Software libraries: cxxopts

Victor Eijkhout, Susan Lindsey

Fall 2024

last formatted: November 20, 2024

Commandline arguments

1. Traditional commandline parsing

Use:

```
int main( int argc,char **argv ) { // stuff };
```

then

Code:

```
1 // args/argcv.cpp
2 cout << "Program name: "
3      << argv[0] << '\n';
4 for (int iarg=1; iarg<argc;
       ++iarg)
5   cout << "arg: " << iarg
6        << argv[iarg] << " => "
7        << atoi( argv[iarg] ) <<
       '\n';
```

Output:

```
./argcv 5 12
Program name: ./argcv
arg 1: 5 => 5
arg 2: 12 => 12
./argcv abc 3.14 foo
Program name: ./argcv
arg 1: abc => 0
arg 2: 3.14 => 3
arg 3: foo => 0
```

Major headache dealing with general arguments:

2. Example: `cxxopts`

`https://github.com/jarro2783/cxxopts`

Find the `3.1.1` release or newer.
Use `wget` or `curl` to download straight to the class machine.

```
wget https://github.com/jarro2783/cxxopts/archive/refs/tags/v3.0.0.tar.
```

Unpack it:

```
tar fxv v3.0.0.tar.gz
```

3. Cmake based installation

- Download from: `https://github.com/jarro2783/cxxopts`
- CMake installation as usual
- Found through *pkg-config*;
- add *mylibs/cxxopts/lib/pkgconfig* to *PKG_CONFIG_PATH*

4. CMake discovery

Header-only, so only set include directory:

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

5. Let's use this library

```
#include "cxxopts.hpp"
```

```
1 // args/cxxopts.cpp
2 // in the main program:
3 cxxopts::Options options
4   ("cxxopts",
5    "Commandline options demo");
```

Compile

```
icpc -o program source.cpp \
   -I/path/to/cxxopts/installdir/include
```

Can you compile and run this?

6. Help option

You want your program to document its own usage:

```
1 // args/cxxopts.cpp
2 options.add_options()
3   ("h,help","usage information")
4   ;
5   /* ... */
6 auto result = options.parse(argc, argv);
7 if (result.count("help")>0) {
8   cout << options.help() << '\n';
9   return 0;
10 }
```

Use:

```
./myprogram -h
```

7. Numerical options

```
1 // args/cxxopts.cpp
2 // define `-n 567' option:
3 options.add_options()
4   ("n,ntimes","number of times",
5    cxxopts::value<int>()
6    ->default_value("37")
7    )
8   ;
9   /* ... */
10 // read out `-n' option and use:
11 auto number_of_times = result["ntimes"].as<int>();
12 cout << "Using number of times: " << number_of_times << '\n';
```

8. Array options

```
1 // args/cxxopts.cpp
2 //define `-a 1,2,5,7' option:
3 options.add_options()
4   ("a,array","array of values",
5    cxxopts::value< vector<int> >()->default_value("1,2,3")
6    )
7   ;
8   /* ... */
9 auto array = result["array"].as<vector<int>>();
10 cout << "Array: " ;
11 for ( auto a : array ) cout << a << ", ";
12 cout << '\n';
```

9. Positional arguments

```
1 // args/cxxopts.cpp
2 // define `positional argument' option:
3 options.add_options()
4   ("keyword","whatever keyword",
5    cxxopts::value<string>())
6   ;
7 options.parse_positional({"keyword"});
8   /* ... */
9 // read out keyword option and use:
10 auto keyword = result["keyword"].as<string>();
11 cout << "Found keyword: " << keyword << '\n';
```

10. Put it all to the test

Now make your program do something with the inputs:

```
./myprogram -n 10 whatever
```