C++ Korea 4th Seminar

C++ 프로젝트 ~처음 만난 세계~

처음 만난 Modern C++의 세계

C++ Standard의 위엄

SK telecom
장정철 rickyjang@gmail.com

Table of Contents

1. Introduce

SK telecom Car life 서비스 개발 Cell 소속의 잡부(Manager)

T map Route Plan(Machine)

Linux Mandrake A STAR IS BORN

GTK

CUBRID

DARK EDEN EXPANSION

마계촌 ONLINE

KINGDOM UNDER FIRE II

1. Introduce

Windows

Linux

VS

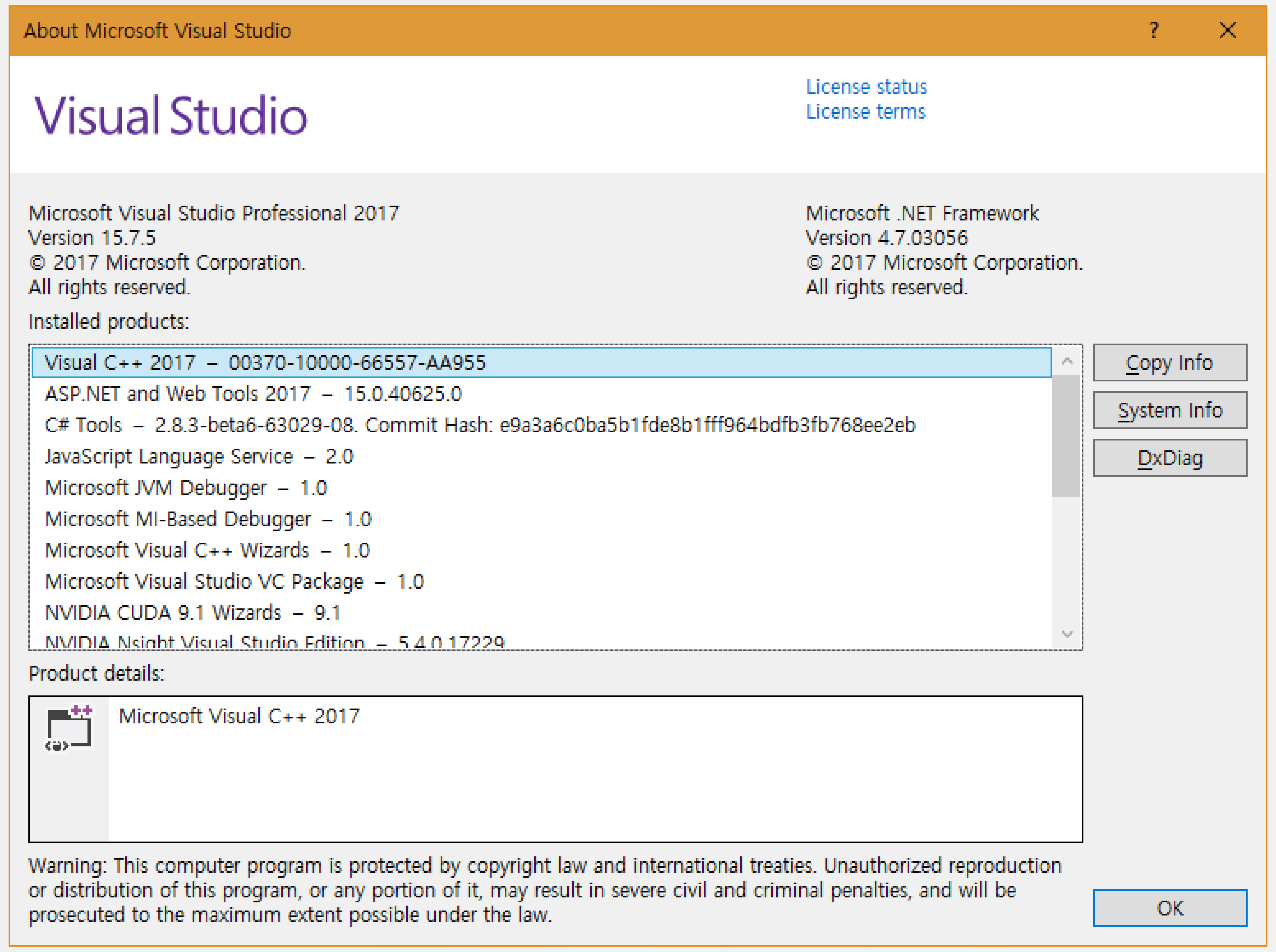
2. Background

새로운 실시간 교통정보의 유효성 검증

1일 평균 1억 건 이상의 길안내 요청.
하나의 Trip 으로 완성되는 비율은 ○ ○%.
그 완성된 Trip 중 10만개를 Sampling 하여 새로운 교통정보를 검증
10만 Trip은 각 시작, 종료 요청을 가지고 있으므로 20만 요청 기록
하나의 요청당 평균 100Kb 정도, 전체 Load 시 2Gb 가량 소모.
20만 요청의 기록은 전체 ○ ○ ○ 대의 서버에 분산 저장되어 있음.
FTP로 서버로 부터 기록을 가져와서 분석하고 정렬한 뒤 질의한 결과를 저장함.
교통정보는 5분단위로 만들어짐. 하루 총 288번의 교통정보로 나뉨.
20만개의 정보가 288 그룹으로 정렬되어야 하고, 가장 빠른 시간부터 순차적으로 질의.
서버에 질의 전에 해당 서버의 교통정보 시간을 조정.
서버와의 통신 Protocol은 HTTP를 사용함.
이 툴을 만들면서 사용된 Modern C++ Feature들을 가지고 이야기를 하려고 함.

3. Environment - Develop

Windows 10



Microsoft Visual Studio Professional 2017
Version 15.7.5
© 2017 Microsoft Corporation.
All rights reserved.

Microsoft .NET Framework
Version 4.7.03056
© 2017 Microsoft Corporation.
All rights reserved.

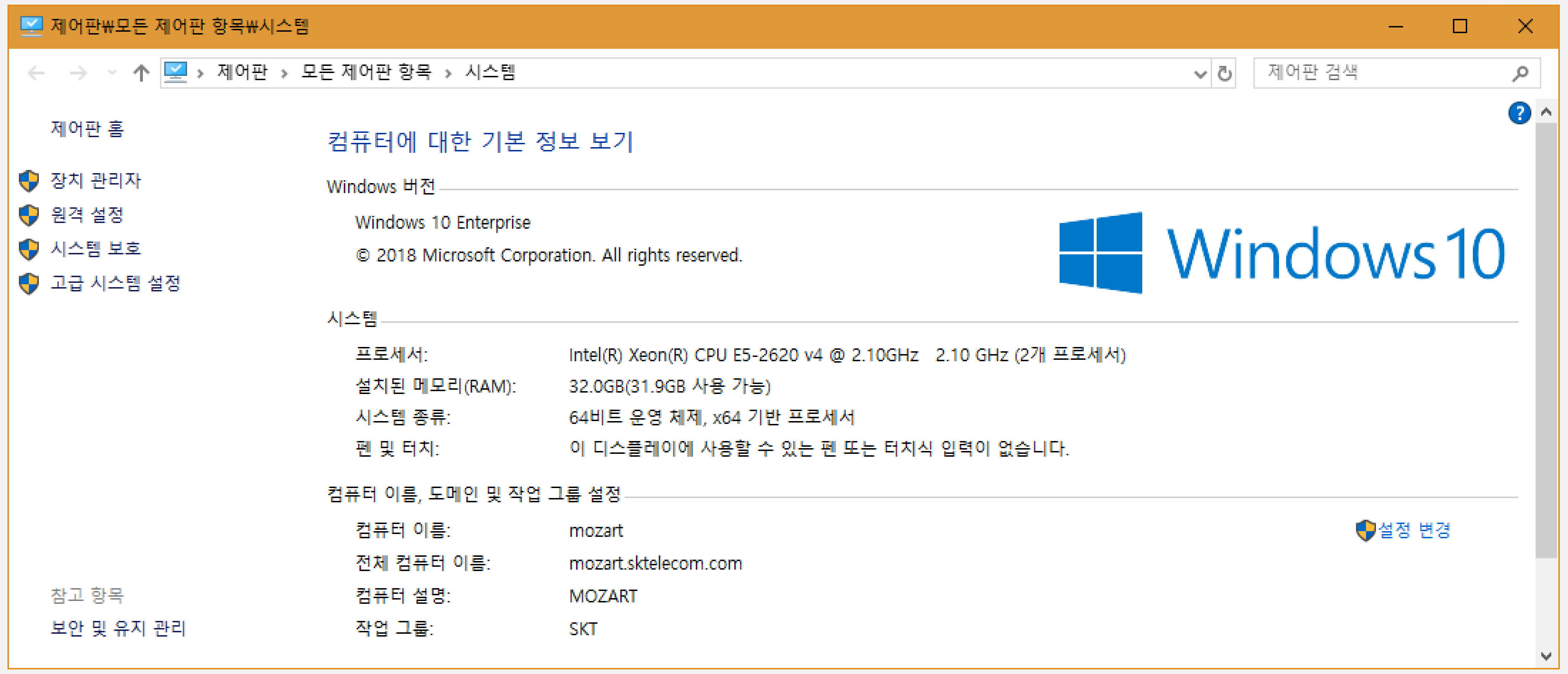
Installed products:

```
Visual C++ 2017 – 00370-10000-66557-AA955
ASP.NET and Web Tools 2017 – 15.0.40625.0
C# Tools – 2.8.3-beta6-63029-08. Commit Hash: e9a3a6c0ba5b1fde8b1fff964bdfb3fb768ee2eb
JavaScript Language Service – 2.0
Microsoft JVM Debugger – 1.0
Microsoft MI-Based Debugger – 1.0
Microsoft Visual C++ Wizards – 1.0
Microsoft Visual Studio VC Package – 1.0
NVIDIA CUDA 9.1 Wizards – 9.1
NVIDIA Nsight Visual Studio Edition – 5.4.0.17229
```

Product details:

Microsoft Visual C++ 2017

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.



컴퓨터에 대한 기본 정보 보기

Windows 버전

Windows 10 Enterprise
© 2018 Microsoft Corporation. All rights reserved.

시스템

프로세서:	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz 2.10 GHz (2개 프로세서)
설치된 메모리(RAM):	32.0GB(31.9GB 사용 가능)
시스템 종류:	64비트 운영 체제, x64 기반 프로세서
펜 및 터치:	이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

컴퓨터 이름, 도메인 및 작업 그룹 설정

컴퓨터 이름:	mozart
전체 컴퓨터 이름:	mozart.sktelecom.com
컴퓨터 설명:	MOZART
작업 그룹:	SKT

Windows Subsystem for Linux

3. Environment - Production

CentOS

redhat
LINUX

4. Pre-requirement – Windows 10

Windows 10

Windows Subsystem for Linux를 사용하기 위해서

Windows OS가 설치된 파티션에 2GB 이상의 여유공간

4. Pre-requirement –MSVC 2017

Visual Studio 2017

Visual Studio Community Edition

1. 개인은 무료
2. 250대 이하의 PC를 사용하는 사업장

   연매출 100만 달러 이하의 사업장에서는 5 대까지 무료
3. Version 15.7.5 기준 C++ 17 표준 완벽 지원

Windows Subsystem for Linux

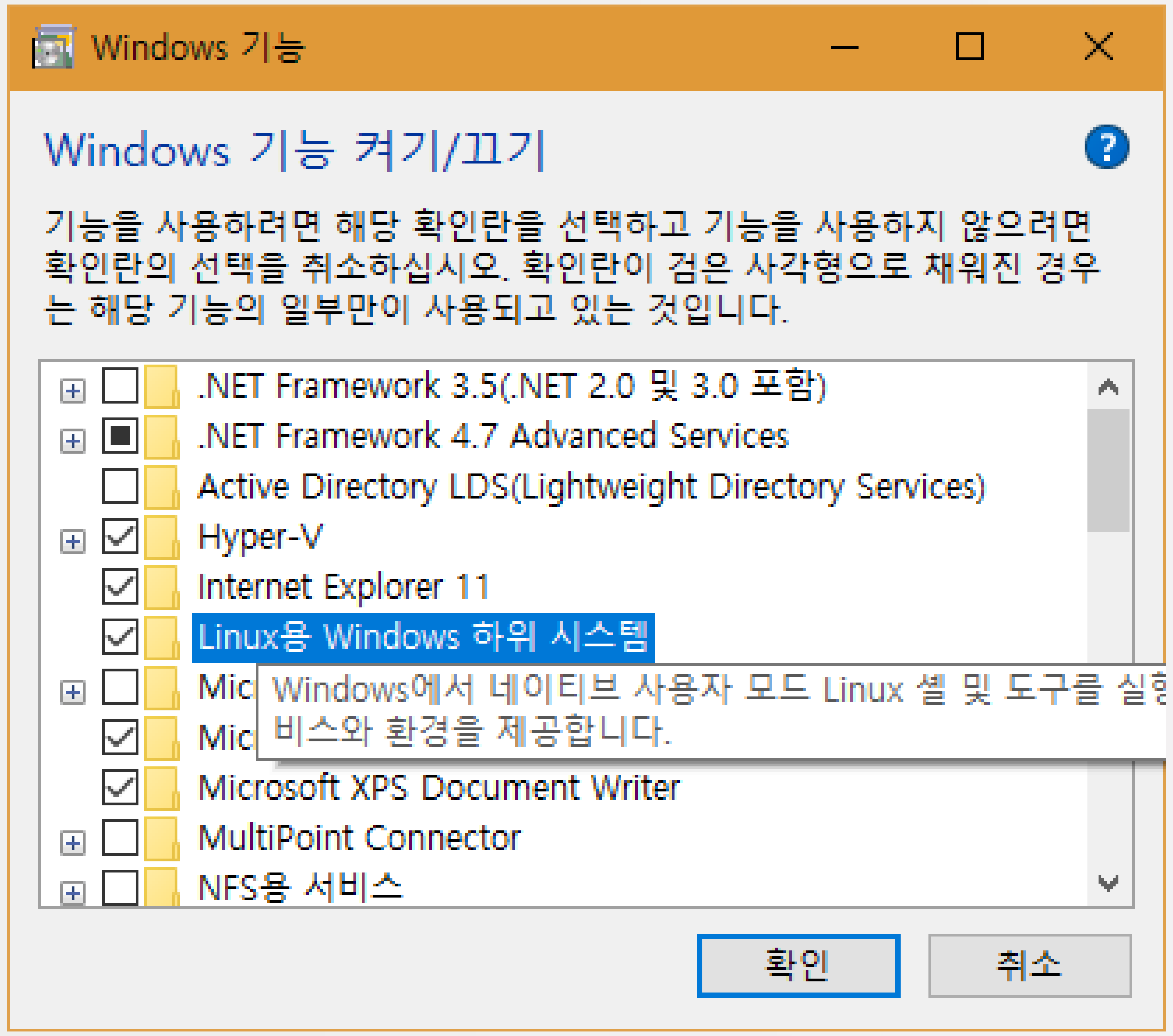
WSL은 ELF64 바이너리에 대하여, 바이너리수준의 호환성을 보장하는 리눅스 실행 환경입니다.

WSL로 할 수 있는 것

1. **C/C++**, Node.JS, Ruby, Python 등의 개발언어 사용
2. FIND, GREP, SED, AWK 등의 유틸리티 사용
3. VIM, EMACS, TMUX 등의 도구 사용
4. 리눅스 콘솔에서Windows 프로그램 실행
5. Windows 프로그램이 리눅스 프로그램 실행

4. Pre-requirement – WSL

Windows Subsystem for Linux



Windows 기능

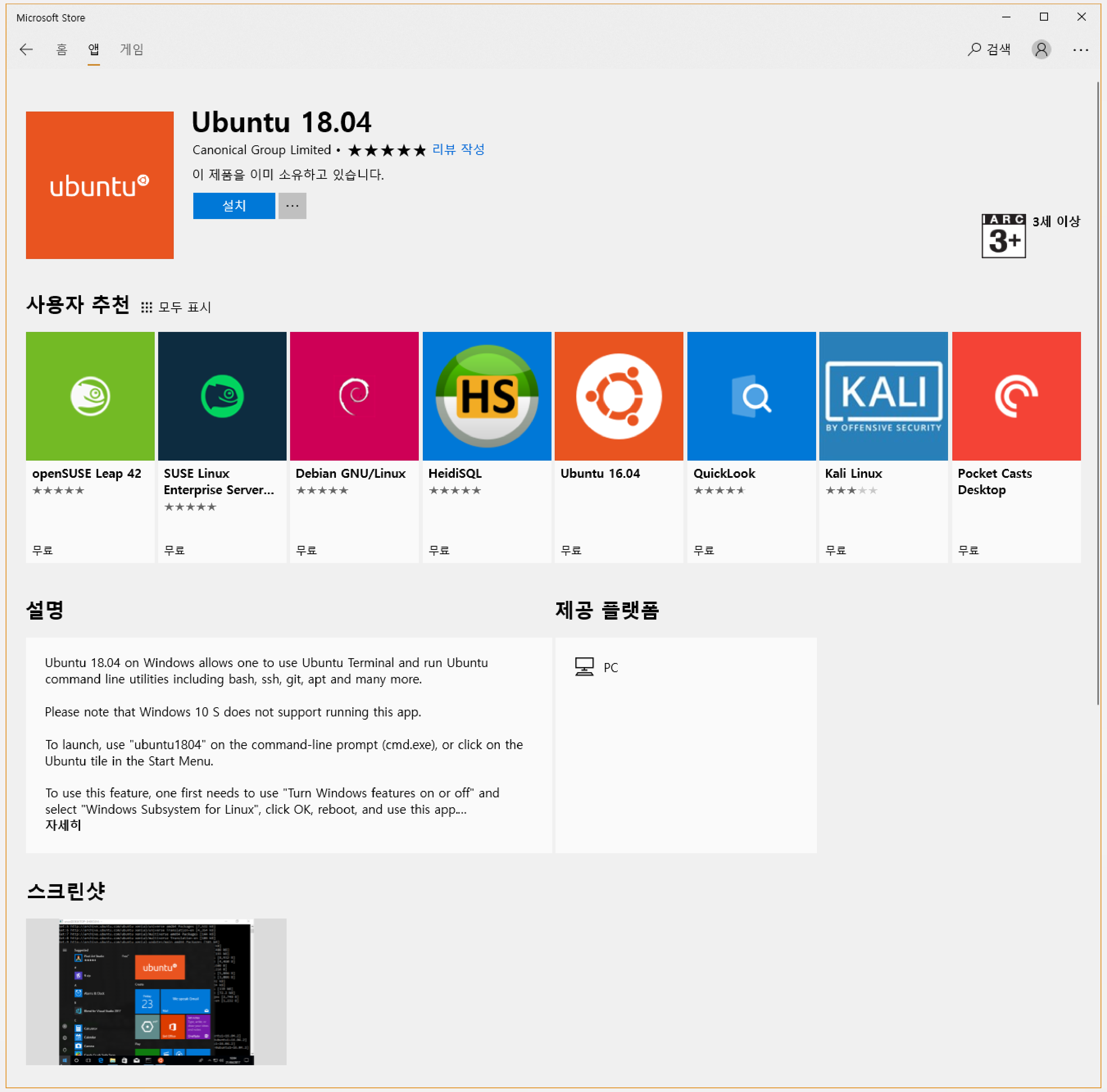
Windows 기능 켜기/끄기

기능을 사용하려면 해당 확인란을 선택하고 기능을 사용하지 않으려면 확인란의 선택을 취소하십시오. 확인란이 검은 사각형으로 채워진 경우는 해당 기능의 일부만이 사용되고 있는 것입니다.

- ☐ .NET Framework 3.5(.NET 2.0 및 3.0 포함)
- ■ .NET Framework 4.7 Advanced Services
- ☐ Active Directory LDS(Lightweight Directory Services)
- ☑ Hyper-V
- ☑ Internet Explorer 11
- ☑ Linux용 Windows 하위 시스템
- ☐ Mic
- ☑ Mic
- ☑ Microsoft XPS Document Writer
- ☐ MultiPoint Connector
- ☐ NFS용 서비스

Windows에서 네이티브 사용자 모드 Linux 셸 및 도구를 실행
비스와 환경을 제공합니다.

확인 | 취소



Microsoft Store

홈 앱 게임 검색

Ubuntu 18.04

Canonical Group Limited • ★★★★★ 리뷰 작성

이 제품을 이미 소유하고 있습니다.

설치

IARC 3+ 3세 이상

사용자 추천 모두 표시

openSUSE Leap 42	★★★★★	무료
SUSE Linux Enterprise Server...	★★★★★	무료
Debian GNU/Linux	★★★★★	무료
HeidiSQL	★★★★★	무료
Ubuntu 16.04		무료
QuickLook	★★★★★	무료
Kali Linux	★★★★★	무료
Pocket Casts Desktop		무료

설명

Ubuntu 18.04 on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

Please note that Windows 10 S does not support running this app.

To launch, use "ubuntu1804" on the command-line prompt (cmd.exe), or click on the Ubuntu tile in the Start Menu.

To use this feature, one first needs to use "Turn Windows features on or off" and select "Windows Subsystem for Linux", click OK, reboot, and use this app...
자세히

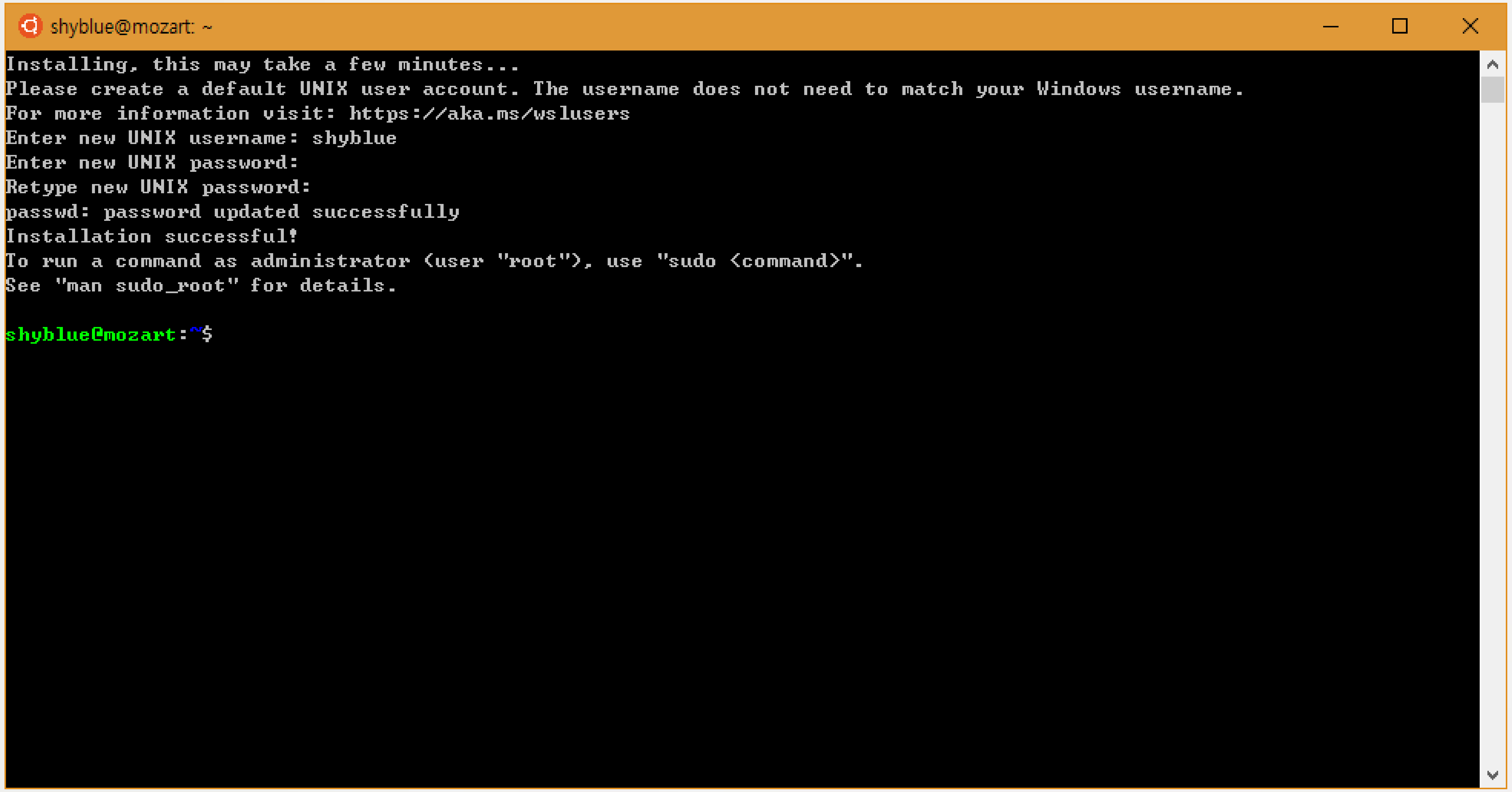
제공 플랫폼

PC

스크린샷

4. Pre-requirement – WSL

Windows Subsystem for Linux



```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: shyblue
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

shyblue@mozart:~$
```

4. Pre-requirement – WSL

Windows Subsystem for Linux

1. WSL update & upgrade

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2. Build essential install and Cmake

```
$ sudo apt-get install g++ build-essential cmake
```

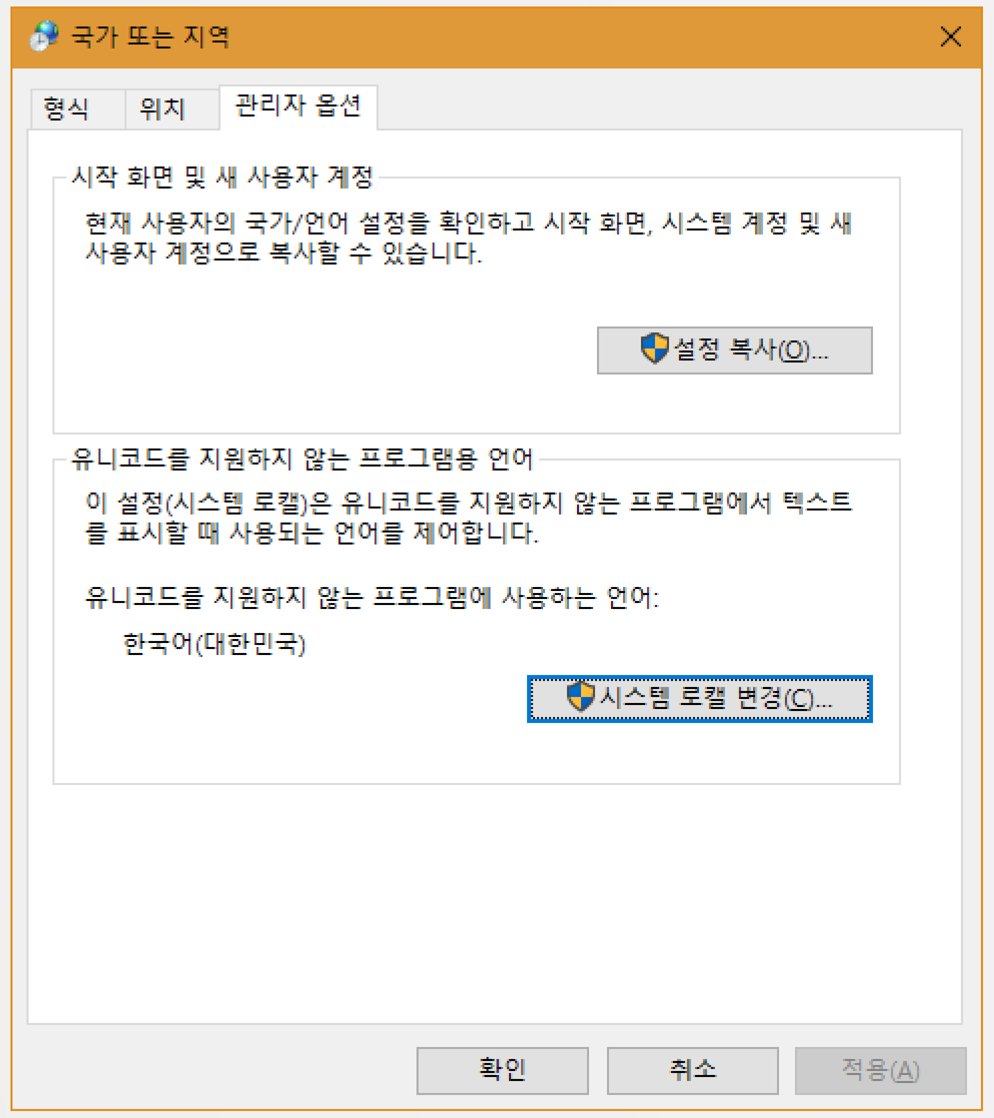
```
shyblue@mozart:/$ g++ --version
g++ (Ubuntu 7.3.0-16ubuntu3) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

C++ 17 표준 중 P0512R0 만 제외하고 모두 지원 (클래스 템플릿 인자 추론)

4. Pre-requirement – UTF-8

System Locale

Windows


국가 또는 지역

형식 | 위치 | 관리자 옵션

시작 화면 및 새 사용자 계정

현재 사용자의 국가/언어 설정을 확인하고 시작 화면, 시스템 계정 및 새 사용자 계정으로 복사할 수 있습니다.

설정 복사(O)...

유니코드를 지원하지 않는 프로그램용 언어

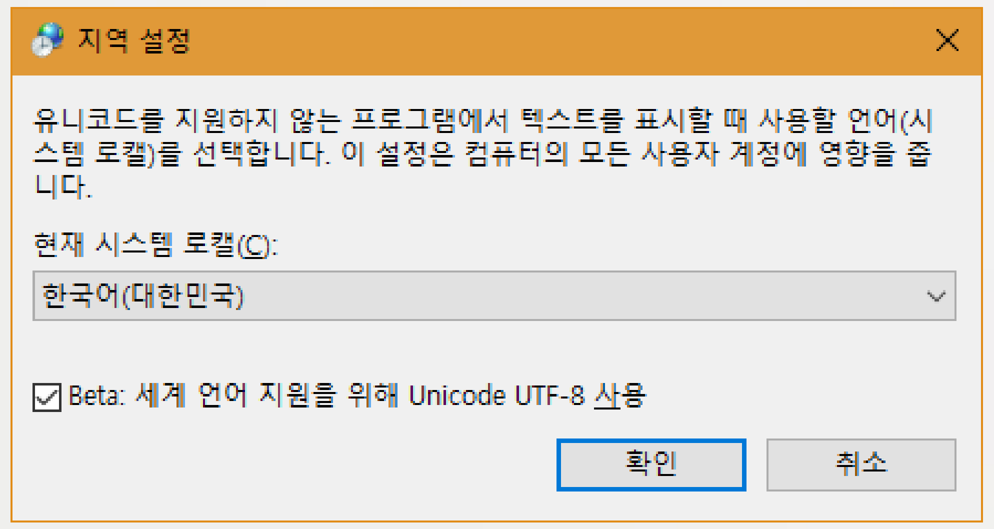
이 설정(시스템 로캘)은 유니코드를 지원하지 않는 프로그램에서 텍스트를 표시할 때 사용되는 언어를 제어합니다.

유니코드를 지원하지 않는 프로그램에 사용하는 언어:

한국어(대한민국)

시스템 로캘 변경(C)...

확인 | 취소 | 적용(A)


지역 설정

유니코드를 지원하지 않는 프로그램에서 텍스트를 표시할 때 사용할 언어(시스템 로캘)를 선택합니다. 이 설정은 컴퓨터의 모든 사용자 계정에 영향을 줍니다.

현재 시스템 로캘(C):

한국어(대한민국)

☑ Beta: 세계 언어 지원을 위해 Unicode UTF-8 사용

확인 | 취소

Linux

```
export LC_ALL="ko_KR.UTF-8"
export LANG="ko_KR.UTF-8"
```

4. Pre-requirement - libcurl

curl:// libcurl

libcurl - the multiprotocol file transfer library

libcurl is a free and easy-to-use client-side URL transfer library, supporting DICT, FILE, **FTP**, FTPS, Gopher, **HTTP**, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP. libcurl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, Kerberos), file transfer resume, http proxy tunneling and more!

libcurl is highly portable, it builds and works identically on numerous platforms, including Solaris, NetBSD, FreeBSD, OpenBSD, Darwin, HPUX, IRIX, AIX, Tru64, Linux, UnixWare, HURD, Windows, Amiga, OS/2, BeOs, Mac OS X, Ultrix, QNX, OpenVMS, RISC OS, Novell NetWare, DOS and more...

libcurl is free, thread-safe, IPv6 compatible, feature rich, well supported, fast, thoroughly documented and is already used by many known, big and successful companies.

libcurl static library build

Support PROTOCOLS – FTP, HTTP

Target – Win64, x86_64

Library output - Static

Windows

```
\projects\Windows\VC141\lib
        libcurl.sln

        libcurl.lib
        libcurld.lib
```

Linux

```
./Configure
make

libcurl.a
libcurl.la
```

4. Pre-requirement - Boost

C++ 개발자들의 영원한 동반자 boost C++ LIBRARIES

Windows

Linux

Prebuilt binaries for MSVC14

Build from source in WSL
Using b2

5. Modern C++ Boost singleton

Global? Singleton!

```
#include <boost/serialization/singleton.hpp>

class Configure : public boost::serialization::singleton<Configure>
{
  …
}

#define ST_CONFIG Configure::get_mutable_instance()
```

Use simple

```
ST_CONFIG.Initialize("./teac.conf");

std::string logPackBasePath = ST_CONFIG.GetConfigureData<std::string>("BASE_DIR", "OUT");
bool downloadOnly = ST_CONFIG.GetConfigureData<bool>("DO_REQUEST", false);
int requestThread = ST_CONFIG.GetConfigureData<short>("THREAD_CNT", 8);
```

INI? JSON? No problem!

```
#include <boost/property_tree/ptree.hpp>

boost::property_tree::ptree ini_tree_;

fs::path filePath(config_file_);
auto fileSize = fs::file_size(filePath);
if(fileSize == 0){
  …
}
boost::property_tree::ini_parser::read_ini(config_file_, ini_tree_);

template<typename _T>
const _T GetConfigureData(std::string key, const _T default_value)
{
  return ini_tree_.get<_T>(key, default_value);
}

std::string logPackBasePath = ST_CONFIG.GetConfigureData<std::string>("BASE_DIR", "OUT");
std::string startPeriod = ST_CONFIG.GetConfigureData<int>(“PERIOD", 20180804);
bool downloadOnly = ST_CONFIG.GetConfigureData<bool>("DO_REQUEST", false);
int requestThread = ST_CONFIG.GetConfigureData<short>("THREAD_CNT", 8);
```

INI? JSON? No problem!

```
"property" : [
  {
    "symbolname" : "CLobbyServer#1"
  },
  {
    "symbolname" : "CLobbyServer#1"
  } ]
```

```
#include <boost/property_tree/json_parser.hpp>


boost::property_tree::ptree json_tree_;


boost::property_tree::read_json("filename", json_tree_);
boost::property_tree::ptree &children = json_tree_.get_child("property");
for (const auto& kv : children)
{
  const std::string name = kv.second.get<std::string>("symbolname");
...
}

for (const auto& [data, type] : children)
{
  const std::string name = type.get<std::string>("symbolname");
  ...
}
```

Command line argument

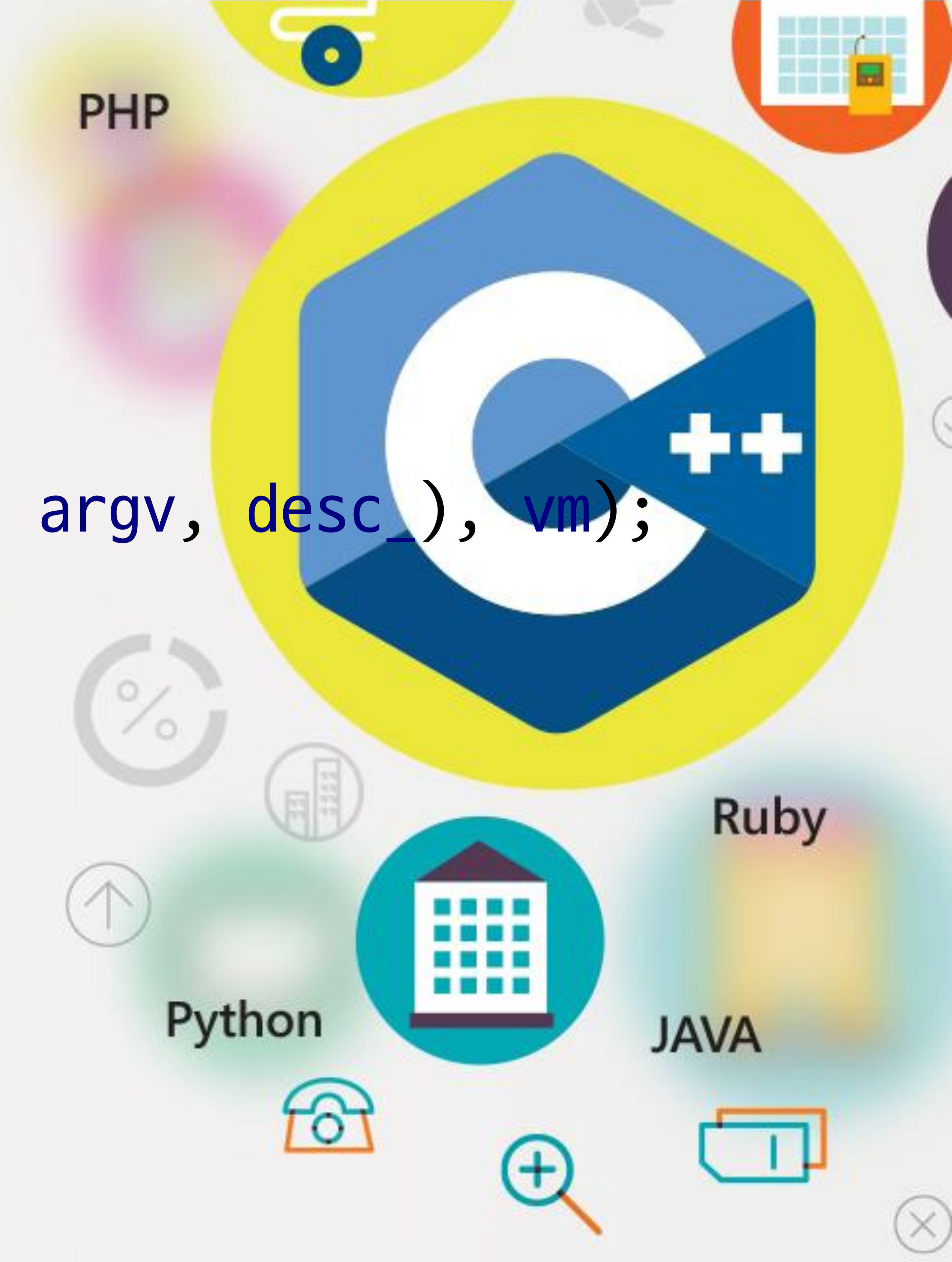
gopt, CmdLine, … program options

```
#include <boost/program_options.hpp>

boost::program_options::options_description desc_;

desc_.add_options()
("help,h", "present the useage of this program.")
("version,v", "print the version number.")
("verbose,V", "print all configuration options.")
("input,i", boost::program_options::value<std::string>(), "input csv file name")
("logdir,l", boost::program_options::value<std::string>(), "base directory name to save logpack file.")
("level,e", boost::program_options::value<std::string>(), "set log level")
("download,d", boost::program_options::value<bool>(), "Downloads logpack files only.")
("thread,t", boost::program_options::value<short>(), "Count of threads to request tm server");
```

5. Modern C++ Boost program options

Command line argument

```
boost::program_options::variables_map vm;
boost::program_options::store(boost::program_options::parse_command_line(argc, argv, desc_), vm);

if (vm.count("help"))
{
  std::cout << "Usage: teac [options]" << std::endl;
  std::cout << desc_;
  return 0;
}

if (vm.count("level")){
  logLevel = vm["level"].as<std::string>();
}
if (vm.count("download")){
  downloadOnly = vm["download"].as<bool>();
}
if (vm.count("thread")){
  requestThread = vm["thread"].as<short>();
}
```



https://github.com/gabime/spdlog

- Very fast - performance is the primary goal (see benchmarks below).
- Headers only, just copy and use.
- Feature rich call style using the excellent fmt library.
- Fast asynchronous mode (optional)
- Custom formatting.
- Conditional Logging
- Multi/Single threaded loggers.
- Various log targets:
- Rotating log files.
- Daily log files.
- Console logging (colors supported).
- syslog.
- Windows debugger (OutputDebugString(..))
- Easily extendable with custom log targets (just implement a single function in the sink interface).
- Severity based filtering - threshold levels can be modified in runtime as well as in compile time.

5. Modern C++ spdlog

Logger

```
std::vector<spdlog::sink_ptr> sinks_;
std::shared_ptr<spdlog::logger> spLogger_;

sinks_.push_back(std::make_shared<spdlog::sinks::daily_file_sink_mt>("logs/server-log",23,59));
sinks_.push_back(std::make_shared<spdlog::sinks::stdout_sink_mt>());

spLogger_ = std::make_shared<spdlog::logger>("teac", sinks_.begin(), sinks_.end()) ;

spdlog::register_logger(spLogger_);

spLogger_->set_pattern("%Y-%m-%d %X,%l,%v");
spLogger_->set_level(spdlog::level::trace);
```

Use

```
ST_LOGGER.log()->info("LOGPACK Download base dir set to LOGPACK/{}", logPackBasePath);
```

5. Modern C++ fmt

{fmt} `#include "spdlog/fmt/fmt.h"`

```cpp
fmt::format("./RESULT/RESULT-{}", file_name)

fmt::format("http://{}:{}/{}", tmServer, tmServerPort, r0001ServerUri);

fmt::format("{}T{}{}", filePath.substr(0, 8), filePath.substr(8, 11), std::string("959"))

auto srcFileName = fmt::format("ftp://{}//DATA/rsd/log/mls/{}/{}", ip, filePath, info.file_nm);
auto outFileName = fmt::format("{}/{}", outFilePath.string(), info.file_nm);

auto json = fmt::format("{{\"trafficTime\" : \"{}\", \"patternVer\" : \"{}\", \"requestSource\" : 1, \"tiType\" :
1}}", kv.first, patternDay);

std::string cmd = fmt::format("./teac -i {0}.csv -l OUT{0} -d {1}", file_name, is_donw_only);
```

libcurl Wrapper class curlite

https://github.com/grynko/curlite

curlite is developed as a lightweight wrapper over cURL library with the following points in mind:

- Support of C++11 features
- Type safety of curl options
- Ease-of-use
- No external dependencies (except libcurl)
- The project is in development stage. Currently only Easy interface is implemented.

curlite FTP

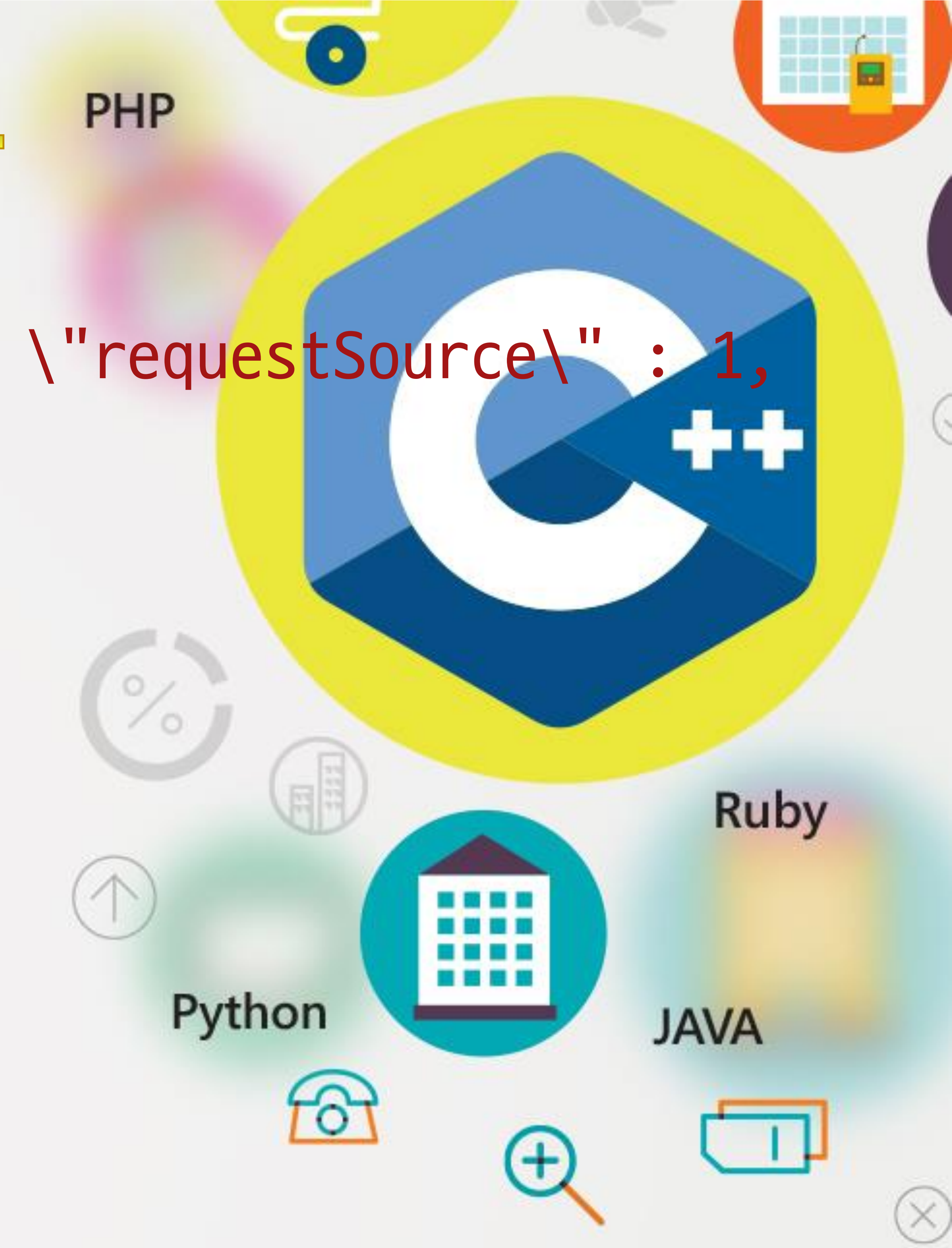
```cpp
#include "curlite.h"

curlite::Easy ftp;

const auto& [userName,passWord] = makeLoginInfo(ip, passwdBase, passwdBaseTidc);

auto srcFileName = fmt::format("ftp://{}//DATA/rsd/log/mls/{}/{}", ip, filePath, info.file_nm);
auto outFileName = fmt::format("{}/{}", outFilePath.string(), info.file_nm);

ftp.set(CURLOPT_USERNAME, userName);
ftp.set(CURLOPT_PASSWORD, passWord);
ftp.set(CURLOPT_CONNECTTIMEOUT_MS, 100000);
ftp.set(CURLOPT_URL, srcFileName);

std::ofstream outFile(outFileName, std::ios::binary);
ftp >> outFile;
outFile.close();
```

curlite HTTP, POST

```cpp
curlite::Easy tmRequest;

curlite::List headerList;

headerList.append("User-Agent: teac/0.0.1");
headerList.append("Accept-Encoding: deflate, gzip");
headerList.append("Content-Type: application/json");
headerList.append("Accept: application/json");
headerList.append("Content-Length: 100");

const std::string r0001Uri = fmt::format("http://{}:{}/{}",
                             tmServer, tmServerPort, r0001ServerUri);
```

curlite HTTP, POST

```
auto json = fmt::format("{{\"trafficTime\" : \"{}\", \"patternVer\" : \"{}\", \"requestSource\" : 1,
\"tiType\" : 1}}", trafficTime, patternDay);

tmRequest.set(CURLOPT_CONNECTTIMEOUT_MS, 15000L);
tmRequest.set(CURLOPT_URL, r0001Uri);
tmRequest.set(CURLOPT_HTTPHEADER, headerList.get());
tmRequest.set(CURLOPT_POST, true);
tmRequest.set(CURLOPT_POSTFIELDS, json);
tmRequest.set(CURLOPT_POSTFIELDSIZE, -1L);

std::stringstream response;
response << tmRequest;

ST_LOGGER.log()->info("Response code : {}", response.str());
if (CURLE_OK == tmRequest.error())
{
  …
}

long httpResCode = tmRequest.getInfo<long>(CURLINFO_RESPONSE_CODE, 404L);
```



curlite GET & Follow Location

```
curlite::Easy nexusRequest;
curlite::List headerList;

auto uri = fmt::format("http://{}/{}?r={}&g={}&a={}&v={}&p=war",
           host, base, repo, group_id, artifact_id, version);

nexusRequest.set(CURLOPT_URL, uri);
nexusRequest.set(CURLOPT_FOLLOWLOCATION, true);

nexusRequest.onHeader_([&headerList](char* header, size_t size)->bool {
  headerList.append(header);
  return true;
});

std::ofstream ofs(fmt::format("{}.{}.war", artifact_id, version), std::ios::binary);
nexusRequest >> ofs;
```

5. Modern C++ filesystem

C++11

Boost::filesystem

C++14

std::experimental::filesystem

C++17

std::filesystem

std::filesystem

```
namespace fs = std::experimental::filesystem;

const fs::path path(logpack_file_name);

if(!fs::exists(path))

const auto file_size = fs::file_size(path);

auto outFilePath = fs::path(fmt::format("./LOGPACK/{}/{}", logPackBasePath, filePath));
fs::create_directories(outFilePath);


std::vector<std::string> v;
for (const auto& e : fs::directory_iterator(inputPath))
{
  if (!boost::filesystem::is_directory(e))
  {
    v.emplace_back(e.path().generic_string());
  }
}
```

Parallel for

```
std::vector<std::thread> rsdWebThreads;

const auto blockSize = length / nThread;

if (blockSize)
{
  size_t blockStart = 0;
  for (int i = 0; i < (nThread - 1); ++i)
  {
    auto blockEnd = blockStart + blockSize;
    if (blockEnd > length) blockEnd = length;

    rsdWebThreads.emplace_back(std::thread(&Rp001Request, blockStart, blockEnd, sessionInfo));
    blockStart = blockEnd;
  }
}

if (blockStart < length) Rp001Request(blockStart, length, sessionInfo);
for (auto& t : rsdWebThreads)
  if(t.joinable())
    t.join();
```

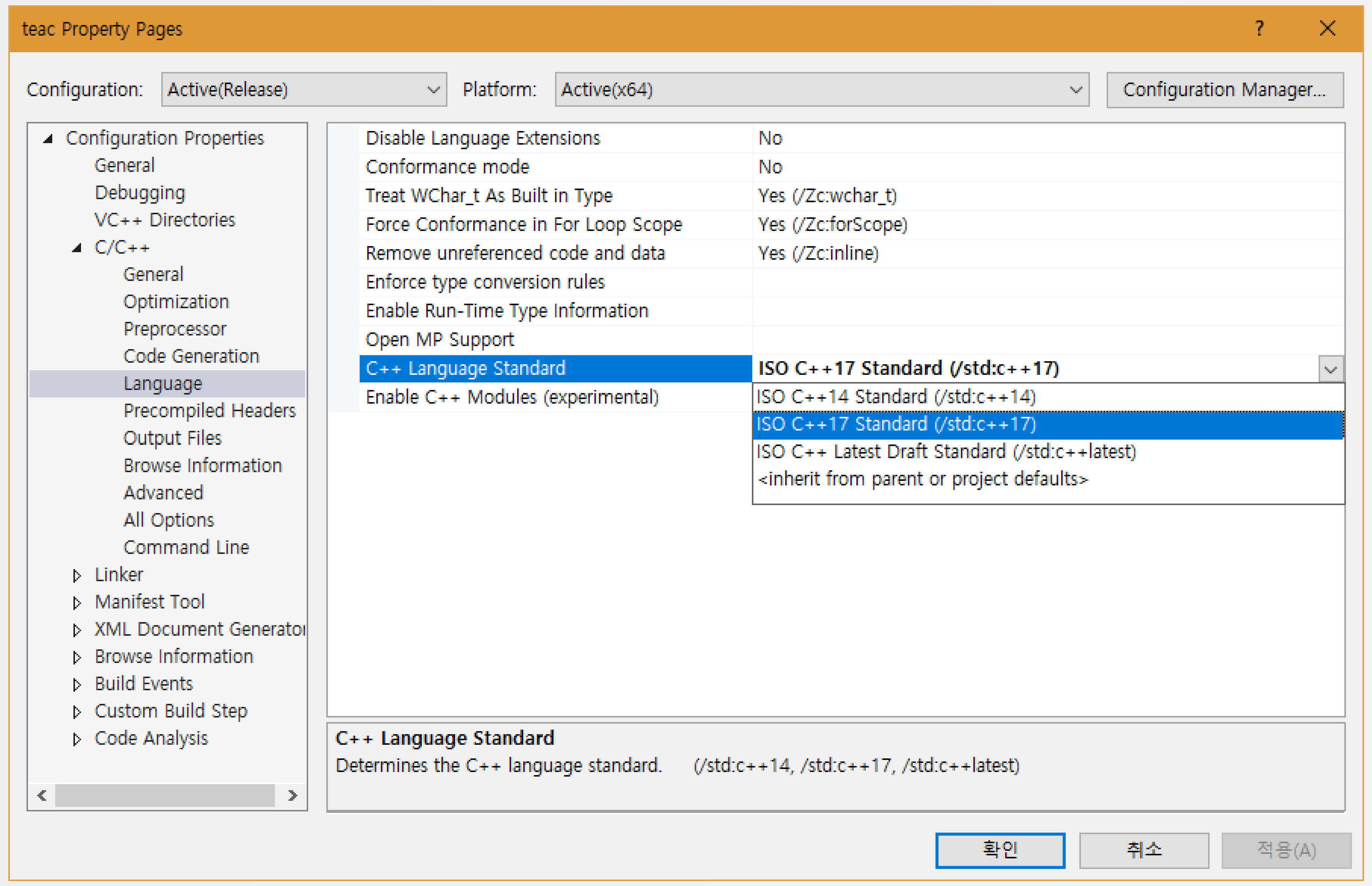
5. Modern C++ thread

detach

```
auto execAsync = std::thread([&]() {
boost::process::ipstream is;
std::string cmd = fmt::format("./teac -i {0}.csv -l OUT{0} -d {1}", file_name, is_donw_only);
boost::process::child teac(cmd, boost::process::std_out > is);

std::string line;

while (teac.running() && std::getline(is, line) && !line.empty())
  {
    …
  }
  teac.wait();
});

execAsync.detach();
```

6. Conclusion

Build Windows



teac Property Pages

Configuration: Active(Release) Platform: Active(x64) Configuration Manager...

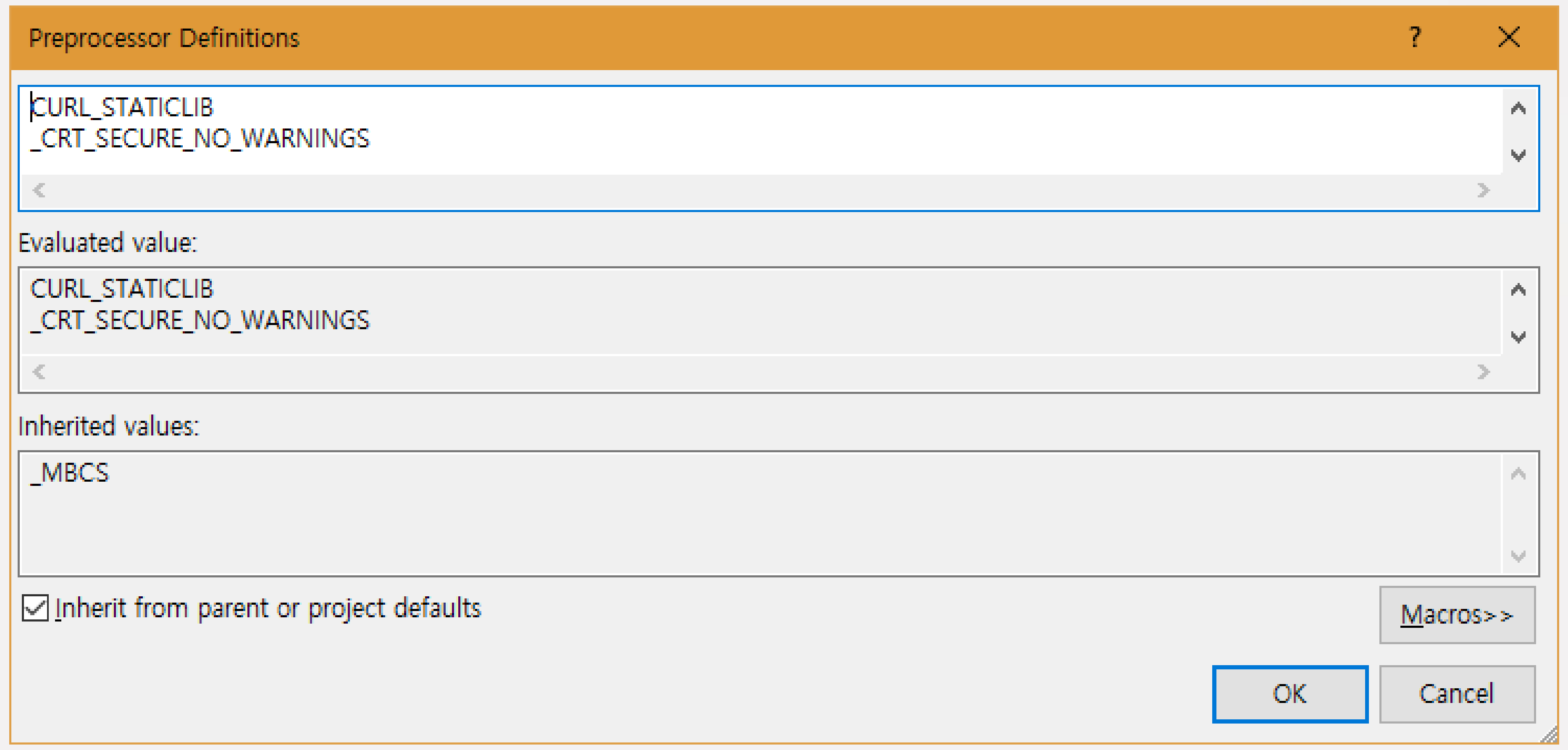
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - All Options
    - Command Line
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

Disable Language Extensions	No
Conformance mode	No
Treat WChar_t As Built in Type	Yes (/Zc:wchar_t)
Force Conformance in For Loop Scope	Yes (/Zc:forScope)
Remove unreferenced code and data	Yes (/Zc:inline)
Enforce type conversion rules	
Enable Run-Time Type Information	
Open MP Support	
C++ Language Standard	ISO C++17 Standard (/std:c++17)
Enable C++ Modules (experimental)	

ISO C++14 Standard (/std:c++14)
ISO C++17 Standard (/std:c++17)
ISO C++ Latest Draft Standard (/std:c++latest)
<inherit from parent or project defaults>

C++ Language Standard
Determines the C++ language standard. (/std:c++14, /std:c++17, /std:c++latest)

확인 취소 적용(A)



Preprocessor Definitions

```
CURL_STATICLIB
_CRT_SECURE_NO_WARNINGS
```

Evaluated value:

```
CURL_STATICLIB
_CRT_SECURE_NO_WARNINGS
```

Inherited values:

```
_MBCS
```

☑ Inherit from parent or project defaults

Macros>> OK Cancel

6. Conclusion

Build Linux

```
CMakeList.txt
project (teac)

set(Boost_INCLUDE_DIR "~/devel/boost/include")
set(PROJECT_LIB_DIR "~/devel/boost/lib")
set(PROJECT_INCLUDE_DIR ${PROJECT_SOURCE_DIR}/common)
include_directories(${Boost_INCLUDE_DIR})
include_directories(${PROJECT_INCLUDE_DIR})
include_directories(${PROJECT_SOURCE_DIR})

set(CMAKE_CXX_FLAGS "-DCURL_STATICLIB -DZIP_STD -D_REENTRANT -O2 -std=c++17 ")

add_subdirectory(teac)
```

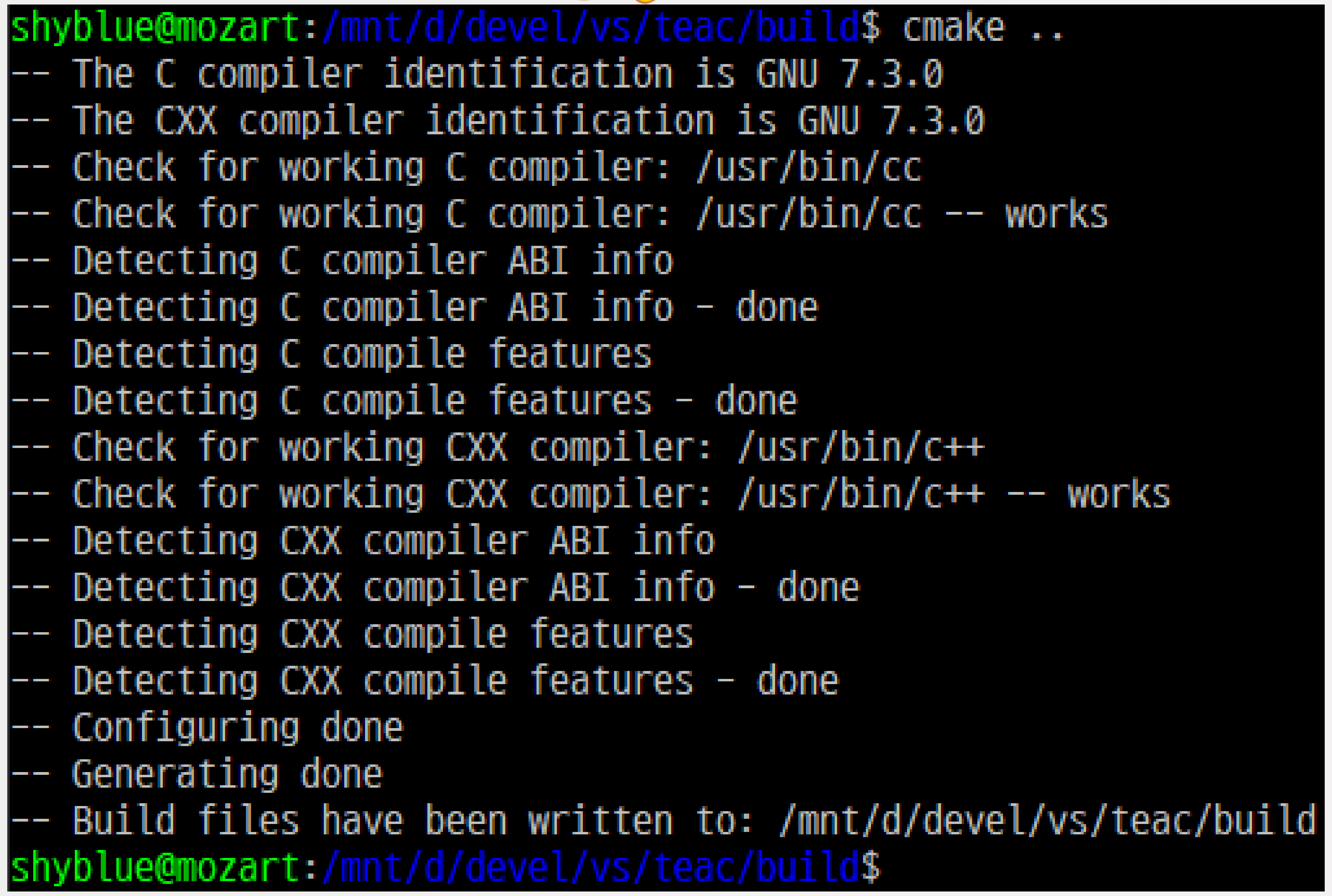
6. Conclusion

Build Linux

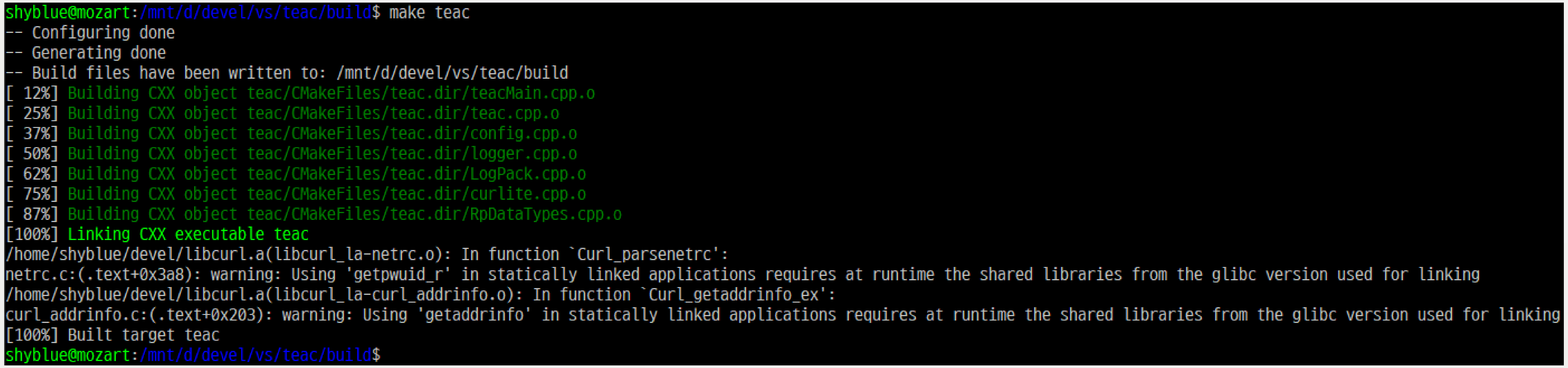
```
project (teac)

set(SOURCES
      teacMain.cpp
      teac.cpp
      config.cpp
      logger.cpp
      LogPack.cpp
      curlite.cpp
      RpDataType.coo
   )

add_executable(teac ${SOURCES})

SET(CMAKE_EXE_LINKER_FLAGS "-static")

target_link_libraries(teac /home/shyblue/devel/boost/lib/libboost_serialization.a)
target_link_libraries(teac /home/shyblue/devel/boost/lib/libboost_program_options.a)
target_link_libraries(teac /home/shyblue/devel/libcurl.a)
target_link_libraries(teac  "-lstdc++ -lstdc++fs -lpthread")
```

6. Conclusion

Build Linux



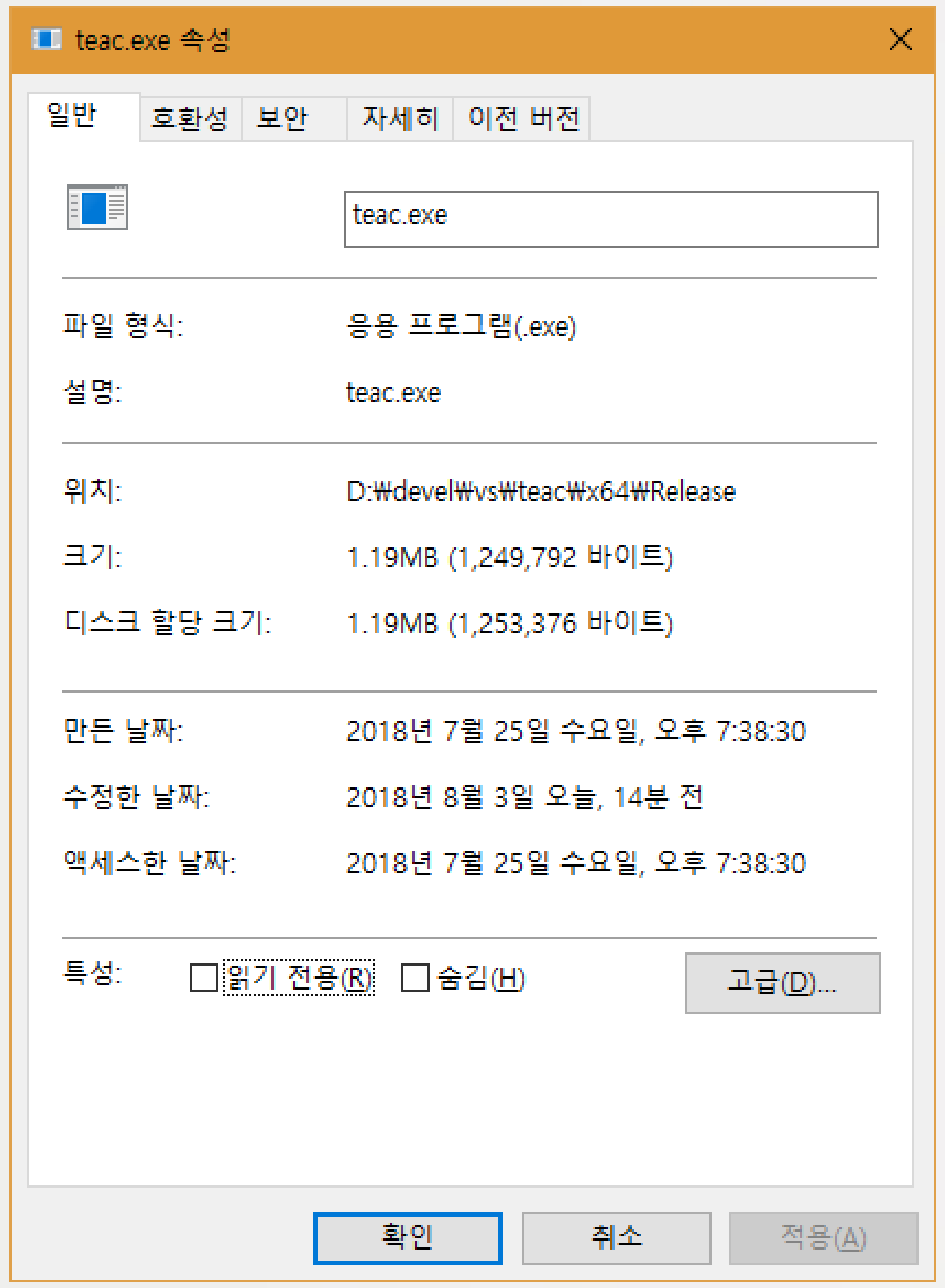
```
shyblue@mozart:/mnt/d/devel/vs/teac/build$ cmake ..
-- The C compiler identification is GNU 7.3.0
-- The CXX compiler identification is GNU 7.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/d/devel/vs/teac/build
shyblue@mozart:/mnt/d/devel/vs/teac/build$
```



```
shyblue@mozart:/mnt/d/devel/vs/teac/build$ make teac
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/d/devel/vs/teac/build
[ 12%] Building CXX object teac/CMakeFiles/teac.dir/teacMain.cpp.o
[ 25%] Building CXX object teac/CMakeFiles/teac.dir/teac.cpp.o
[ 37%] Building CXX object teac/CMakeFiles/teac.dir/config.cpp.o
[ 50%] Building CXX object teac/CMakeFiles/teac.dir/logger.cpp.o
[ 62%] Building CXX object teac/CMakeFiles/teac.dir/LogPack.cpp.o
[ 75%] Building CXX object teac/CMakeFiles/teac.dir/curlite.cpp.o
[ 87%] Building CXX object teac/CMakeFiles/teac.dir/RpDataTypes.cpp.o
[100%] Linking CXX executable teac
/home/shyblue/devel/libcurl.a(libcurl_la-netrc.o): In function `Curl_parsenetrc':
netrc.c:(.text+0x3a8): warning: Using 'getpwuid_r' in statically linked applications requires at runtime the shared libraries from the glibc version used for linking
/home/shyblue/devel/libcurl.a(libcurl_la-curl_addrinfo.o): In function `Curl_getaddrinfo_ex':
curl_addrinfo.c:(.text+0x203): warning: Using 'getaddrinfo' in statically linked applications requires at runtime the shared libraries from the glibc version used for linking
[100%] Built target teac
shyblue@mozart:/mnt/d/devel/vs/teac/build$
```

6. Conclusion

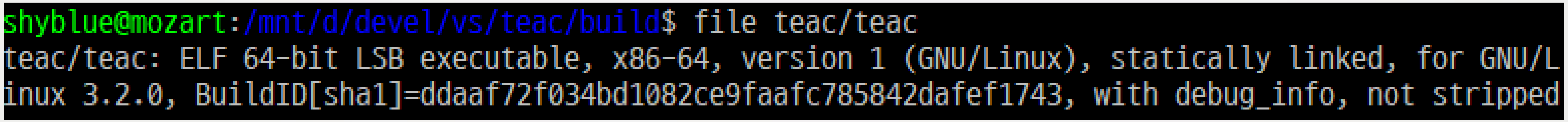
Conclusion


teac.exe 속성

일반 | 호환성 | 보안 | 자세히 | 이전 버전

teac.exe

파일 형식:	응용 프로그램(.exe)
설명:	teac.exe
위치:	D:₩devel₩vs₩teac₩x64₩Release
크기:	1.19MB (1,249,792 바이트)
디스크 할당 크기:	1.19MB (1,253,376 바이트)
만든 날짜:	2018년 7월 25일 수요일, 오후 7:38:30
수정한 날짜:	2018년 8월 3일 오늘, 14분 전
액세스한 날짜:	2018년 7월 25일 수요일, 오후 7:38:30
특성:	☐ 읽기 전용(R) ☐ 숨김(H) 고급(D)...


```
shyblue@mozart:/mnt/d/devel/vs/teac/build$ file teac/teac
teac/teac: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/L
inux 3.2.0, BuildID[sha1]=ddaaf72f034bd1082ce9faafc785842dafef1743, with debug_info, not stripped
```

감사합니다.