Unit testing library: Catch2

Victor Eijkhout, Susan Lindsey

Fall 2023
`last formatted: February 6, 2024`

1. Don't reinvent the wheel: use a library

Many things you want to program have been thought of before:
see if there is a library for it.

Library: 'program without main':
you supply the main, functionality comes from library

2. External libraries: usage

Suppose the ‘fancy’ library does what you need.

1. Include a header file
2. Then use the functions defined there.

```
#include "fancylib.h"

int main() {
  x = fancyfunction(y);
}
```

3. External libraries: compile

1. Compiler needs to know where the header is:

   ```
   icpc -c yourprogram.cpp -I/usr/include/fancylib
   ```

2. You may need to link a library file:

   ```
   icpc -o yourprogram yourprogram.o \
       -L/usr/lib/fancylib -lfancy
   ```

   (not for 'header only' libraries)

4. Libraries with CMake

Use ‘package config’ to find the library,
then use variables with include and library paths/names

```
find_package( PkgConfig REQUIRED )
pkg_check_modules( OPTS REQUIRED cxxopts )
target_include_directories(
  ${PROGRAM_NAME} PUBLIC
  ${OPTS_INCLUDE_DIRS}
)
```

5. Where to find libraries

Search ...
There is a lot of stuff on github.

6. Example: catch2

Clone development version
(better than the 2.x releases)

```
git clone https://github.com/catchorg/Catch2.git catch-git
```

Build:

```
mkdir build
cd build
cmake -D CMAKE_INSTALL_PREFIX=../installation \
    ../catch-git
make
make install
```

7. Usage

See the TDD slides.