguaje de programación

Bash Shell Scripting

```
$ cat hello_name
echo 'Hello, my name is' $1
$ bash hello_world pablo
Hello, my name is pablo
```

Bash: es un lenguaje de programación

Bash Shell Scripting

```
$ cat count
if [ $1 -le 10 ]
then echo $(seq 0 $1)
else echo 'Sorry, I can count upto 10 only'
fi
$ bash count 9
0 1 2 3 4 5 6 7 8 9
$ bash count 31
Sorry, I can count upto 10 only
```

Bash: ejecutando programas

Modificar los permisos con chmod

```
$ ./count
bash: ./count: Permission denied
```

Bash: ejecutando programas

Modificar los permisos con chmod

```
$ ./count
bash: ./count: Permission denied
$ ls -l count
-rw-r--r-- 1 palcain palcain ... hello_world
```

Bash: ejecutando programas

Modificar los permisos con chmod

```
$ ./count
bash: ./count: Permission denied
$ ls -l count
-rw-r--r-- 1 palcain palcain ... hello_world
$ chmod u+x count
-rwxr--r-- 1 palcain palcain ... hello_world
$ ./count 4
0 1 2 3 4
```

Ojo con los directorios! Son siempre “ejecutables”

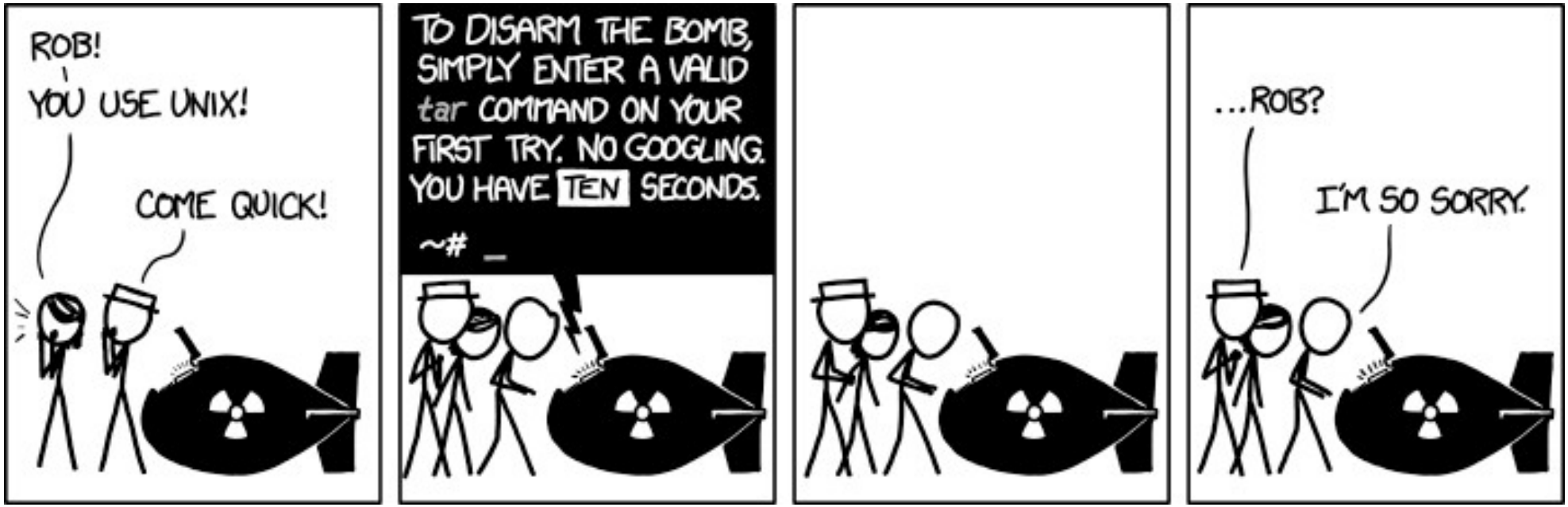
Bash: ejecutando programas

Para ser correctos, el shebang (#!) indica el intérprete

```
$ cat count
#!/bin/bash
if [ $1 -le 10 ]
then echo $(seq 0 $1)
else echo 'Sorry, I can count upto 10 only'
fi
```

Bash: otras herramientas

- ln: crea links (accesos “directos”)
- tar: empaqueta archivos (y comprime)



Makefiles

Herramienta que controla “compilados” desde source files

Mucho más sofisticado que hacer un pequeño script de bash...
y no tan difícil!

Basado en reglas:

```
target:   dependencies ...
          commands
          ...
```

Makefiles

Por ejemplo, compilar hello_world.c a hello_world.e

```
$ cat Makefile
hello_world.e: hello_world.c
    gcc hello_world.c -o hello_world.e
$ make
gcc hello_world.c -o hello_world.e
$ make
make: 'hello_world.e' is up to date.
```

Makefiles

Un ejemplo más real: dos archivos .c en un solo ejecutable

```
$ cat Makefile
programa.e: calculadora.c trig.c
    gcc calculadora.c trig.c -o programa.e
$ make
gcc calculadora.c trig.c -o programa.e
$ make
make: 'programa.e' is up to date.
```

Makefiles

Un ejemplo más real: dos archivos .c en un solo ejecutable

```
$ cat Makefile
programa.e: calculadora.c trig.c
    gcc calculadora.c trig.c -o programa.e
$ make
gcc calculadora.c trig.c -o programa.e
$ make
make: 'programa.e' is up to date.
$ vi trig.c
$ make
gcc calculadora.c trig.c -o programa.e
```

Makefiles: Más que un script

```
$ cat Makefile
programa.e: calculadora.o trig.o
    gcc calculadora.o trig.o -o programa.e

calculadora.o: calculadora.c
    gcc -c calculadora.c

trig.o: trig.c
    gcc -c trig.c

$ make
gcc -c calculadora.c
gcc -c trig.c
gcc calculadora.o trig.o -o programa.e
```

Makefiles: Más que un script

```
$ vi trig.c
$ make
gcc -c trig.c
gcc calculadora.o trig.o -o programa.e
```

Makefiles: variables automáticas

```
$ cat Makefile
programa.e: calculadora.o trig.o
    gcc $^ -o $@

calculadora.o: calculadora.h calculadora.c
    gcc -c $<

trig.o: trig.h trig.c
    gcc -c $<
```

$^: Las dependencias
$@: El target
$<: La última dependencia

Makefiles: reglas implícitas

```
$ cat Makefile
programa.e: calculadora.o trig.o
	gcc $^ -o $@

%.o: %.h %.c
	gcc -c $<
```

$^: Las dependencias
$@: El target
$<: La última dependencia

Makefiles: variables

```
$ cat Makefile
CC = gcc
CFLAGS = -O3
LDFLAGS = -lm
programa.e: calculadora.o trig.o
	$(CC) $^ $(LDFLAGS) -o $@

%.o: %.h %.c
	$(CC) $(CFLAGS) -c $<
```

$^: Las dependencias
$@: El target
$<: La última dependencia

Makefiles: obtener los archivos

```
$ cat Makefile
CC = gcc
CFLAGS = -O3
LDFLAGS = -lm
SRC = $(wildcard *.c)
OBJ = $(patsubst %.c, %.o, $(SRC))
programa.e: $(OBJ)
    $(CC) $^ $(LDFLAGS) -o $@

%.o: %.h %.c
    $(CC) $(CFLAGS) -c $<
```

Makefiles: algunos detalles

```
$ cat Makefile
...
.PHONY: all clean
all: programa.e

programa.e: $(OBJ)
	$(CC) $^ $(LDFLAGS) -o $@

%.o: %.h %.c
	$(CC) $(CFLAGS) -c $<

clean:
	rm -rfv $(OBJ) programa.e
```

no son archivos

la primera regla

clean (por las dudas)

Pablo Alcain
pabloalcain@gmail.com

Herramientas GNU o: cómo aprendí a dejar de preocuparme y amar la línea de comandos