

2019 년 봄학기

CSED232 객체지향프로그래밍

프로그래밍 과제 컴파일 환경 및 설치 관련 문서

최종 업데이트 : 정유철 (ycjung@postech.ac.kr)

2019 년 2 월 20 일

업데이트 #5: 김진유 (jinyou91@postech.ac.kr)

2018 년 9 월 18 일

업데이트 #4: 한상도 (hansd@postech.ac.kr)

2017 년 9 월 13 일

업데이트 #3: 박영덕 (ydpark@postech.ac.kr)

2016 년 9 월 1 일

업데이트 #2: 김해준 (khj809@postech.ac.kr)

2015 년 9 월 5 일

업데이트 #1: 김동은 (deiori@postech.ac.kr)

2014 년 9 월 2 일

작성: 이운성 (dnrjd1212@postech.ac.kr)

2013 년 9 월 2 일

1. MinGW 소개

이번 학기에는 프로그래밍 과제를 위하여 MinGW 를 사용합니다. MinGW 는 Windows 를 위한 Free and Open Source 개발 환경입니다. 이 환경 속에서, 당신은 Windows 개발을 위해 필요한 여러가지 필수적인 도구를 얻을 수 있습니다. 예를 들어, C/C++ 코드를 바이너리 파일로 만들어 줄 컴파일러(g++)와 링커(ld), 필수적인 라이브러리(libstdc++...), 코드 에디터(vim) 등 도구들을 무료로 쉽게 다운받고, 관리할 수 있습니다. MinGW 속에서 Windows 상에서 실행할 수 있는 프로그램을 작성하고, 테스트하고, 배포할 수 있습니다. 이 환경에서 만든 프로그램들은 적절한 컴파일 방법을 따른다면, 다른 컴퓨터에서 MinGW 의 설치 없이 바로 실행시킬 수 있습니다.

MinGW 가 제공하는 개발환경은 gcc, g++, ld, libstdc++ 등 흔히 Linux 라고 불리는 GNU 운영체제에 포함된 소프트웨어를 사용합니다. 따라서 이 환경에서 사용하는 워크플로우는 Ubuntu, Linux Mint 같은 GNU/Linux 용 프로그램 개발 과정과 비슷합니다.

최근에는 MinGW 의 업데이트가 뜸해지고 있기 때문에, 더 활발하게 개발되고, 피드백이 이루어지고 있는 MinGW 의 대체재인 MinGW-w64 를 기준으로 환경 설정을 설명합니다.

2. 설치 과정

(1) Msys2 설치 프로그램 다운로드

Msys2 는 Windows 환경에서 기본 CMD 명령프롬프트를 대체할 다양한 유틸리티를 제공하는 패키지입니다. Msys2 는 MinGW-w64 개발환경 설치와 사용을 위한 도구로 사용되는 터미널입니다. Msys2 위에서 bash, git, ssh, grep, vim 등 여러 편리한 개발용 도구를 설치할 수 있습니다.

다음 주소에서 최신 버전을 받을 수 있습니다. (msys2-x86_64-*.exe 다운로드)

<https://www.msys2.org/>

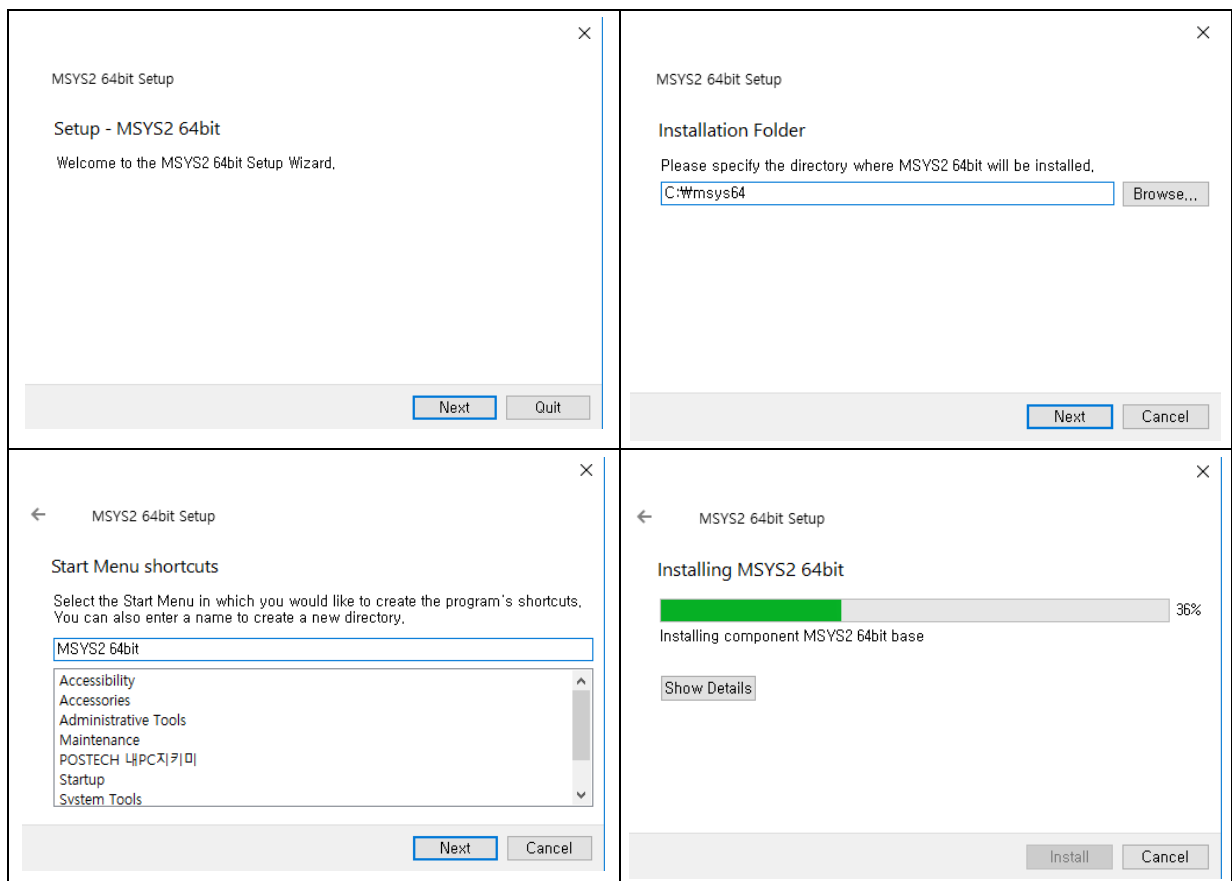


Table 1 : Msys2 설치 과정

(2) MinGW-w64 개발 환경 실행

설치를 완료하면 자동으로 터미널 하나가 실행되는데, 이 터미널의 프롬프트에는 “MSYS”라는 문자열이 붙어 있습니다. 이것은 이 터미널이 MSYS 환경을 위한 개발환경이라는 것을 의미합니다. 우리가 원하는 것은 Windows 환경을 위한 MinGW 개발환경입니다. 이 터미널을 끄고 윈도우 시작 버튼을 눌러 Msys 를 검색해, “MSYS2 MinGW 64-bit”을 선택합니다.

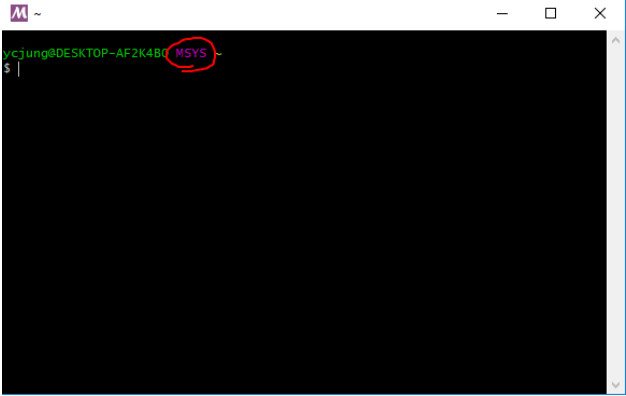
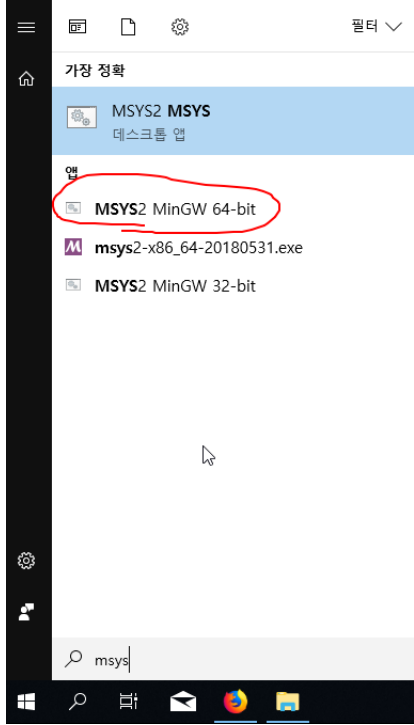
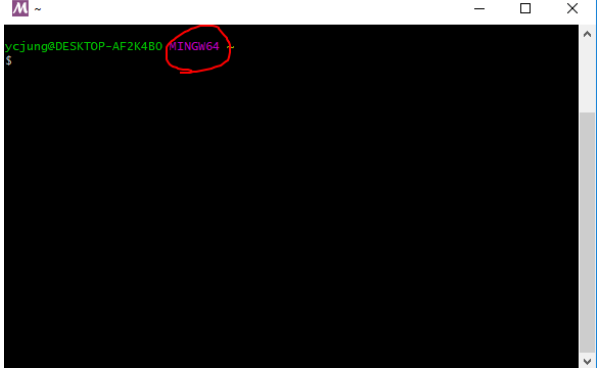
	
<p>설치를 종료하고 나오는 프롬프트에서 MSYS 라는 문자열을 확인할 수 있다. 이 창을 종료한다.</p>	<p>윈도우 시작 화면에서 msys 를 검색하면, “MSYS2 MinGW 64-bit”이라는 앱을 확인할 수 있다. 해당 앱을 실행한다.</p>
	
<p>실행된 터미널에서 MINGW64 문자열을 확인할 수 있다. 이 터미널에서는 Windows 네이티브 어플리케이션을 개발할 수 있다. 앞으로도 이 터미널에서 작업한다.</p>	

Table 2: Msys 실행

(3) MinGW-w64 개발 환경 설정

이제 개발에 필요한 컴파일러, 라이브러리 등 개발 도구를 설치해야 합니다. Msys2 는 소프트웨어를 간단히 찾고, 설치하고, 관리할 수 있는 패키지 매니저인 “pacman”을 내장하고 있습니다. 이 개발환경에서는 대부분의 소프트웨어를 “pacman”을 통해 설치합니다.

pacman 의 커맨드는 다음과 이루어져 있습니다.

pacman <동작> [...]

<동작>은 dash(-) 하나와 대문자 알파벳 하나, 그리고 여러 소문자 알파벳들로 이루어집니다. 대문자 알파벳은 수행하고 싶은 동작의 종류를 나타내고, 소문자 알파벳들은 해당 동작에 대한 옵션을 의미합니다. 다음의 표는 자주 사용되는 pacman 명령어 예시입니다.

명령어	설명
-S : sync 의 약자. 패키지 설치, 업데이트, 탐색 등의 동작을 수행합니다.	
pacman -S [패키지 이름들]	패키지들을 설치합니다.
pacman -Ss [문자열]	설치될 수 있는 패키지를 인덱스로부터 검색합니다.
pacman -Sy	설치될 수 있는 패키지의 인덱스를 업데이트합니다.
pacman -Syu	설치된 모든 패키지를 최신 버전으로 업데이트합니다.
-R : remove 의 약자. 패키지 제거 동작을 수행합니다.	
pacman -R [패키지 이름들]	패키지들을 제거합니다.
pacman -Ru [패키지 이름들]	해당 패키지와 필요하지 않은 추가 의존성들을 제거합니다.
-Q : query 의 약자. 설치된 패키지에 대한 질의를 수행합니다.	
pacman -Qs [문자열]	설치된 패키지를 검색합니다.
공통	
pacman -[동작]h (i.e. pacman -Sh)	해당 동작의 옵션들에 대한 설명을 제공합니다.

Table 3: pacman 의 주요 커맨드

다음은 Pacman 을 이용하여 기본적인 C/C++ 개발 환경을 설치하는 예시입니다. 설치하는 패키지는 Vim (텍스트 에디터), GCC (컴파일러와 필수 라이브러리, 헤더 파일) 입니다.

```
# 소프트웨어의 새 버전이 패키지 저장소에 업로드 되었다면,  
# 새 버전의 소프트웨어들로 인덱스를 업데이트 해야 합니다.
```

```
$ pacman -Sy
```

```
:: 꾸러미 데이터베이스 동기화 중 ...  
mingw32                541.8 KiB   560K/s  00:01 [#####] 100%  
mingw32.sig            119.0  B    0.00B/s  00:00 [#####] 100%  
mingw64                543.0 KiB   2.59M/s  00:00 [#####] 100%  
mingw64.sig            119.0  B    0.00B/s  00:00 [#####] 100%  
msys                   178.7 KiB   87.3M/s  00:00 [#####] 100%  
msys.sig                119.0  B   116K/s  00:00 [#####] 100%
```

```
# 인덱스에 vim 이 있는지 찾아봅니다.(선택 사항)
```

```
$ pacman -Ss vim
```

```
msys/vim 8.1.0877-1 (editors)
```

```
msys/vimpager 2.06-1
```

```
Use ViM as PAGER
```

```
msys/vimpager-git r279.bc5548d-1
```

```
Use ViM as PAGER
```

```
# 인덱스에 vim 이 있다는 것을 확인했습니다. (첫 번째 패키지)
```

```
# vim 이 msys/ 저장소에 위치합니다.
```

```
# 패키지가 msys/ 저장소에 있다는 것은
```

```
# 이 프로그램이 Windows 에서 네이티브하게 실행되는 것이 아니라,
```

```
# Msys 라는 가상 환경의 도움을 받아 작동한다는 것을 의미합니다.
```

```
# 따라서 여기서 설치한 vim 은 이 Msys2 터미널에서만 사용할 수 있습니다.
```

```
#
```

```
# pacman -S 로 패키지를 설치합니다.
```

```
$ pacman -S vim
```

```
의존성 해결 중 ...
```

```
꾸러미 충돌을 찾는 중 ...
```

```
꾸러미 (1) vim-8.1.0877-1
```

```
총 다운로드 크기 : 6.80 MiB
```

```
총 설치 크기 : 43.86 MiB
```

```
:: 설치를 진행하시겠습니까? [Y/n] Y
```

```
:: 꾸러미 가져오는 중 ...
```

```
vim-8.1.0877-1-x86_64      6.8 MiB  1945K/s  00:04 [#####] 100%  
(1/1) 키링의 키를 검사 중 [#####] 100%  
(1/1) 꾸러미 무결성 검사 중 [#####] 100%  
(1/1) 꾸러미 파일 불러오는 중 [#####] 100%  
(1/1) 파일 충돌 검사 중 [#####] 100%  
(1/1) 사용 가능한 디스크 공간 검사 중 [#####] 100%  
:: 꾸러미 변경사항을 처리 중 ...  
(1/1) 설치 중 vim [#####] 100%
```

```
# 컴파일러와 필수 라이브러리를 포함한 GCC 패키지를 설치합니다.
```

```
# vim 과 마찬가지로, 먼저 인덱스에서 GCC 를 검색합니다.(선택사항)
```

```
$ pacman -Ss gcc
```

```

mingw32/mingw-w64-i686-gcc 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (C,C++,OpenMP) for MinGW-w64
mingw32/mingw-w64-i686-gcc-ada 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (Ada) for MinGW-w64
mingw32/mingw-w64-i686-gcc-fortran 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (Fortran) for MinGW-w64
mingw32/mingw-w64-i686-gcc-libgfortran 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (libgfortran) for MinGW-w64
mingw32/mingw-w64-i686-gcc-libs 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (libraries) for MinGW-w64
mingw32/mingw-w64-i686-gcc-objc 7.4.0-1 (mingw-w64-i686-toolchain)
  GNU Compiler Collection (ObjC,Obj-C++) for MinGW-w64
mingw32/mingw-w64-i686-lcov 1.13-2
  front-end for GCC's coverage testing tool gcov
mingw32/mingw-w64-i686-perl 5.28.0-1
  A highly capable, feature-rich programming language (mingw-w64)
mingw64/mingw-w64-x86_64-gcc 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain) ←
  GNU Compiler Collection (C,C++,OpenMP) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-ada 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain)
  GNU Compiler Collection (Ada) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-fortran 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain)
  GNU Compiler Collection (Fortran) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-libgfortran 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain)
  GNU Compiler Collection (libgfortran) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-libs 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain)
  GNU Compiler Collection (libraries) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-objc 8.2.1+20181214-1 (mingw-w64-x86_64-toolchain)
  GNU Compiler Collection (ObjC,Obj-C++) for MinGW-w64
mingw64/mingw-w64-x86_64-lcov 1.13-2
  front-end for GCC's coverage testing tool gcov
mingw64/mingw-w64-x86_64-perl 5.28.0-1
  A highly capable, feature-rich programming language (mingw-w64)
msys/gcc 7.4.0-1 (msys2-devel)
  The GNU Compiler Collection - C and C++ frontends
msys/gcc-fortran 7.4.0-1 (msys2-devel)
  Fortran front-end for GCC
msys/gcc-libs 7.4.0-1 (base) [설치함 : 7.3.0-1]
  Runtime libraries shipped by GCC
msys/mingw-w64-cross-gcc 7.3.0-2 (mingw-w64-cross-toolchain mingw-w64-cross)
  Cross GCC for the MinGW-w64

```

mingw32/ mingw64/ msys/ 저장소에서 여러 검색 결과가 나옵니다.

이 예시에서는 mingw64/ 저장소의 패키지만을 사용합니다.

mingw64/ 저장소의 패키지는 64 비트 Windows 에서 Native 로 작동할 수 있는

소프트웨어 패키지들만을 가지고 있습니다. 이 예시에서는 64 비트 Windows

프로그램을 작성할 것이기 때문에, mingw64/ 저장소에서

GCC 를 설치하는 것이 좋습니다.

#

pacman -S 로 설치합니다.

\$ pacman -S mingw-w64-x86_64-gcc

(설치 과정은 vim 과 같습니다. 생략합니다)

지금까지 텍스트 에디터와 컴파일러, 필수 C/C++라이브러리를 설치했습니다.

다음의 두 커맨드를 실행한 것과 같습니다.

```

$ pacman -Sy
$ pacman -S mingw-w64-x86_64-gcc vim

```

간단한 C++ 프로그램을 컴파일해보고, 개발 환경이 제대로 설치되었다는 것을
확인합니다. Hello world 프로그램을 작성합니다.

#

아래의 커맨드는 Hello world 프로그램을 담은 hello.cpp 라는
파일을 생성합니다.

```
$ echo '  
#include<iostream>  
int main(){  
std::cout << "Hello" << std::endl;  
return 0;  
' > hello.cpp
```

'ls' 커맨드를 사용하면 현재 폴더에 있는 파일들을 표시합니다.

hello.cpp 가 생긴 것을 확인합니다. (선택 사항)

```
$ ls  
hello.cpp
```

GCC 를 이용해 컴파일합니다.

C++ 소스 코드는 'g++'이라는 프로그램을 사용하여 컴파일 합니다.

아래의 커맨드는 hello.cpp 소스 코드를 a.exe 실행파일로 컴파일 합니다.

```
$ g++ hello.cpp
```

다시 'ls' 하면 a.exe 가 생성된 것을 확인할 수 있습니다.

```
$ ls  
a.exe  hello.cpp
```

a.exe 를 실행하면 프로그램이 작동하는 것을 확인할 수 있습니다.

```
$ ./a.exe  
Hello
```

만약 C++가 아닌 C 코드를 컴파일해야 한다면,

'g++' 대신에 'gcc'를 사용하면 됩니다.

```
$ echo '  
#include<stdio.h>  
int main(){  
printf("Hello\n");  
return 0;  
' > hello.c  
$ gcc hello.c  
$ ./a.exe  
Hello  
$
```


3. 참고 사항

(1) 유용한 Msys2 콘솔 명령어

명령어	설명
pwd	현재 경로를 출력
ls [옵션] [파일명]	현재 경로에서 하위 디렉토리 및 파일을 표시
cd [디렉토리]	경로 이동 명령어로 “.”은 현재 디렉토리 “..”은 상위 디렉토리를 의미
cp [Source] [Destination]	Source 파일을 Destination 의 디렉토리 or 파일명으로 복사
mv [Source] [Destination]	Source 파일을 Destination 의 디렉토리 or 파일명으로 이동
mkdir [옵션] [디렉토리명]	디렉토리를 생성
rm [옵션] [파일 or 디렉토리]	파일이나 디렉토리를 제거, 해당 디렉토리의 모든 하위 디렉토리 및 파일을 제거하고 싶다면 -rf 옵션을 주면 됨 “rm -rf [디렉토리명]”
find [옵션] [파일명 or 디렉토리]	찾고자 하는 파일 혹은 디렉토리의 경로를 보여줌 (-name 옵션이 주로 쓰임 - 파일의 이름으로 검색함)
grep [옵션] [패턴] [파일명]	파일 내부에 해당 패턴을 검색해서 찾아줌 (파일명을 *로 할 경우, 모든 파일에 해당하는 패턴을 찾아줌) (예) <code>grep -r -n -I “Hello world” *</code> -> 현재 디렉토리의 하위 파일들을 대상으로 Hello world 가 속해있는 파일들의 경로를 보여줌. -> [-I 옵션] 대소문자 무시 / [-n 옵션] 해당 파일에서의 line number 를 표시 / [-r 옵션] 현재 디렉토리의 하위 디렉토리까지 모두 포함하여 찾음
cat [파일명]	파일 내용을 출력할 때 사용
exit	종료

Table 4: Msys2 콘솔 명령어

(2) 유용한 패키지

패키지 이름	설명
zip	파일/폴더를 압축한다. “zip -r 파일명.zip 폴더/” 과제가 소스 코드 여러 개로 되어 있을 때 유용하다.
unzip	.zip 압축 파일을 해제한다. “unzip 파일명.zip”

Table 5: Msys2 유용한 패키지

(3) 텍스트 에디터

MinGW 상에서 사용하기 좋은 텍스트 데이터들을 소개합니다. 텍스트 에디터의 선택은 자유입니다. Msys2 밖에서 직접 C:\msys64\home\<유저이름>\ 에 들어가서 Windows 용 텍스트 에디터를 사용해도 상관 없습니다.

에디터 이름	설명
vim	‘vim’ 패키지로 설치 가능. 패키지 설치 후 ‘vimtutor’를 실행해 기본적인 사용법을 익힐 수 있으니 참고 바람.
nano	‘nano’ 패키지로 설치 가능 간단한 UI 를 제공하여 처음 접하는 사람도 쉽게 텍스트를 편집 가능.
gedit	‘mingw-w64-x86_64-gedit’ 패키지로 설치 가능 메모장과 유사한 그래픽 인터페이스를 제

Table 5: 에디터

(4) 기타 팁

- Msys2 상의 기본 작업 공간은 Windows 파일 시스템의 C:\msys64\home\<유저이름>\ 에서 찾아볼 수 있습니다. 과제 제출 시에 이 경로에서 자신의 소스코드와 파일을 가져오면 됩니다.