The taming of the Software

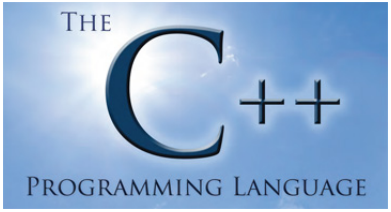
C++, embedded. . . and more

Bartek 'BaSz' Szurgot

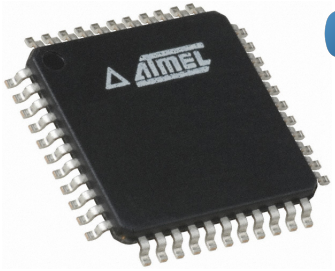
https://www.baszerr.eu

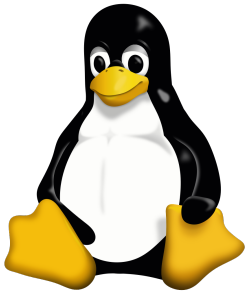
November 14, 2016

Geeking around



#!/bin/bash

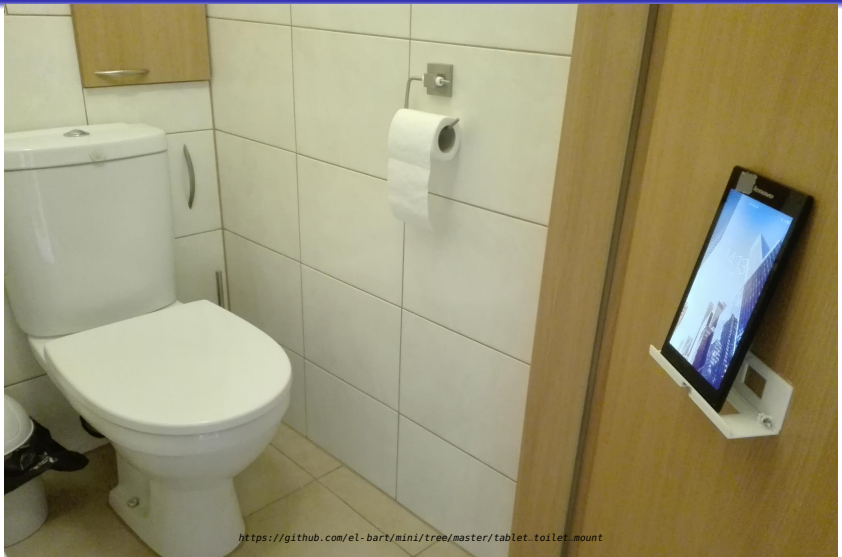




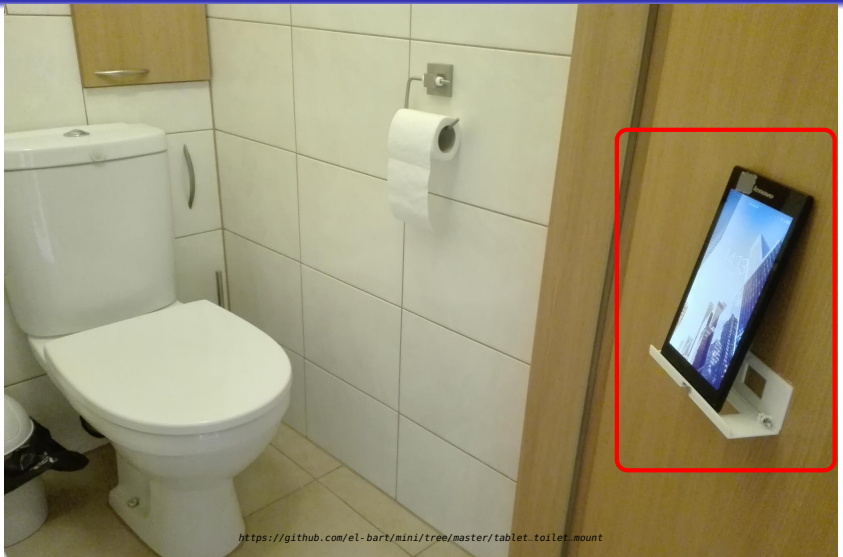
LaTeX

https://isocpp.org/images/uploads/logo-sun.jpg
https://upload.wikimedia.org/wikipedia/commons/3/35/Tux.svg
https://upload.wikimedia.org/wikipedia/commons/9/92/LaTeX_logo.svg
https://upload.wikimedia.org/wikipedia/en/2/22/Heckert_GNU_white.svg
https://upload.wikimedia.org/wikipedia/commons/f/f8/Python_logo_and_wordmark.svg
https://2.bp.blogspot.com/-jt7Tr8ojSYM/UCFiDzLYchI/AAAAAAAAAWk/he6Lx_7q2U0/s1600/Atmel-ATMEGA32U4-AU.jpg

Spare time. . .


https://github.com/el-bart/mini/tree/master/tablet_toilet_mount

Spare time. . .



https://github.com/el-bart/mini/tree/master/tablet_toilet_mount

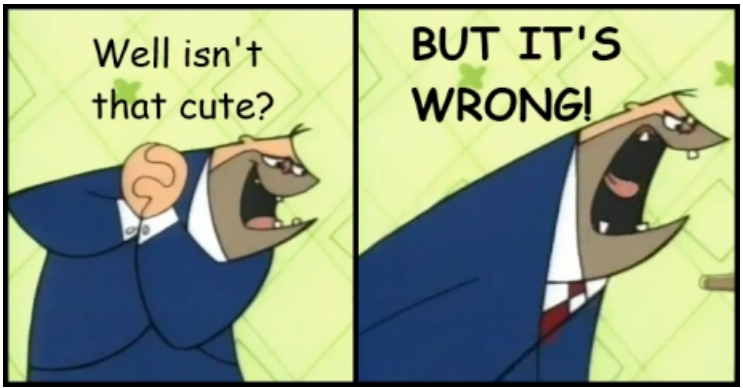
http://fastnbulbous.com/wp-content/uploads/2013/03/ilm-70s-poll.png

What?

Building
Decomposing
Abstracting
Testing

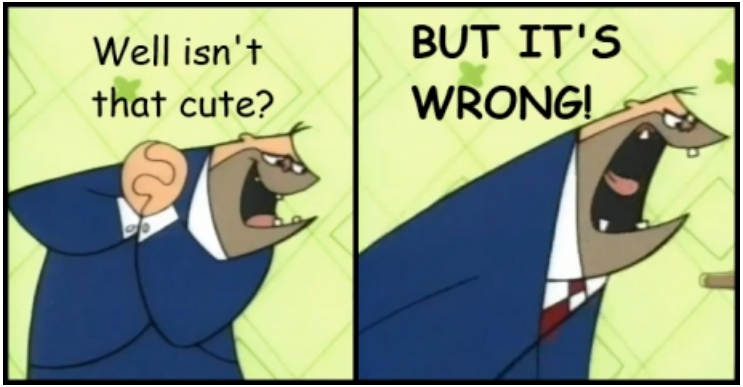
Why?

Why?



https://c2.staticflickr.com/4/3207/2724551187_0a1c71658b.jpg
https://7770647a14b0867efc75-b939f832d8cd9c860ce8909163419528.ssl.cf2.rackcdn.com/401583.jpg

Why?



https://c2.staticflickr.com/4/3207/2724551187_0a1c71658b.jpg
https://7770647a14b0867efc75-b939f832d8cd9c860ce8909163419528.ssl.cf2.rackcdn.com/401583.jpg


4335 km Biegun Północny
Londyn 1183 km
Warszawa
Rzym 1072 km
918 km Sztokholm
Kapsztad 9432 km

Let’s talk about. . .

Builds

For example. . .


COLOR FORECAST STATION
FORECAST
OUTDOOR
HUMIDITY
103.5 °F
46 %
RH
INDOOR
76.8 °F
HUMIDITY
35 %
RH
12:34
FAT
7:20
La Crosse TECHNOLOGY

For example. . .

1. `make clean`
2. `make image`

For example. . .

1. `make clean`
2. `make image`

1. `make clean`
2. `make image`
3. `make test`

http://blogfr.communes.com/wp-content/uploads/2012/01/free_wifi-copie.png
http://www.weathershop.com/images/cellular_weather_station_top_detail_1024.jpg
http://top10bestpro.com/wp-content/uploads/2014/11/La-Crosse-Technology-308-1412S-Color-LCD-Wireless-Weather-Station.jpg

https://i.chzbgr.com/full/6137711616/h4FD4ABB5

http://www.quickmeme.com/img/1e/1ec0667354eae3d8178511551c8959e1e493847 08f9b4ddf5420d39898e8688b.jpg

Build options

Build options

CORE

NICE-TO-HAVE

Build options

CORE

- Type/Mode:
  - Debug
  - Release

NICE-TO-HAVE

Build options

CORE

- Type/Mode:
  - Debug
  - Release
- Set/Target:
  - Production code
  - Tests

NICE-TO-HAVE

Build options

CORE

- Type/Mode:
  - Debug
  - Release
- Set/Target:
  - Production code
  - Tests
- Architecture:
  - ARMv8 ("our HW")
  - AMD64 (!)

NICE-TO-HAVE

Build options

CORE

- Type/Mode:
  - Debug
  - Release
- Set/Target:
  - Production code
  - Tests
- Architecture:
  - ARMv8 ("our HW")
  - AMD64 (!)

NICE-TO-HAVE

- Sanitizers:
  - none
  - address
  - thread
  - undefined
  - leak

Build options

CORE

- Type/Mode:
  - Debug
  - Release
- Set/Target:
  - Production code
  - Tests
- Architecture:
  - ARMv8 ("our HW")
  - AMD64 (!)

NICE-TO-HAVE

- Sanitizers:
  - none
  - address
  - thread
  - undefined
  - leak
- Coverage

Build options

CORE

- Type/Mode:
  - Debug
  - Release
- Set/Target:
  - Production code
  - Tests
- Architecture:
  - ARMv8 ("our HW")
  - AMD64 (!)

NICE-TO-HAVE

- Sanitizers:
  - none
  - address
  - thread
  - undefined
  - leak
- Coverage
- Permit warnings

Rule no.1

Orthogonal build

All build options should be independent of one another.

Rule no.1

Orthogonal build

All build options should be independent of one another.

1. Type (2)
2. Set (2..6)
3. Architecture (2)
4. Sanitizers (5)
5. Coverage (2)
6. Warnings (2)

Rule no.1

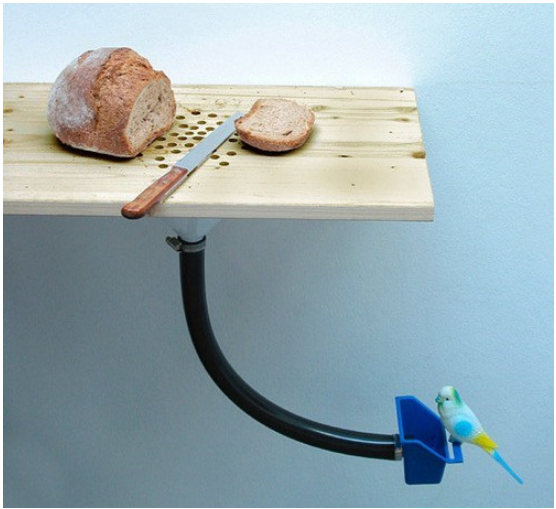
Orthogonal build

All build options should be independent of one another.

1. Type (2)
2. Set (2..6)
3. Architecture (2)
4. Sanitizers (5)
5. Coverage (2)
6. Warnings (2)

http://api.ning.com/files/nII-N*HRVI0ap6EIiZm6hqokLoabPQi2eihY5bGzio7YrYltTcFJyvbRmQgTpjuNby9TlySlQPdjH4IGEUOUUbRVi-XSnTq*/saywhat.png

http://putnamkerr.com/wp-content/uploads/2015/04/do-what-i-want.jpg



https://i.chzbgr.com/full/7948672256/h46ADA3D0
https://i.chzbgr.com/full/6372798464/h82CF94DF

Sooo much stuffff...

```
build_command TYPE=debug SET=production
ARCHITECTURE=mips64 SANITIZER=address
COVERAGE=true PERMIT_WARNINGS=true
```

Sooo much stuffff...

```
build_command TYPE=debug SET=production
ARCHITECTURE=mips64 SANITIZER=address
COVERAGE=true PERMIT_WARNINGS=true
```

http://www.mining.com/wp-content/uploads/2012/05/default-300x250.jpg

Rule no.2

Easy build

Build should be easy to use. Any required complexity must be hidden inside.

Rule no.2

Easy build

Build should be easy to use. Any required complexity must be hidden inside.

- 1-command build (+CMake)

Rule no.2

Easy build

Build should be easy to use. Any required complexity must be hidden inside.

- 1-command build (+CMake)
- Defaults for devs

Rule no.2

Easy build

Build should be easy to use. Any required complexity must be hidden inside.

- 1-command build (+CMake)
- Defaults for devs
- Defaults for local tests?

http://static.techspot.com/images/teaser/lazy-coder.jpg

BTW. . .

- Make?
- CMake + Make?
- . . .

BTW. . .

- Make?
- CMake + Make?
- . . .
- CMake + Ninja!

Ninja

Ninja is a small build system with a focus on speed. It differs from other build systems in two major respects: it is designed to have its input files generated by a higher-level build system, and it is designed to run builds as fast as possible.

Why yet another build system?

Where other build systems are high-level languages Ninja aims to be an assembler.

Ninja build files are human-readable but not especially convenient to write by hand. (See the *generated* build file used to build Ninja itself.) These constrained build files allow Ninja to evaluate incremental builds quickly.

Should you use Ninja?

Ninja's low-level approach makes it perfect for embedding into more featureful build systems; see a list of existing tools. Ninja is used to build Google Chrome, parts of Android, LLVM, and can be used in many other projects due to CMake's Ninja backend.

See the manual for more: philosophical background, whether and how you can use Ninja for your project, platform support, and details about the language semantics.

What's new

The last Ninja release is **v1.7.1**, released 28 Apr 2016. Read the release notes.

Getting Ninja

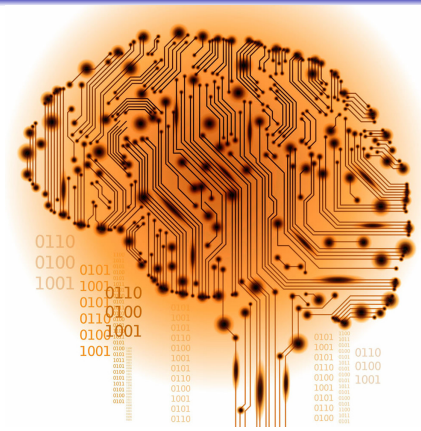
You can download the Ninja binary or find it in your system's package manager.

Or, build from source:

```
$ git clone git://github.com/ninja-build/ninja.git && cd ninja
$ git checkout release
$ cat README
```

Flashing into memory. . .

1. Orthogonal build
2. Easy to use

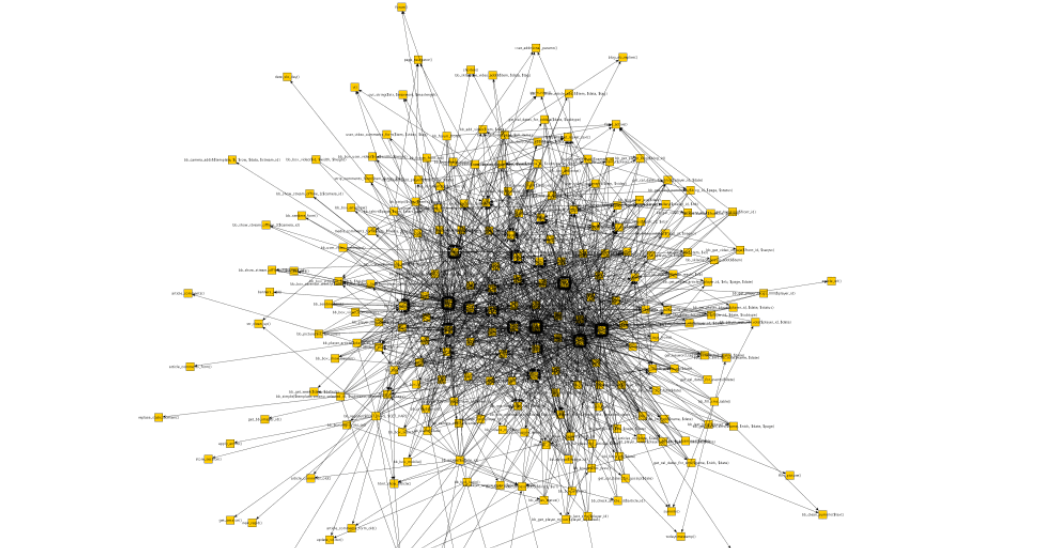


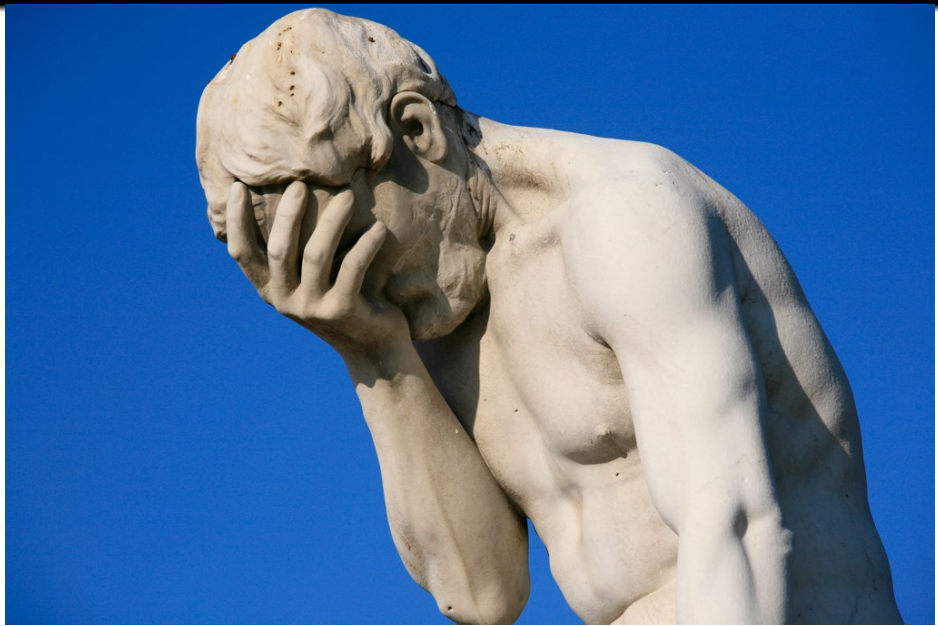
http://novadentmodena.it/wp-content/uploads/2013/01/brain.jpg

Let’s talk about. . .

Components

Looks familiar?





https://upload.wikimedia.org/wikipedia/commons/3/3b/Paris_Tuileries_Garden_Facepalm_statue.jpg

https://i.redd.it/ypa5a3phwljx.jpg

Linking time!

gcc main.c -Wl,–start-group -lYiequ1 -lelu4Th -lz -lz -lz -lnoaF0p -li7 -l9p -lc5 -lg -lg -lg -lN6cG -lUu3E -lu -lu -lu -l8s -loF4J -lh9T -lChoo7 -lr2 -lqui6A -lu1 -l0mF -l87 -lF40 -l1n -lZeis6F -lo -lo -lo -lSaI0 -lt3Mo -l3n -l9f -l49 -l90 -ln3W -laGa8ah -l7OX -leiFa8 -li -li -li -l3M7X -lj -lJ6s -lYPl8 -lfE4 -la -loem0N -lgU7N -lzee6eN -l8u -l61 -lZD7 -lc -loong2U -lsieT6 -le -ljei1Ja -l5i -lkV7 -lJaese1 -lu1X -lEomoh6 -legah5W -lcoGh7 -lx3 -lHoh9m -l7DfA -lkaiCi1 -l6j -l9Sf -lGie5p -ljee2Nu -lGei8wo -lb3 -lxooB6 -laiTu9u -lHah7ai -lEigh8a -lTP1 -l9VnI -l3m -lGar1 -l99W -l8h -lPhaid9 -ly -lVae6ce -lb -lb -lb -l41 -ls3K -lKoox6a -lx -lx -lx -lx -lub5Mue -lFa2pi -lAhch4 -lEeNg4E -lt -lt -lt -l5xT -lAe4Efo -lFV24 -lohP2u -lYo1 -lbO2c -lFDj1 -liV4je -lf8D -lieDie4 -lain4O -lQ7z -lp3X -lX5CS -lw -l5m -l10 -lu4 -ln -ln -lAeph3 -lF74Q -l0j -l0j -l0l -laij5W -lOa6gu -lAm4 -lzoh8P -l6gB -lCq5 -l7k -lOoz0e -l4uK -lXaa4J -lg3 -looB3m -lE7o -lh -lh -lh -l1h -l1h -lGae7i -lJ55B -l6mC -l3Cbm -lEshei4 -loot0X -lOong9z -l3ZYs -lsuNg9 -lOod8Y -l1Xy -lfoo8Y -lIeng8 -ld9 -l5fD -lUe4w -lTTJ1 -l5b -lDaVe5 -lHv6Z -lmai4F -look1B -ld3Yv -lzIp7 -lFu1 -lahm7X -lOhgo1d -lJVr9 -lT3uZ -lwWN5 -l25 -lOcha6 -lpie8B -l7b -lS5VT -lGai4a -l0Oo5 -l6pOC -lR9Sz -lJw3G -l9LWa -l7Id -lEr0Ee -lwD1u -lAiP0qu -ld0F -lz9 -lca4Ahl -lOoth3 -lLaix0 -l3M6 -lDAD0 -l2HW9 -lohKo5 -lJP6 -lOhTas1 -li1 -lZ2E -lvR4 -l11 -lal5Iew -lxM5 -la1 -l91X -lO1e -ljohK2 -lr -lr -lr -l6wU -l4Tw7 -lz0 -lii7Lu -lc3G6 -liBo3ah -lAch9Az -l3FJ -ls -lT1O -lBAc4 -ltu1ooF -loos7Av -lpo2Zoo -l5a -laec6Ah -lc2 -lpa0Ie -luo2Ie -l55 -l0QSH -l7Ad -ln0 -lieBah3 -lPi85 -looH0i -lT8o -l5TJl -luuReu8 -lv5Km -lahM3ah -lyH6 -lquu9Ae -lk -lk -lGan2y -luhie0U -lShi7Z -lyC8 -lahs2I -l91 -leh0Ah -l4Vj -lq -lq -lq -lq -lak7G -lrah5Ae -l1Fe -ly3 -l9tS -ldah4Th -lj7 -loa8aPh -lAhC5U -lchi9Y -lOe1iex -l4O8 -l64 -loq0U -lXaS6ei -l8U4h -l76 -lr6K -lv -lK1M -liWun3 -la2Zq -lm -lm -lm -lBing2k -le2Mf -laij2Du -l6n -lT5ci -lUE17 -lCae3so -Wl,–end-group

Linking O(ops)...

- Problem:
  - –start-group
  - –end-group

Linking O(ops)...

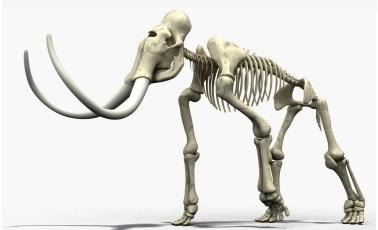
- Problem:
  - –start-group
  - –end-group
- Having:
  - *G* – libraries
  - *M* – dependencies each

Linking O(ops)...

- Problem:
  - –start-group
  - –end-group
- Having:
  - G – libraries
  - M – dependencies each
- $O(MG)$!

Linking O(ops)...

- Problem:
  - –start-group
  - –end-group
- Having:
  - G – libraries
  - M – dependencies each
- $O(MG)$!
- $O(N^2)$!



http://img2-3.timeinc.net/people/i/2015/news/150622/wooly-mammoth-1024.jpg
http://preview.turbosquid.com/Preview/2014/05/12__18_29_02/m1HD.jpg7dcb3c2a-7a59-4377-9d76-ee9ed125598fOriginal.jpg

Rule no.1

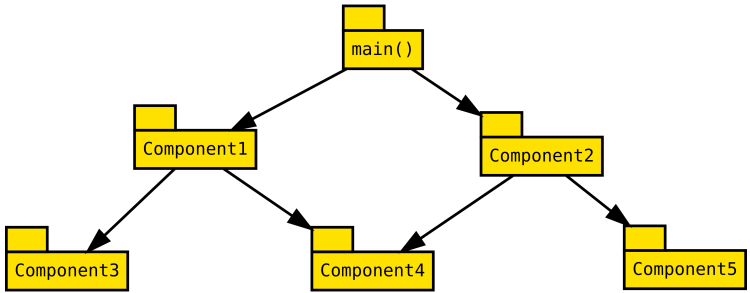
Dependency tree

Dependencies of each component must form a tree.

Rule no.1

Dependency tree

Dependencies of each component must form a tree.



https://2.bp.blogspot.com/-QQ7i_vRV45U/UblUoLIZ-mI/AAAAAAAAAM4/sgo0X8bXR4U/s1600/Temperate+Rainforest.jpg

Where do i find. . .

Where do i find. . .

g++ stuff.cpp -I2WJ/V5h/Z14 -I0xL/3XO/Kj8 -ILu7un/Ke9ah/Ahgh7 -Ik/b/y -IueT7F/bee2D/coo8P -IEiJ5/Ez9p/s46K
-IaiSi6/uw3So/aiVi2 -Iv/o/k -Il/f/u -IAiz8U/cu2We/eed0W -IeiPh6b/IuW1oh/kah6Do -IJ8P/V1I/Z2r
-IUa3se/Dooj7/Ou2Ma -Iri4Ai/oox7C/Ohv7E -Ioom8Ca/nahz2I/Ash7wi -Iy/u/z -Iv/x/n -Ik/g/v -Ii4/9y/0u
-IID0/T7n/H7m -IBH7a/212E/30lF -IX2J/VX5/0qQ -Ib/f/m -I4l/l4/7g -IGov3B/eeH4o/wut3A -I6j/9k/04 -I00/4k/s1
-Ieu7Ch/Aec8u/Veaz0 -Ie/m/j -I1ReK/Hy1G/Wx8G -Iq4lW/Nyc2/P5tF -Iw3/3l/8e -IPhah5f/lahr9B/Pah8li
-ISb7/oW8/sG5 -IBy4d/hLk6/5HuZ -IO3w/vD1/L4u -IW6GV/2P5J/wa9B -I9k/6c/b0 -If4/3o/z0 -Ihi0uM/ash5L/oe4Ni
-Ia/p/b -Ipa7Ahl/xi3Zah/heiz2K -IU1Qi/9uxK/6LZm -I6u/9t/9q -INoh3ae/Uiyoh7/UYoh5p -I6yX/RO2/Q8d
-IK1D/qN6/I3Y -IDooz6m/Cha1uu/thie4O -Ir8vB/yL0N/z6L3 -I1SG4/1lYK/1GB6 -I3k/36/m6 -IT2cl/Zoy5/iRH3 -Il/z/h
-If69L/DmI9/Yi2k -I5z/t7/0n -IyF0/eD0/B0T -Ic/x/v -Ibe6iBu/aim3Ei/Mooch1 -IAhr1e/oPu5f/oht9W -II9B/Q1O/9Sk
-I4lzM/DHp3/8s4F -Ii6T/Bl0/8Pj -IEiTh8w/Ais8we/Bakem5 -I2Bw/M5H/Z04 -I56/h9/0c -ICu6Th/uMux6/aGh8E
-IChae3/EeWi7/Tuew1 -IPhee7/Laip6/af2He -I30/2q/w2 -IEeCh6/Thai8/Mei0O -IJie1x/La0aj/Zex1h
-Imo0Aiv/Huk7ol/PuM0Ph -IuBaa5e/aaGie8/saey1U -I2St/c2C/9kM -Iz5/3y/2m -Iq/n/w -I4t/8o/t0 -IY9G/q0V/Z1T
-IVz2q/ZeV5/ja0Y -Iu/n/a -I6f/6t/m8 -Iw/g/n -Ib/p/o -IKoh7O/Hoqu7/uo4Aw -IDahk0y/eer0Ec/Xie4oo -I82/6w/24
-I30X/Gx7/1ZZ -Ik7/g9/6a -IAiz2xu/iG7hoo/aeTae4 -Iik3Aiy/Cei0Ai/aeY0ka -Im9F/Av1/e0M -IQNg0/5ODG/Las4
-I5g4U/hW2q/K9Bi -IiDiaX4/Oogeg3/aeCa1o -Il/j/w -IWuij3/uB7vu/zo7Or -IoD0gug/Mae0ce/Beiqu6
-Im53V/W6qo/4FOx -IooziB7/ChiiJ9/ohNg7a -I0LMq/5uSw/zQD0 -o stuff.o

Some observations. . .

1. Looooooooong compiler commands. . .

Some observations. . .

1. Looooooooong compiler commands. . .
2. Non-unique file names
   - `Foo/Writer.hpp`
   - `Bar/Writer.hpp`

Some observations. . .

1. Looooooooong compiler commands. . .
2. Non-unique file names
   - `Foo/Writer.hpp`
   - `Bar/Writer.hpp`
3. Order of include-path matters
   - `-IFoo -IBar`
   - `-IBar -IFoo`
   - `#include "Writer.hpp"?!`

Some observations. . .

1. Looooooooong compiler commands. . .
2. Non-unique file names
   - `Foo/Writer.hpp`
   - `Bar/Writer.hpp`
3. Order of include-path matters
   - `-IFoo -IBar`
   - `-IBar -IFoo`
   - `#include "Writer.hpp"?!`
4. Longer file names? ;-)
   - `Foo/Foo_Writer.hpp`
   - `Bar/Bar_Writer.hpp`

https://7373-presscdn-0-43-pagely.netdna-ssl.com/wp-content/uploads/2013/06/binoculars1.jpg

Rule no.2

Namespace segregation

Keep files inside directories named after namespaces they contain. Each file is named after class it contains.

Rule no.2

Namespace segregation

Keep files inside directories named after namespaces they contain. Each file is named after class it contains.

Example:

- Class Foo::Bar::Writer
- Foo/Bar/Writer.hpp
- Foo/Bar/Writer.cpp

Rule no.2

Namespace segregation

Keep files inside directories named after namespaces they contain. Each file is named after class it contains.

Example:

- Class Foo::Bar::Writer
- Foo/Bar/Writer.hpp
- Foo/Bar/Writer.cpp

Properties:

1. Obvious and intuitive
2. Single `-I/my/project`
3. Unique names (filesystem!)

Private includes?

Private includes?

- `boost::detail` convention

Private includes?

- `boost::detail` convention
- Example:
  1. `Foo/Public.hpp`
  2. `Foo/Interface.hpp`

Private includes?

- `boost::detail` convention
- Example:
  1. `Foo/Public.hpp`
  2. `Foo/Interface.hpp`
  3. `Foo/detail/Internall.hpp`
  4. `Foo/detail/DoNotTouch.hpp`

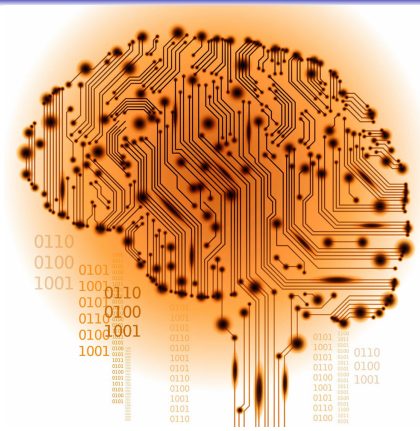
Private includes?

- `boost::detail` convention
- Example:
  1. `Foo/Public.hpp`
  2. `Foo/Interface.hpp`
  3. `Foo/detail/Internall.hpp`
  4. `Foo/detail/DoNotTouch.hpp`
- Support:
  - Caught on review?
  - Ensured via CI?

http://www.mimiz.fr/en/wp-content/uploads/sites/2/2012/02/plan_photo_1347191970.jpg
https://dll4od7sxc8nwf.cloudfront.net/sites/default/files/styles/article_full_width_600_/public/code-review-v1.5.png

Flashing into memory. . .

1. Dependency tree
2. Namespace-to-dir



http://novadentmodena.it/wp-content/uploads/2013/01/brain.jpg

Let’s talk about. . .

HAL

"Portable code"

"Portable code"

```
#ifdef SYSLOG
#ifdef BSD_42
openlog("nntpxfer", LOG_PID);
#else
openlog("nntpxfer", LOG_PID, SYSLOG);
#endif
#endif
#ifdef DBM
if (dbminit(HISTORY_FILE) < 0)
{
#ifdef SYSLOG
  syslog(LOG_ERR,"couldn't open history file: %m");
#else
  perror("nntpxfer: couldn't open history file");
#endif
  exit(1);
}
#endif
#ifdef NDBM
if ((db = dbm_open(HISTORY_FILE, O_RDONLY, 0)) == NULL)
{
#ifdef SYSLOG
  syslog(LOG_ERR,"couldn't open history file: %m");
#else
  perror("nntpxfer: couldn't open history file");
#endif
  exit(1);
}
#endif
if ((server = get_tcp_conn(argv[1],"nntp")) < 0)
{
#ifdef SYSLOG
  syslog(LOG_ERR,"could not open socket: %m");
#else
  perror("nntpxfer: could not open socket");
#endif
  exit(1);
}
if ((rd_fp = fdopen(server,"r")) == (FILE *) 0){
#ifdef SYSLOG
  syslog(LOG_ERR,"could not fdopen socket: %m");
#else
  perror("nntpxfer: could not fdopen socket");
#endif
  exit(1);
}
#ifdef SYSLOG
syslog(LOG_DEBUG,"connected to nntp server at %s", argv[1]);
#endif
#ifdef DEBUG
printf("connected to nntp server at %s\n", argv[1]);
#endif
```

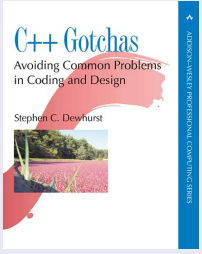
Not quite right. . .

https://41.media.tumblr.com/9b3b27877aa8b832cca7e957d70ab28d/tumblr_n1w3qmLkey1qbioy1o1_500.png

"C++ Gotchas", Stephen Dewhurst

"I have to use `#if` to handle the different platform requirements". To prove your point, such as it is, you display the following code:

```
// some portable code
#ifdef PLATFORM_A
// do one thing
a(); b(); c();
#endif
#ifdef PLATFORM_B
// do other thing
d(); e();
#endif
```



This code is not platform-independent. It's multiplatform dependent. Any change to any of the platforms requires not only a recompilation of the source but change to the source for all platforms. You've achieved maximal coupling among platforms: a remarkable achievement, if somewhat impractical.

Rule no.1

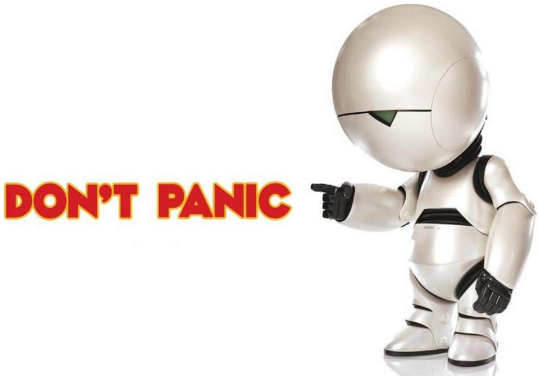
No #ifdefs

When you're using `#ifdefs` – you're doing something wrong.

Rule no.1

No #ifdefs

When you're using `#ifdefs` – you're doing something wrong.

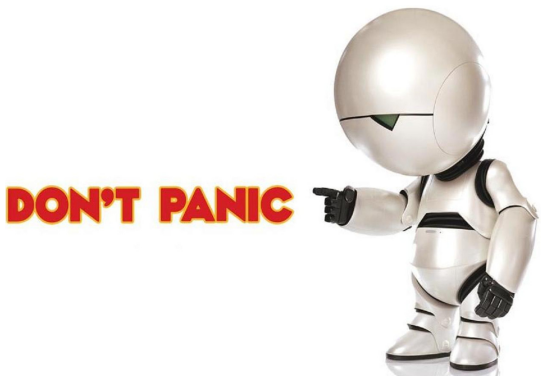


Rule no.1

No #ifdefs

When you're using `#ifdefs` – you're doing something wrong.

- Separate platforms
- Provide abstractions



https://lizistheword.files.wordpress.com/2012/03/dont-panic.jpg

About HAL

About HAL

Hardware **A**bstraction **L**ayer

About HAL

Hardware **A**bstraction **L**ayer

but not:

HAL **A**bstraction **L**ayer

Where to use HAL

Where to use HAL

Good examples

- `void initLcd()`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`
- `uint8_t USART::read()`
- `...`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`
- `uint8_t USART::read()`
- `...`

Bad examples

- `int readStatusRegister()`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`
- `uint8_t USART::read()`
- `...`

Bad examples

- `int readStatusRegister()`
- `bool statusLine15()`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`
- `uint8_t USART::read()`
- `...`

Bad examples

- `int readStatusRegister()`
- `bool statusLine15()`
- `uint8_t readPortA()`
- `...`

Where to use HAL

Good examples

- `void initLcd()`
- `bool backKeyPressed()`
- `uint8_t USART::read()`
- `...`

what?

Bad examples

- `int readStatusRegister()`
- `bool statusLine15()`
- `uint8_t readPortA()`
- `...`

how?

Interface and implementation

```
1 // enableStuff.hpp:
2 void enableStuff();
```

Interface and implementation

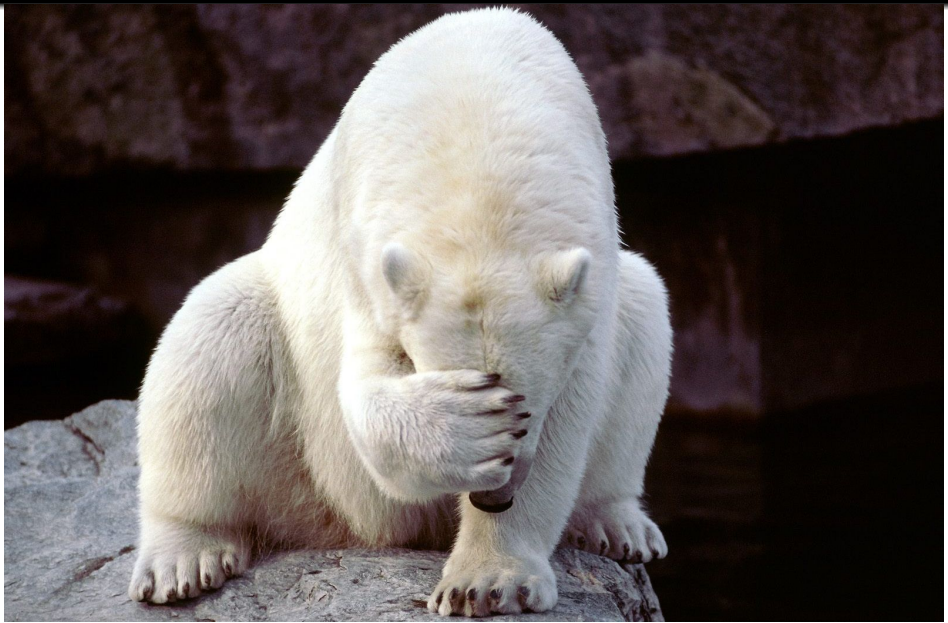
```
1 // enableStuff.hpp:
2 void enableStuff();
```

```
1 // enableStuff.cpp:
2 void enableStuff()
3 {
4 #ifdef PLATFORM_A
5   platformFoo();
6 #else
7   platformBar();
8 #endif
9 }
```

Interface and implementation

```
1 // enableStuff.hpp:
2 void enableStuff();
```

```
1 // enableStuff.cpp:
2 void enableStuff()
3 {
4 #ifdef PLATFORM_A
5   platformFoo();
6 #else
7   platformBar();
8 #endif
9 }
```



http://wallpoper.com/images/00/42/07/97/funny-facepalm_00420797.jpg

Split-backend

```
1 // enableStuff_foo.cpp:
2 void enableStuff()
3 {
4   platformFoo();
5 }
```

Split-backend

```
// enableStuff_foo.cpp:
void enableStuff()
{
  platformFoo();
}
```

```
// enableStuff_bar.cpp:
void enableStuff()
{
  platformBar();
}
```

Split-backend

```
// enableStuff_foo.cpp:
void enableStuff()
{
  platformFoo();
}
```

```
SRCS+=impl_${PLATFORM}.cpp
```

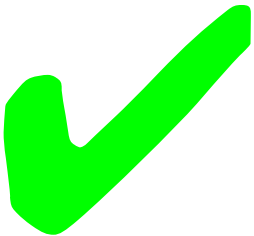
```
// enableStuff_bar.cpp:
void enableStuff()
{
  platformBar();
}
```

Split-backend

```
// enableStuff_foo.cpp:
void enableStuff()
{
  platformFoo();
}
```

```
// enableStuff_bar.cpp:
void enableStuff()
{
  platformBar();
}
```

```
SRCS+=impl_${PLATFORM}.cpp
```



Wait a minute. . .

Wait a minute. . .

http://www.clker.com/cliparts/m/g/n/S/n/o/footprint-sign-hi.png

- Q: "What about a footprint?"

Wait a minute. . .

http://www.clker.com/cliparts/m/g/n/S/n/o/footprint-sign-hi.png

- Q: "What about a footprint?"
- A: "Have you measured?"

Wait a minute. . .

http://www.clker.com/cliparts/m/g/n/S/n/o/footprint-sign-hi.png

- Q: "What about a footprint?"
- A: "Have you measured?"

1. Good enough?

Wait a minute. . .

http://www.clker.com/cliparts/m/g/n/S/n/o/footprint-sign-hi.png

- Q: "What about a footprint?"
- A: "Have you measured?"

1. Good enough?
2. LTO?

Wait a minute. . .

http://www.clker.com/cliparts/m/g/n/S/n/o/footprint-sign-hi.png

- Q: "What about a footprint?"
- A: "Have you measured?"

1. Good enough?
2. LTO?
3. Split-header?

Split-header

```
1 // foo/enableStuff.hpp:
2 inline void enableStuff()
3 {
4   platformFoo();
5 }
```

Split-header

```
// foo/enableStuff.hpp:
inline void enableStuff()
{
  platformFoo();
}
```

```
// bar/enableStuff.hpp:
inline void enableStuff()
{
  platformBar();
}
```

Split-header

```
// foo/enableStuff.hpp:
inline void enableStuff()
{
  platformFoo();
}
```

```
FLAGS+=-I${PLATFORM}/
```

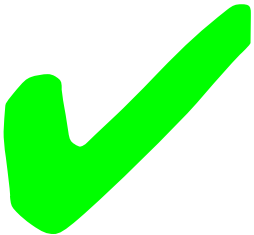
```
// bar/enableStuff.hpp:
inline void enableStuff()
{
  platformBar();
}
```

Split-header

```
// foo/enableStuff.hpp:
inline void enableStuff()
{
  platformFoo();
}
```

```
// bar/enableStuff.hpp:
inline void enableStuff()
{
  platformBar();
}
```

```
FLAGS+=-I${PLATFORM}/
```



Rule no.2

Split-backend

Separate platform-dependent implementations with split-bakend, split-header or both.

http://media2.bazaarvoice.com/images/backend-icon-215.png

http://www.rawstory.com/wp-content/uploads/2013/05/happypolarbear-shutterstock.jpg

New platform?

New platform?

```
find repo/ -name '*_amd64.cpp'
```

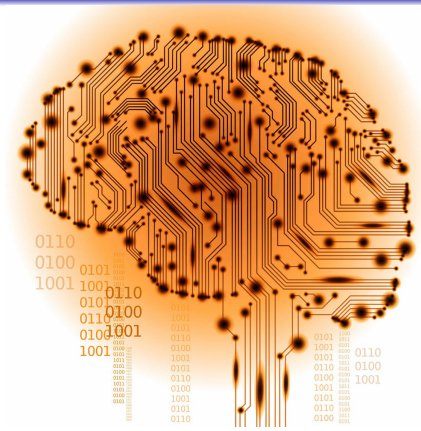
Future and C++’s modules!

Future and C++'s modules!

http://i.telegraph.co.uk/multimedia/archive/02327/dancing-polar-bear_2327218k.jpg

Flashing into memory. . .

1. No #ifdefs!
2. Split-backend/header



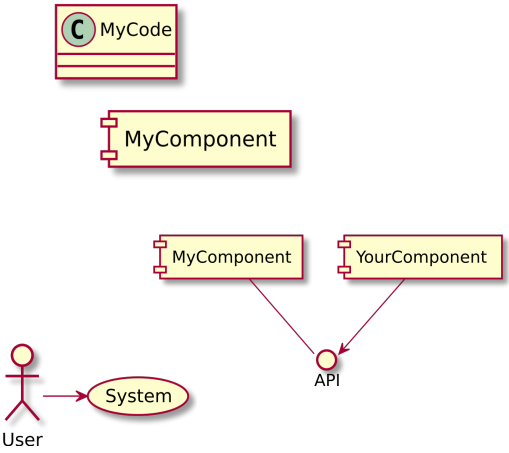
http://novadentmodena.it/wp-content/uploads/2013/01/brain.jpg

Let’s talk about. . .

Testing

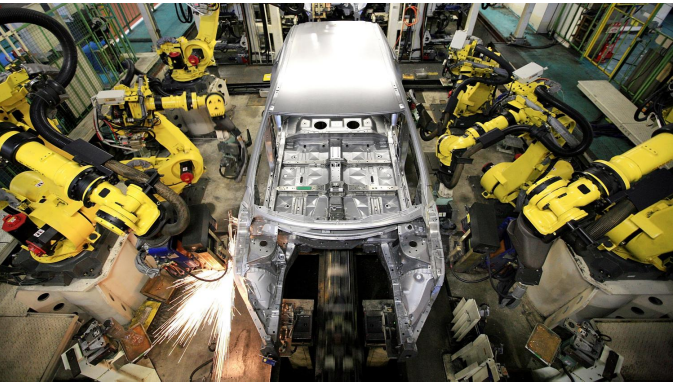
Testing levels

- UTs (Unit Tests)
- MTs (Module Tests)
- ITs (Integration Tests)
- STs (System Tests)

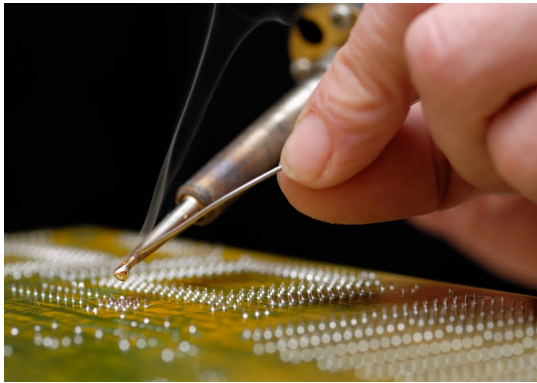


Testing style

AUTOMATED



MANUAL



http://sentryair.com/blog/wp-content/uploads/2010/06/Soldering-Electronics.jpg
https://i.kinja-img.com/gawker-media/image/upload/s--mqcMlzs4--/18toe81gwx0dpjpg.jpg

Testing tradeoffs

Testing tradeoffs

AUTOMATED

+ Repeatable
+ Fast
- Framework setup

Testing tradeoffs

AUTOMATED

+ Repeatable
+ Fast
- Framework setup

MANUAL

- Error-prone
- Slow
+ Minimal setup

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver – manual

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver – manual
- CI integration

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver – manual
- CI integration – automate

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver – manual
- CI integration – automate
- DMA access to LCD buffer

Suggestion no.1

Best-effort

Use automated and manual tests in a complementary fashion, to speed up development.

Some examples:

bias warning! ;)

- Own container implementation – automate
- Serial port driver – manual
- CI integration – automate
- DMA access to LCD buffer – manual

https://image.freepik.com/darmowe-ikony/przek%C5%82adnie-mechaniczne_318-80433.jpg

Suggestion no.2

HAL boundary

HAL is usually a good layer for automated/manual tests split.

Suggestion no.2

HAL boundary

HAL is usually a good layer for automated/manual tests split.

But note:

Suggestion no.2

HAL boundary

HAL is usually a good layer for automated/manual tests split.

But note:

- Automated hardware tests – sure!

Suggestion no.2

HAL boundary

HAL is usually a good layer for automated/manual tests split.

But note:

- Automated hardware tests – sure!
- Hardware emulation – if doable

Suggestion no.2

HAL boundary

HAL is usually a good layer for automated/manual tests split.

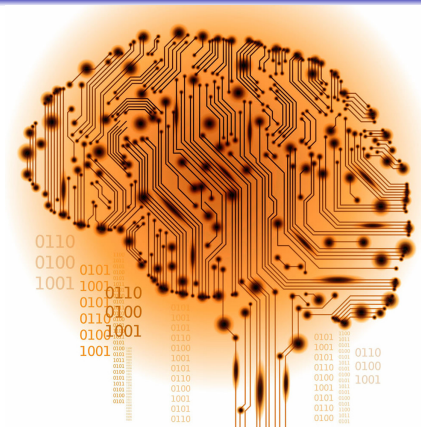
But note:

- Automated hardware tests – sure!
- Hardware emulation – if doable
- Manual tests – implement and version!

http://www.reprap.org/mediawiki/images/thumb/c/c9/STB_Board_1.png/800px-STB_Board_1.png

Flashing into memory. . .

1. Best-effort
2. HAL-boundary

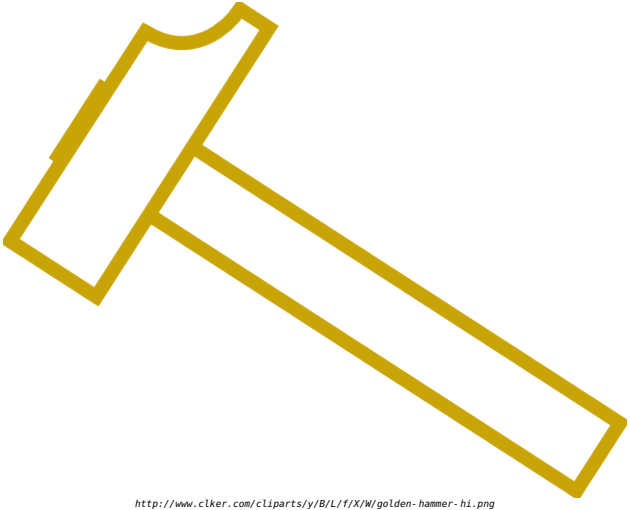


http://novadentmodena.it/wp-content/uploads/2013/01/brain.jpg

Let’s talk about. . .

~Presentation()

No – not really. . .



http://www.clker.com/cliparts/y/B/L/f/X/W/golden-hammer-hi.png

Summary

- Builds:
  1. Orthogonal
  2. Trivial in use

Summary

- Builds:
  1. Orthogonal
  2. Trivial in use
- Components:
  1. Tree-structure
  2. Namespace-to-dir

Summary

- Builds:
  1. Orthogonal
  2. Trivial in use
- Components:
  1. Tree-structure
  2. Namespace-to-dir
- HAL:
  1. No `#ifdefs`!
  2. Split-backend/header

Summary

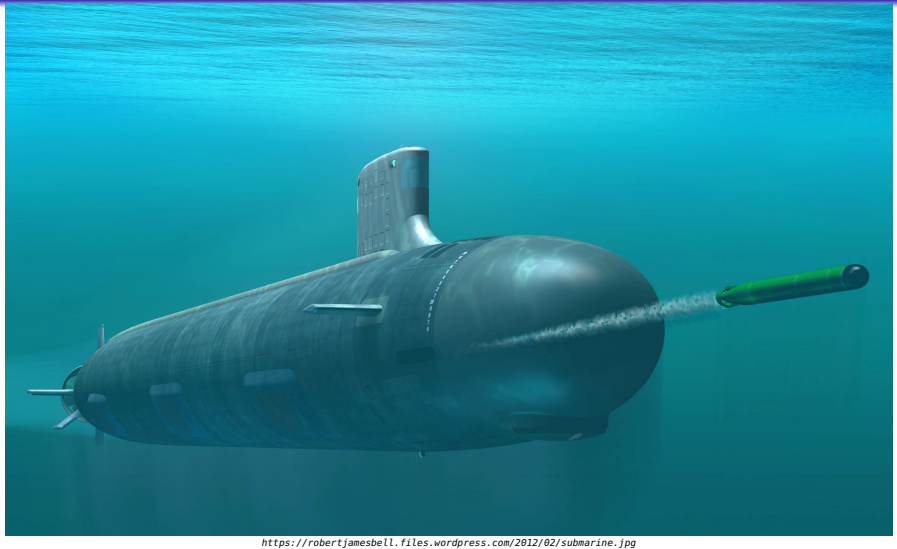
- Builds:
  1. Orthogonal
  2. Trivial in use
- Components:
  1. Tree-structure
  2. Namespace-to-dir
- HAL:
  1. No `#ifdefs`!
  2. Split-backend/header
- Testing:
  1. Best-effort
  2. HAL-split

Some reading

- C++:
  - "*C++ Gotchas: Avoiding Common Problems in Coding and Design*", Stephen C. Dewhurst
  - "*#ifdef Considered Harmful, or Portability Experience With C News*", H. Spencer, G. Collyer
  - "*N4047: A Modules System for C++*", G Reis, M. Hall, G. Nishanov
- Builds:
  - "*Mastering CMake: A Cross-Platform Build System*", K. Martin, B. Hoffman
  - `https://cmake.org`
  - `https://ninja-build.org`
  - `https://en.wikipedia.org/wiki/List_of_build_automation_software`
- Compilation and linking:
  - `https://en.wikipedia.org/wiki/Link-time_optimization`
  - `https://gcc.gnu.org/onlinedocs/gccint/LTO.html`
  - `https://en.wikipedia.org/wiki/Gold_(linker)`

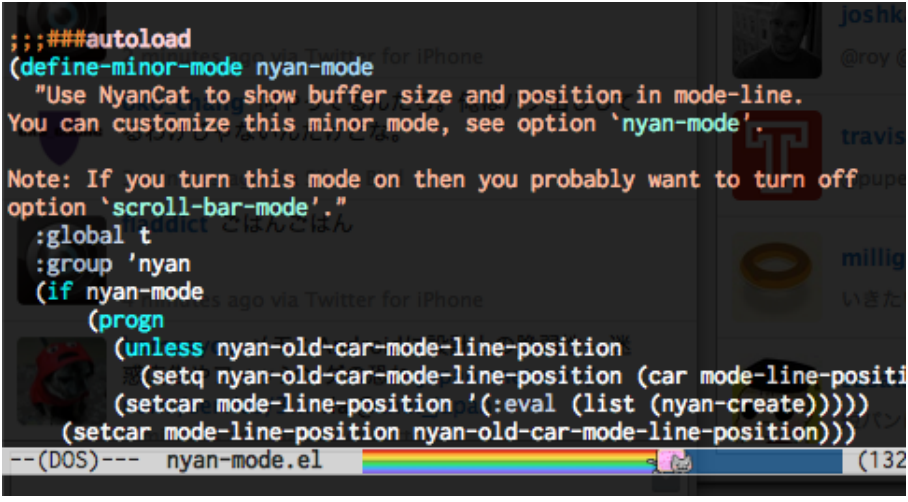
Q&A

Q&A



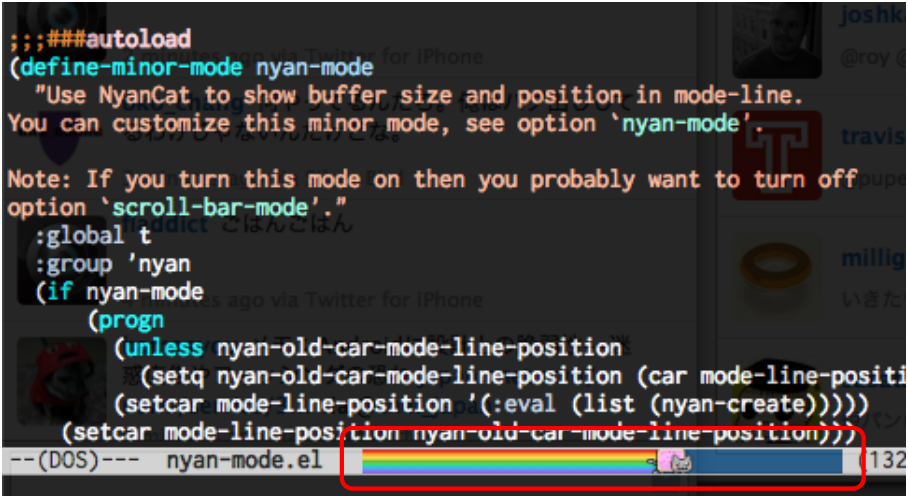
https://robertjamesbell.files.wordpress.com/2012/02/submarine.jpg

VIM vs. EMACS – continued. . .



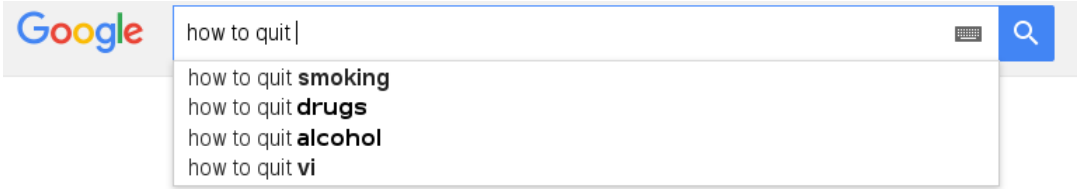
http://temporal.pr0.pl/devblog/download/posts/nyan/x2_7e06acc.png

VIM vs. EMACS – continued. . .

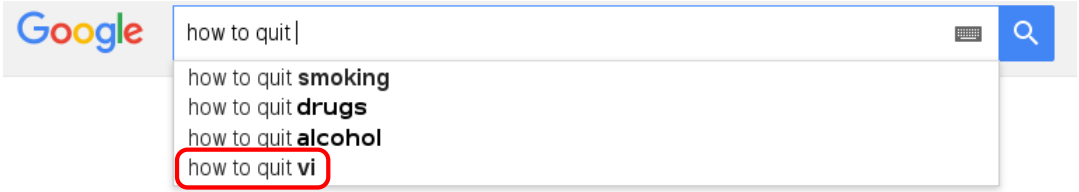


http://temporal.pr0.pl/devblog/download/posts/nyan/x2_7e06acc.png

VIM’s addictive!



VIM’s addictive!



OK – the real version ;-)