

Getting Started with Visual Studio Code

VSCode Step1. GCC 및 MSYS2 설치

- MacOS User -

1. GCC(g++) 설치
2. GCC(g++) 프로그램 테스트

- Windows User -

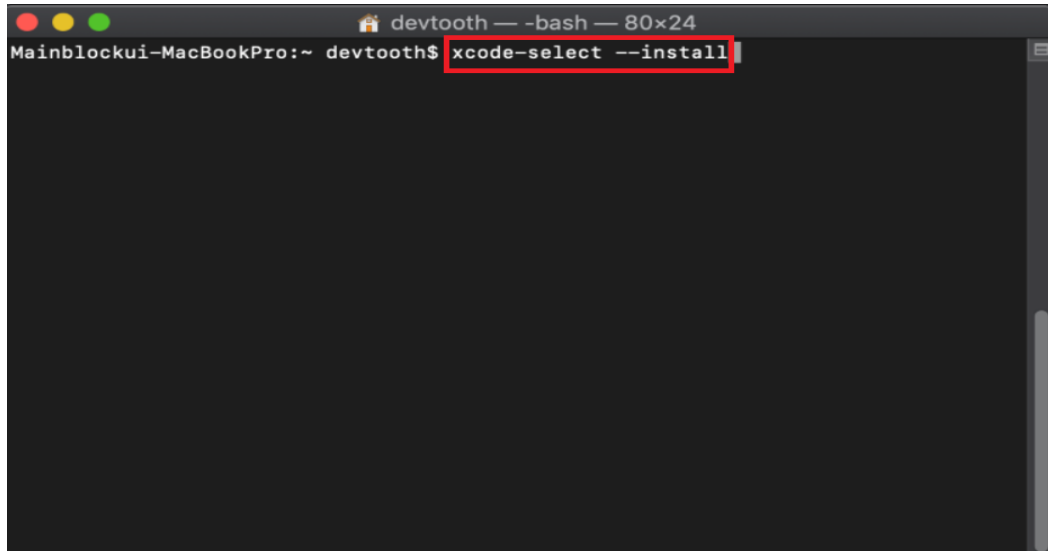
3. MSYS2와 MinGW-w64 패키지 설치
4. MSYS2와 MinGW-w64 환경변수 설정
5. GCC(g++) 프로그램 테스트 & 디버깅

* MacOS 사용자는 1단계와 2단계를 참고하여 g++을 설치해줍니다.

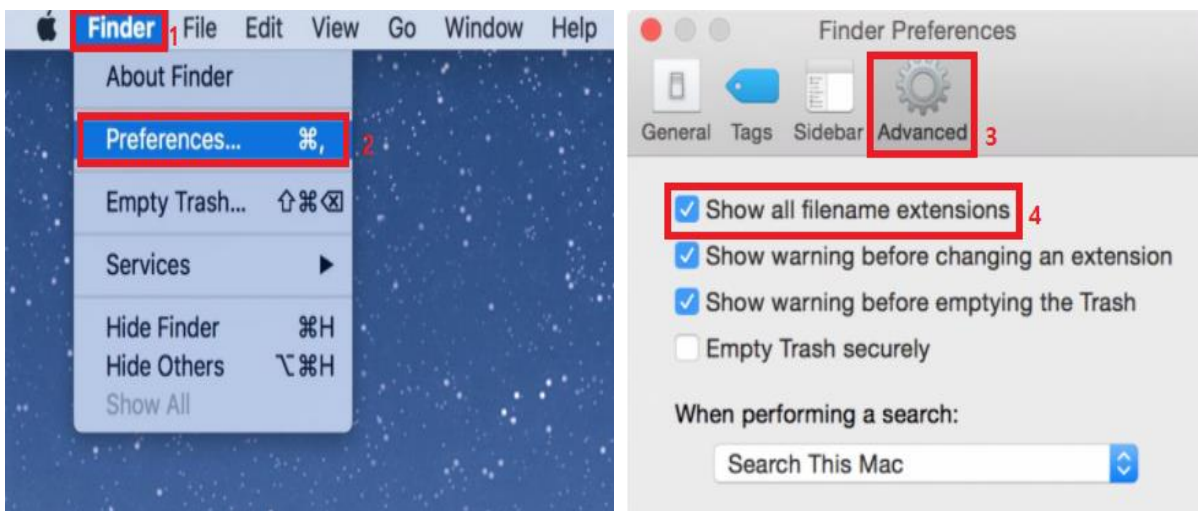
Windows 사용자는 3단계, 4단계, 그리고 5단계를 참고하여 MSYS2와 MinGW-w64를 설치한 후에 g++을 설치해줍니다.

1. GCC(g++) 설치(MacOS User)

- 가장 먼저 터미널을 열어줍니다. 터미널은 Mac의 검색 창에 “terminal” 검색 혹은 [command] + [Space Bar] 단축키를 통해 열 수 있습니다. 터미널이 열렸다면 “xcode-select --install” 명령어를 입력하여 g++을 설치해줍니다.



- g++ 설치가 모두 완료되었다면 터미널에 “g++ -v” 명령어를 입력하여 g++이 성공적으로 설치되었는지 확인해줍니다. 성공적으로 설치 완료했다면 자신이 다운로드 받은 g++의 버전 및 부가 정보가 터미널에 출력됩니다.
- 파일 확장명이 모두 나타나도록 설정해줍니다. 파일 충돌, 오류 방지, 실행 파일 확장명 표시(exe) 등을 위해 설정해주어야 합니다. Mac의 좌측 상단에서 [Finder]을 클릭한 후에 [Preferences...]을 선택해줍니다. Finder Preferences 설정 창이 나타났다면 [Advanced]을 선택하여 들어가 “Show all filename extensions”을 체크(선택)해줍니다.



2. GCC(g++) 프로그램 테스트(MacOS User)

- GCC(g++)는 일종의 플랫폼과도 같습니다. 다른 텍스트 에디터 없이 설치한 g++만을 통해서도 프로그램을 실행할 수 있습니다. **성공적으로 g++이 설치되었음을 확인하기 위해 업데이트 받아온 nowic 폴더 안의 "hello.cpp" 파일을 이용하여 간단한 테스트를 진행해보아야 합니다.**

- Bash(Mac 터미널)을 이용한 프로그램 실행 테스트

1. 먼저 Bash(Mac 터미널)을 실행하여 자신이 업데이트를 받아온 nowic 폴더로 이동합니다. 터미널에 "**cd [자신의 nowic 폴더 위치]**"을 입력하여 이동할 수 있습니다. 먼저 Mac 검색 창에 nowic 폴더를 검색하여 Path를 확인해주는 것이 안전합니다. 예를 들어 자신의 **nowic 폴더가 바탕화면에 존재한다면 "cd Desktop/nowic"를 터미널에 입력하여 이동할 수 있습니다.** 이때에 주의해야 할 점은 Mac에서 각 Path는 **/ (일반 슬래시)**로 연결되는 것입니다.
2. Compile을 위해 현재 열려있는 터미널에 "**g++ -std=c++11 hello.cpp -o hello**"을 입력합니다. 이때에 g++ 뒤의 "hello.cpp" 파일은 빌드할 파일을 나타내며, -o 뒤의 "hello"는 자신이 생성할 실행 파일의 이름을 나타냅니다. Compile에 성공했다면 해당 폴더에 "**hello.exe**"라는 실행 파일(Executable file)이 생성되었음을 확인할 수 있습니다. (MacOS의 경우에는 실행 파일 뒤의 exe 확장명이 나타나지 않을 수도 있습니다.)

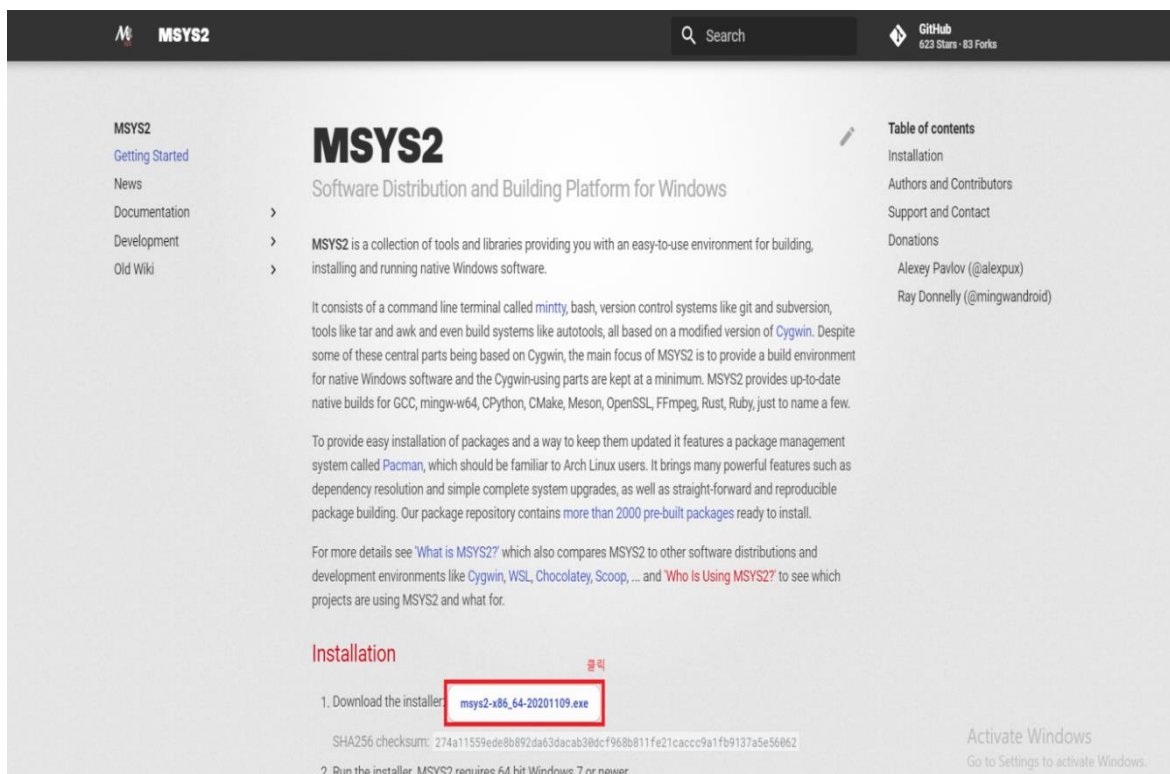
```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello, World!\n";
    return 0;
}
```

(hello.cpp)

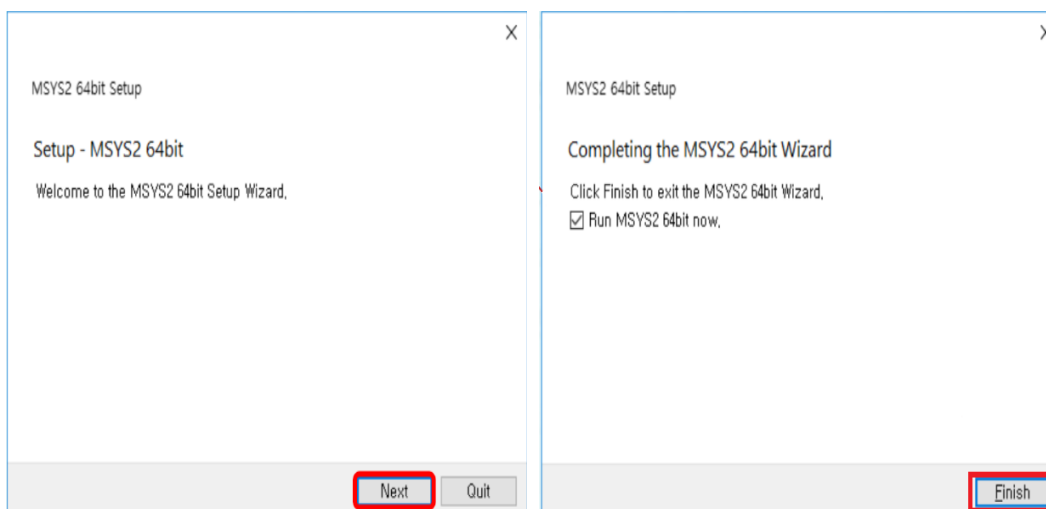
3. Compile하여 생성한 실행 파일(hello.exe)을 실행하기 위해 터미널에 "**./hello**"을 입력합니다. 이때에 ./ 뒤의 "hello"는 자신이 생성한 실행 파일의 이름을 나타냅니다. 프로그램 실행에 성공했다면 "**Hello World!**"라는 메시지를 터미널을 통해 확인할 수 있습니다.

3. MSYS2와 MinGW-w64 패키지 설치(Windows User)

- 가장 먼저 <https://www.msys2.org>에 접속하여 MSYS2를 설치해줍니다. 이때에 MinGW-w64를 먼저 설치할 경우에 설치 오류가 발생하는 경우가 많기에 **꼭 MSYS2를 먼저 설치 해주어야 합니다.** (오류가 발생한다면 기존의 모든 폴더를 완전히 삭제한 후에 재설치를 해야 하므로 굉장히 번거롭습니다.)



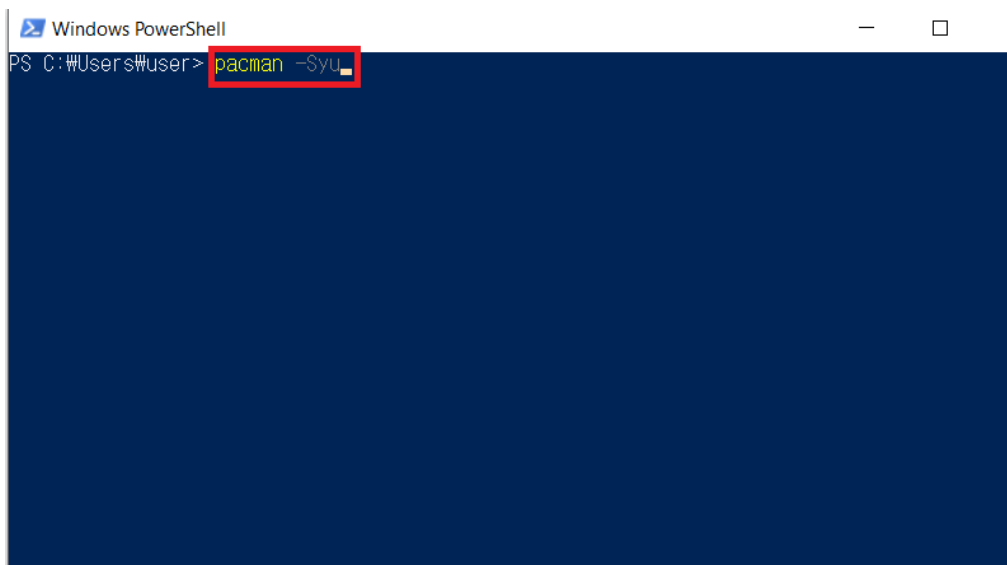
- 다운로드 받은 **msys2-x86_64-[다운로드 받은 버전].exe** 파일을 실행하여 MSYS2를 설치한 후에 실행시켜 줍니다. (다운로드 받는 데에 시간이 많이 소요될 수 있습니다.)



- MSYS2 설치 후에 실행에 성공했다면 MSYS2 콘솔 창을 확인할 수 있습니다. (다음 단계를 위해 콘솔 창을 닫지 않는 것이 좋습니다.)



- Command Prompt(cmd) 혹은 Windows PowerShell을 실행하여 “pacman -Syu” 명령어를 입력한 후에 [Enter]을 눌러 패키지 목록과 MSYS2를 업데이트 합니다.



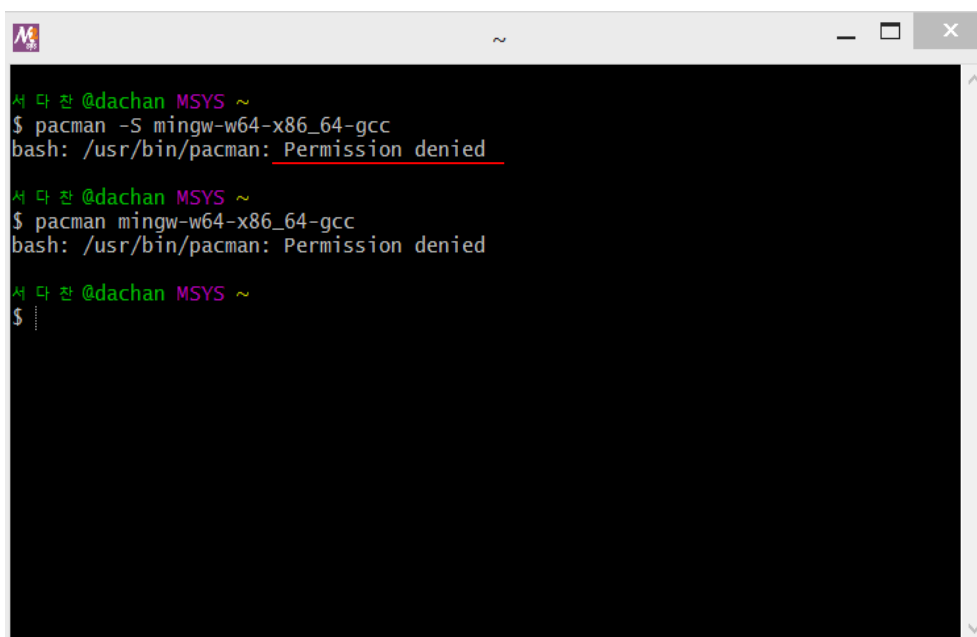
- “설치를 진행하시겠습니까? [Y/n]”라는 질문이 콘솔 창에 나오면 Y를 입력한 후에 [Enter]을 눌러줍니다. 설치 진행 중에 나오는 경고 메시지는 모두 무시하고 Y와 [Enter]을 입력하여 설치를 진행하면 됩니다.

- MSYS2를 실행하여 콘솔 창에 “**pacman -S mingw-w64-x86_64-gcc**”을 입력하여 GCC를 설치해줍니다. (만약에 MSYS2 콘솔 창이 사라졌거나 닫았다면 Windows 검색 창에 MSYS2을 검색하여 “**MSYS2 MSYS**”을 실행하면 됩니다.)



- 설치할 때에 나오는 질문은 모두 콘솔 창에 **Y**를 입력한 후에 **[Enter]**을 눌러 응답하면 됩니다. 설치 진행 중에 나오는 **경고 메시지는 모두 무시하고 Y와 [Enter]을 입력하여 설치를 진행**하면 됩니다.

- 만약 “**Permission denied**”라는 오류 메시지가 나타난다면 패키지 목록과 MSYS2를 업데이트하기 위해 사용했던 “**pacman -Syu**” 명령어를 대신하여 콘솔 창에 “**pacman -Syu --ignore pacman**” 명령어를 입력한 후에 **[Enter]**을 눌러줍니다. (4 페이지 참조)



The following seems to be a work-around to my problems:

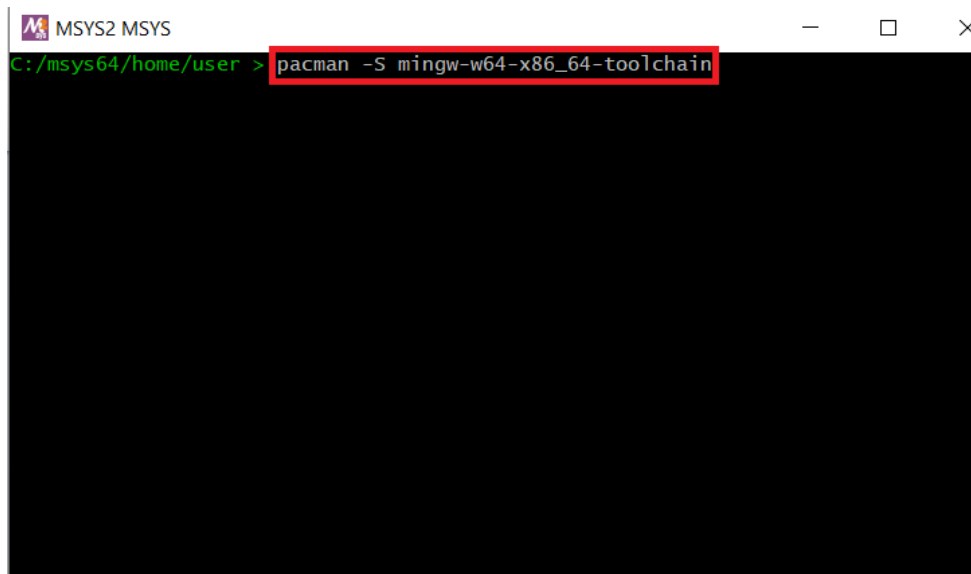
- install `msys2-x86_64-20161025.exe` (from http://repo.msys2.org/distrib/x86_64/) rather than `msys2-x86_64-20180531.exe`
- Use `pacman -Syu --ignore pacman` rather than `pacman -Syu` (giving `warning: pacman: ignoring package upgrade (5.0.1-1 => 5.1.0-4)`)
- Use `pacman -Su --ignore pacman --force` rather than `pacman -Su`

The `--force` option above appears to be required to avoid this error:

```
error: failed to commit transaction (conflicting files)
coreutils: /usr/lib/coreutils/libstdbuf.so exists in filesystem
Errors occurred, no packages were upgraded.
```

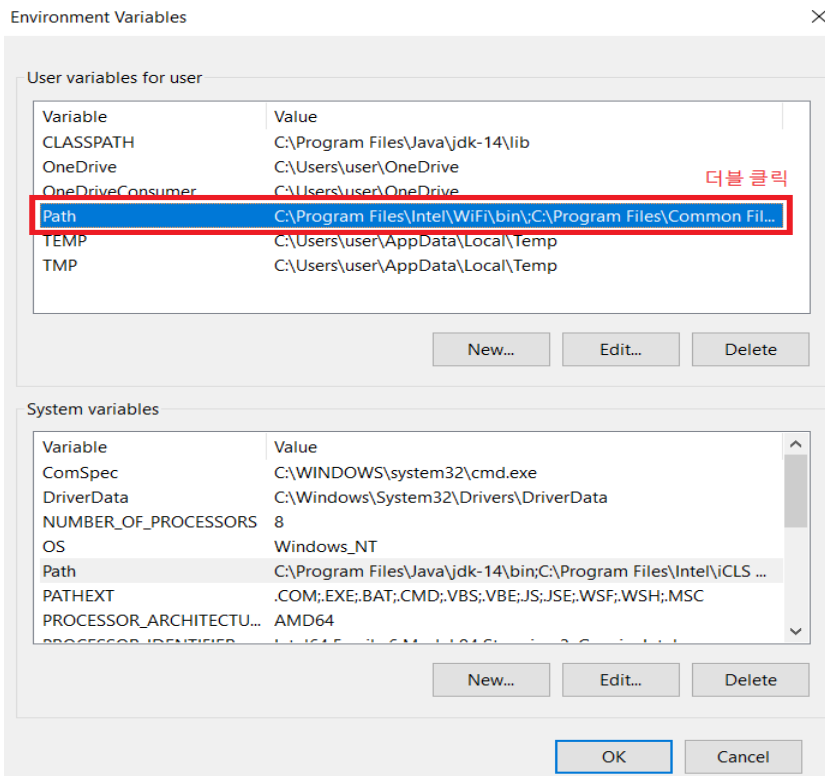
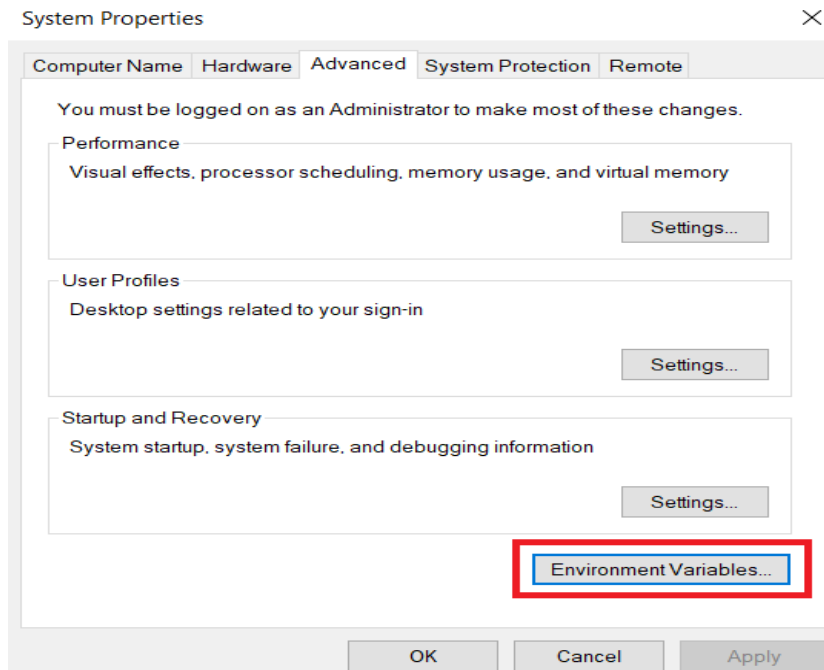
See [#1024](#).

- 같은 방법으로 콘솔 창에 **`pacman -S mingw-w64-x86_64-toolchain`**을 입력하여 Toolchain을 설치해줍니다. Toolchain은 `make`, `gdb` 등과 같이 GCC에 함께 쓰이는 다양한 툴들을 포함하고 있습니다.

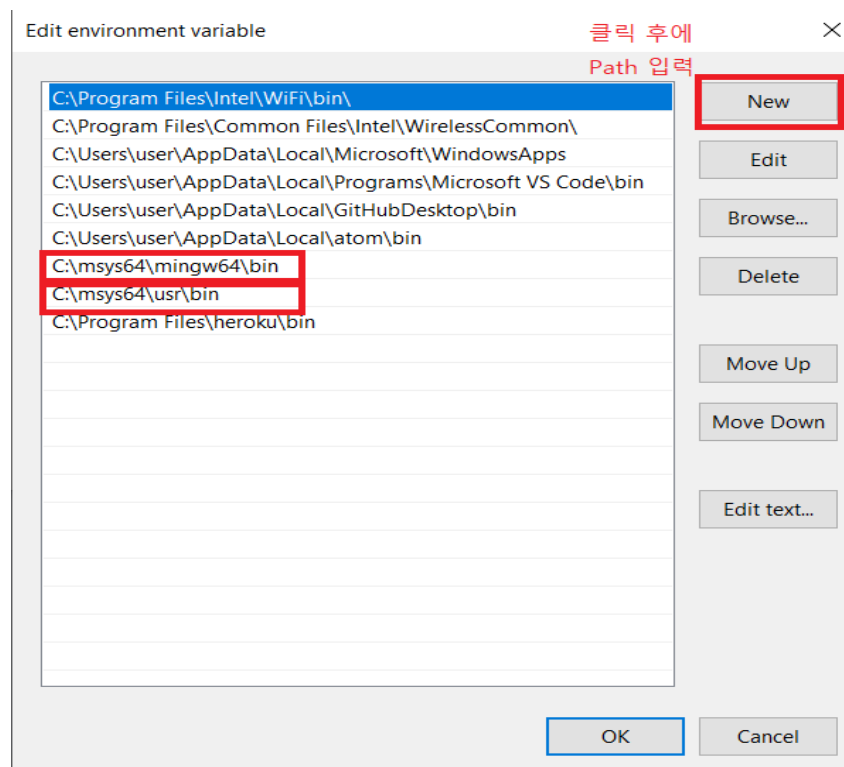


4. MSYS2와 MinGW-w64 환경변수 설정(Windows User)

- Windows 검색 창에 “시스템 환경 변수 편집” (한국어) 또는 “Edit the system environment variables” (영어)를 검색해줍니다.
- “Path”를 더블 클릭하여 사용자 환경 변수 Path를 편집해줍니다.



- **g++.exe 파일이 존재하는 Path**를 새로 입력하거나 붙여 넣어 줍니다. Windows 검색 창에 **"g++"**을 검색하여 폴더의 위치를 먼저 확인해주는 것이 안전합니다.
- **MSYS2와 MinGW64를 설치한 곳에서 g++.exe 파일이 존재하는 폴더**를 검색하여 찾아 해당 Path를 복사하여 붙여 넣기를 하는 것이 가장 안전한 방법입니다.
- **MSYS2 설치 시에 디폴트를 사용했다면(대부분의 경우), Path는 다음과 같습니다.** 하지만, 다시 한번 확인해보는 것이 안전합니다.
 - **C:\msys64\mingw64\bin**
 - **C:\msys64\usr\bin** (ls, cat, rm 등과 같은 콘솔에서의 commands 사용을 위함)



- **Command Prompt(cmd)** 혹은 **Windows PowerShell**을 실행하여 **"g++ --version"**을 입력해줍니다. 성공적으로 Path가 추가되었다면 다음과 같은 메시지가 콘솔 창에 나타나는 것을 확인할 수 있습니다.

```

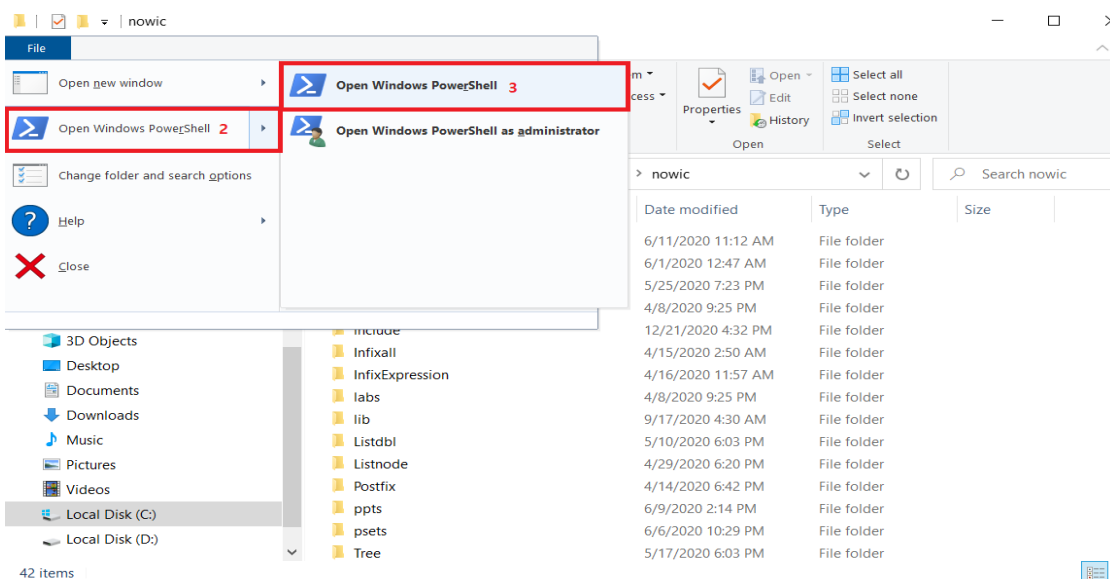
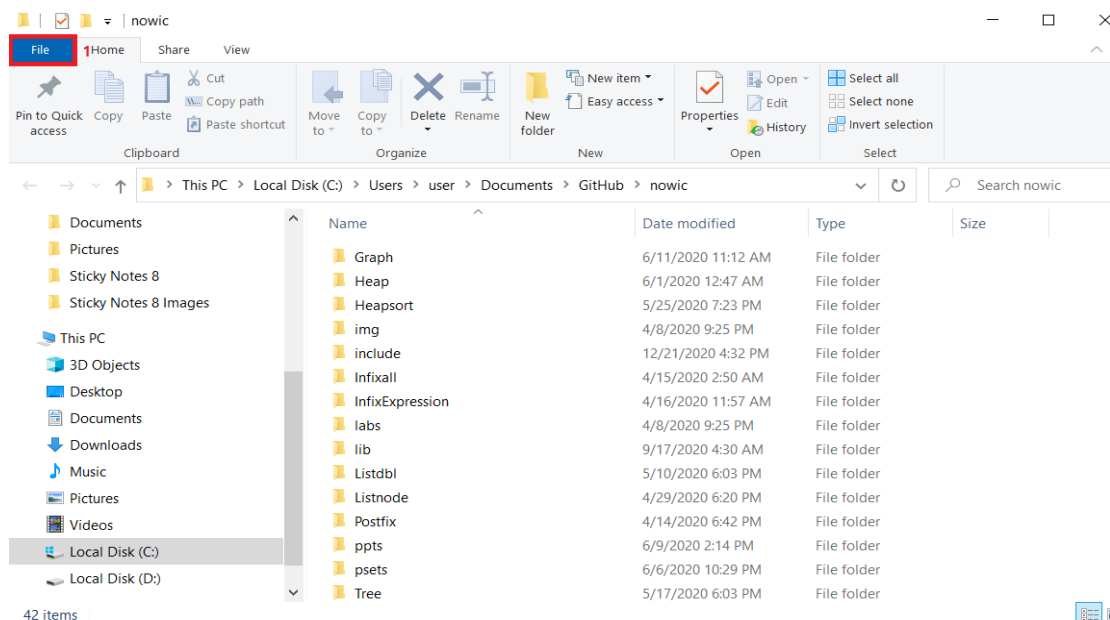
Windows PowerShell
PS C:\Users\user> g++ --version
g++.exe (Rev1, Built by MSYS2 project) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
  
```

5. GCC(g++) 프로그램 테스트(Windows User)

- GCC(g++)는 일종의 플랫폼과도 같습니다. 다른 텍스트 에디터 없이 설치한 g++만을 통해서도 프로그램을 실행할 수 있습니다. **성공적으로 g++이 설치되었음을 확인하기 위해 업데이트 받아온 nowic 폴더 안의 "hello.cpp" 파일을 이용하여 두 개의 간단한 테스트를 진행해보아야 합니다.**

■ Windows PowerShell을 이용한 프로그램 실행 테스트

1. 먼저 **Windows PowerShell**을 실행하여 자신이 업데이트를 받아온 nowic 폴더로 이동합니다. nowic 폴더 좌측 상단의 [File]을 클릭한 뒤에 [Windows PowerShell 열기]를 클릭하면 바로 이동할 수 있습니다.



만약에 [File] 버튼을 찾을 수 없다면 콘솔 창에 "cd [자신의 nowic 폴더 위치]"을 입력하여 이동할 수도 있습니다.

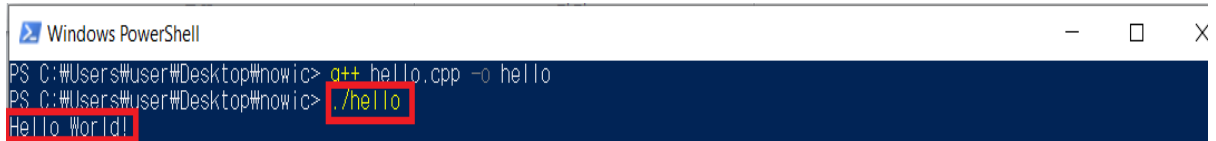
2. Compile을 위해 현재 열려있는 콘솔 창에 "g++ hello.cpp -o hello"을 입력합니다. 이때에 g++ 뒤의 "hello.cpp" 파일은 빌드할 파일을 나타내며, -o 뒤의 "hello"는 자신이 생성할 실행 파일의 이름을 나타냅니다. Compile에 성공했다면 해당 폴더에 "hello.exe"라는 실행 파일(Executable file)이 생성되었음을 확인할 수 있습니다.

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello, World!\n";
    return 0;
}
```

(hello.cpp)

hello.cpp	exe 파일이 생성되었음을 확인할 수	4/15/2020 11:41 PM	CPP File	1 KB
hello.exe	있습니다.	4/8/2020 9:25 PM	Application	55 KB

3. Compile하여 생성한 실행 파일(hello.exe)을 실행하기 위해 콘솔 창에 **“./hello”**을 입력합니다. 이때에 ./ 뒤의 “hello”는 자신이 생성한 실행 파일의 이름을 나타냅니다. 프로그램 실행에 성공했다면 **“Hello World!”**라는 메시지를 콘솔 창을 통해 확인할 수 있습니다.



```
Windows PowerShell
PS C:\Users\user\Desktop\nowic> g++ hello.cpp -o hello
PS C:\Users\user\Desktop\nowic> ./hello
Hello World!
```

■ MSYS2 MINGW64 bash을 이용한 프로그램 실행 테스트

1. MSYS2 MINGW64 bash의 디폴트 주소 값 설정을 위해 자신이 업데이트를 받아온 nowic 폴더 안의 “.bash_profile” 파일을 복사하여 C:\msys64\home\user 폴더 안에 붙여 넣어줍니다. 이때에 **자신의 VSCode 폴더의 Path와 GitHub 폴더의 Path를 반드시 확인하여 변경해주어야 합니다.** 잘못된 Path가 작성되어 있다면 MSYS2 MINGW64 bash가 정상적으로 작동되지 않습니다.

```
alias ls='ls -aGp --color=auto'
alias ll='ls -alkF'
alias rm='rm -i'
alias c='clear'
alias h='history'
alias x='cd /c/GitHub/nowicx'
alias xp='cd /c/GitHub/nowicx/psets'
alias n='cd /c/GitHub/nowic'
alias np='cd /c/GitHub/nowic/psets'

LS_COLORS=$LS_COLORS:no=00:di=36;01'
LS_COLORS=$LS_COLORS:*.h=1;33:*.exe=31:*.o=1;32:*.md=1;33'
export LS_COLORS
export PATH=$PATH:"/c/msys64/mingw64/bin"
export PATH=$PATH:"/c/msys64/usr/bin"
export PATH=$PATH:"/c/Users/$USER/AppData/Local/atom/bin"
export PATH=$PATH:"/c/Users/$USER/AppData/Local/Programs/Microsoft VS Code/bin"
export PATH=$PATH:"/c/Program Files/Git/bin"

echo c:/msys64/home/$USER/.bash_profile
echo $PWD

HOME="/c/GitHub/nowic"
cd $HOME

GREEN="$(tput setaf 2)"
RESET="$(tput sgr0)"
PS1='${GREEN}${pwd -W}> ${RESET}'
```

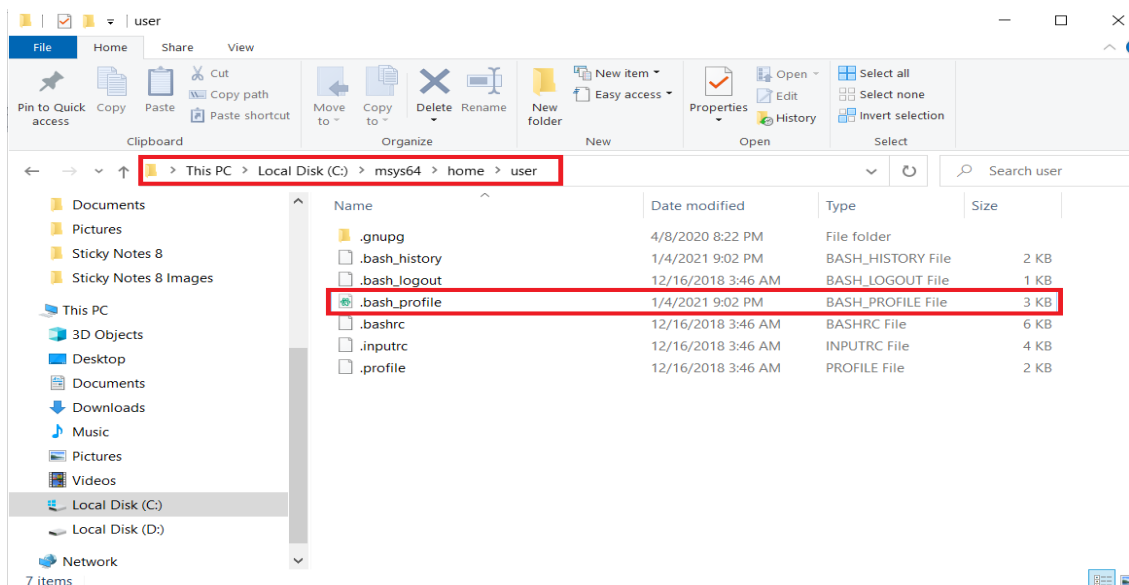
자신의 nowicx 폴더 Path 확인

자신의 nowicx 폴더(작업 폴더) Path 확인

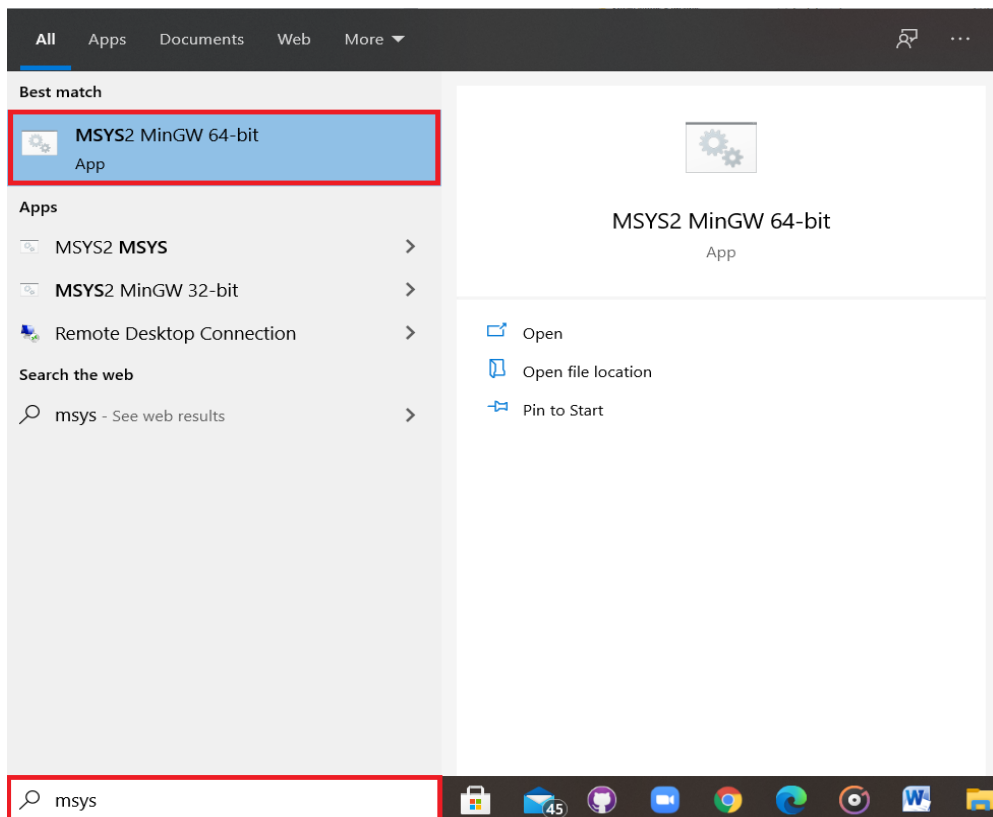
자신의 VSCode가 설치된 Path 확인

자신의 nowicx 폴더 Path 확인

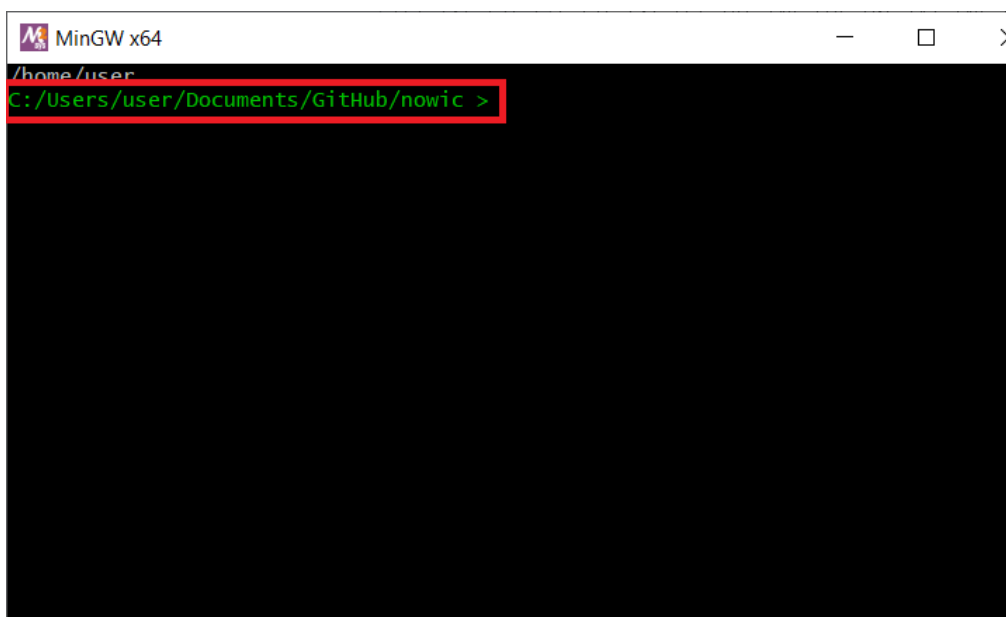
(.bash_profile)



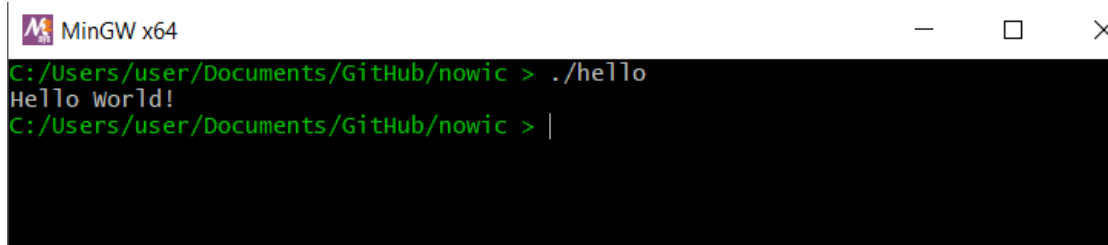
2. Windows 검색 창에 "MSYS"을 검색한 후에 "**MSYS2 MinGW 64-bit**" 프로그램을 실행시켜 줍니다.



3. .bash_profile을 정확히 작성했다면 **MSYS2 MINGW64 bash**의 디폴트 주소 값이 자신의 **nowic** 폴더가 위치한 **Path**로 나타나는 것을 확인할 수 있습니다.



4. Windows PowerShell에서의 테스트 방법과 동일하게 "hello.cpp" 파일을 Compile하여 생성한 실행 파일을 실행해줍니다. 프로그램 실행에 성공했다면 "Hello World!"라는 메시지를 bash를 통해 확인할 수 있습니다.



```
MinGW x64
C:/Users/user/Documents/GitHub/nowic > ./hello
Hello World!
C:/Users/user/Documents/GitHub/nowic > |
```

- 이처럼 g++은 PowerShell뿐만 아니라 bash을 통해서도 실행 가능합니다. PowerShell과 bash에서의 프로그램 테스트가 모두 성공했다면 g++이 성공적으로 설치 및 실행된다는 뜻입니다. 추가적으로 PowerShell과 bash의 차이점에 대해 궁금하다면 아래의 URL을 참고해보면 도움이 될 것입니다.

<<https://m.blog.naver.com/PostView.nhn?blogId=ki630808&logNo=221682450852&proxyReferer=https:%2F%2Fwww.google.com%2F>>