boost
C++ LIBRARIES

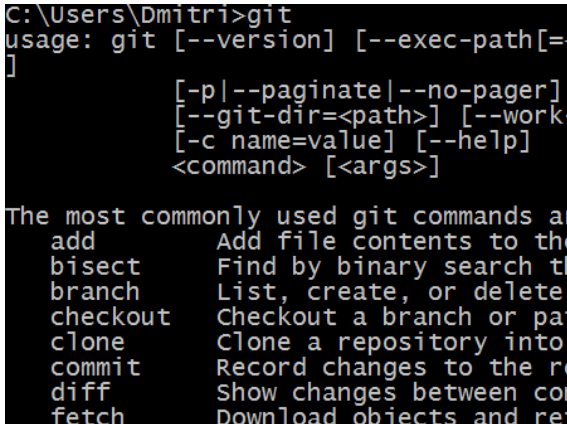
Boost.ProgramOptions

Dmitri Nesteruk
http://activemesa.com
dn@activemesa.com

pluralsight
hardcore developer training

Overview

- `main(int argc, char *argv[])`
- `git commit –m "some changes"`
  - `argc = 4`
  - `argv[0] = c:\git\git.exe`
  - `argv[1] = commit`
  - `argv[2] = -m`
  - `argv[3] = some changes` *quotes aren't included*
- **Arguments can be required or optional**
  - With or without default values
- **Arguments can have longer and shorter form**
  - `git commit -a` *or* `--all`
- **Arguments must be self-documenting**
- **Arguments can be external (e.g. in config file or environment vars)**
- **Boost.ProgramOptions covers these needs!**



Fundamentals

- **`options_description`**
  - Root container
  - All allowed options declared here
  - Writing to a stream
    `cout << desc << endl;`
    prints documentation
- **`od.add_options()`**
  `("help", "print help message")`
  Fluent interface for defining options. Can customize
  - Default values
  - Required/optional
  - Prefix (default `--`) allows delimiting
    `foo.exe --src a.txt b.txt c.txt`
- **`variables_map`**
  - Stores parsed variables
  - `vm.count("foo")` counts # of "foo" arguments (args may repeat)
  - `vm["foo"]` gets the value of the argument
- **Namespace-global functions proxy from `argc/argv` to `variables_map`**
  - `store()` used to get a parser to parse args and store in map
  - `notify()` must be called after parsing – needed in advanced scenarios

Customization

- **Default value**
  - `po::value<string>()->default_value("file.txt")`
- **Binding to variable**
  - `int n;`
    `po::value<int>(&n)`
- **Multiple aliases**
  - `("filename,F", po::value<string>(), …)`
  - Lets you pass argument with `--filename` or `-F`
- **Multiple option arguments**
  - `("files", po::value<vector<string>>())`
  - `foo.exe --files a.txt b.txt`
- **Positional arguments**
  - Command-line tokens which have *no* option name, i.e.
  - Treat
    `foo.exe myfile.txt`
    as
    `foo.exe --filename myfile.txt`
- `po::positional_options_description`
- `p.add("filename", -1);`
- **Explicitly create a `command_line_parser` and initialize it with positional options**
  - `.positional(p).run()`

Summary

- `#include <boost/program_options.hpp>`
  `namespace po = boost::program_options;`
- Describe all options in `options_description`
- Positional options go into `positional_options_description`
- Create a `variables_map` to store the options
- Create a parser, call `store()` and then `notify()` to assign
- Check if option is provided with `vm.count("foo")`
- Get the option with `vm["foo"]` or bind to variable