

A SHORT OVERVIEW

---

# USEFUL DEVELOPER TOOLS

## TOPICS

- ▶ File Management (organize code, collaboration, ...)
- ▶ Debugging (identify bugs in the code)
- ▶ Profiling (optimize code for speed, memory and energy usage, ...)

# FILE MANAGEMENT

- ▶ Version Control
- ▶ Make

# VERSION CONTROL

- ▶ Keeps track of code!
- ▶ Collaborate with other people.
- ▶ Documentation.
- ▶ Undo mistakes.
- ▶ Widely used in all types of development.
- ▶ CVS, SVN, BitKeeper, Mercurial, git, ...
- ▶ Usually included in all IDE:s. Many Open Source.

# VERSION CONTROL > GIT

- ▶ Initialise a (local) directory: `git init`
- ▶ Copy from somewhere else: `git clone http://github.io/a-project`
- ▶ See status: `git status`
- ▶ Add to the index: `git add foo.c foo.h`
- ▶ Commit (locally): `git commit -m "added functionality foo"`
- ▶ Old version: `git log`; `git checkout <hash>`; `git checkout master`
- ▶ Push/pull to/from master: `git push`; `git pull`

### MAKE

- ▶ Historically the most common way to create customizable cross-platform compiler options.
- ▶ Source dependencies, partial recompilation, ...
- ▶ `./configure && make && make install`
- ▶ Modern variant is CMake.
- ▶ Makefile usually generated automatically.

target: <dependencies>  
system command

MAKE

DEMONSTRATION

# DEBUGGING

- ▶ Identify bugs!
- ▶ print
- ▶ gdb/ddd/lldb

# DEMONSTRATION

- ▶ Unit testing, log files, Valgrind, ...

## PROFILING

- ▶ Optimizing tools to find bottlenecks, memory leaks, etc.
- ▶ Python profiling `python -m cProfile myscript.py`

```
python3.6 -m cProfile -s cumtime pythondemo.py > mylog.txt
```

81303169 function calls (81297282 primitive calls) in 58.163 seconds						
Ordered by: cumulative time						
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)	
374/1	0.020	0.000	58.164	58.164	{built-in method builtins.exec}	
1	0.005	0.005	58.164	58.164	pythondemo.py:1(<module>)	
4	0.000	0.000	56.517	14.129	eigen_symmetric.py:504(eigsy)	
4	7.920	1.980	47.695	11.924	eigen_symmetric.py:44(r_sy_tridiag)	
6607314	10.437	0.000	15.424	0.000	matrices.py:433(__getitem__)	
3992201	4.555	0.000	11.713	0.000	<string>:2(__mul__)	
4	1.094	0.274	8.818	2.205	eigen_symmetric.py:377(tridiag_eigen)	
2809962	3.137	0.000	8.567	0.000	<string>:2(__add__)	
3880256	3.436	0.000	6.581	0.000	libmpf.py:677(mpf_add)	

## PROFILING

- ▶ C profiling

# DEMONSTRATION