



시스템 프로그래밍5

S-개발자 4기 2026-03-11(수)

서울 송파구 동남로 130, 2층 제 4강의실

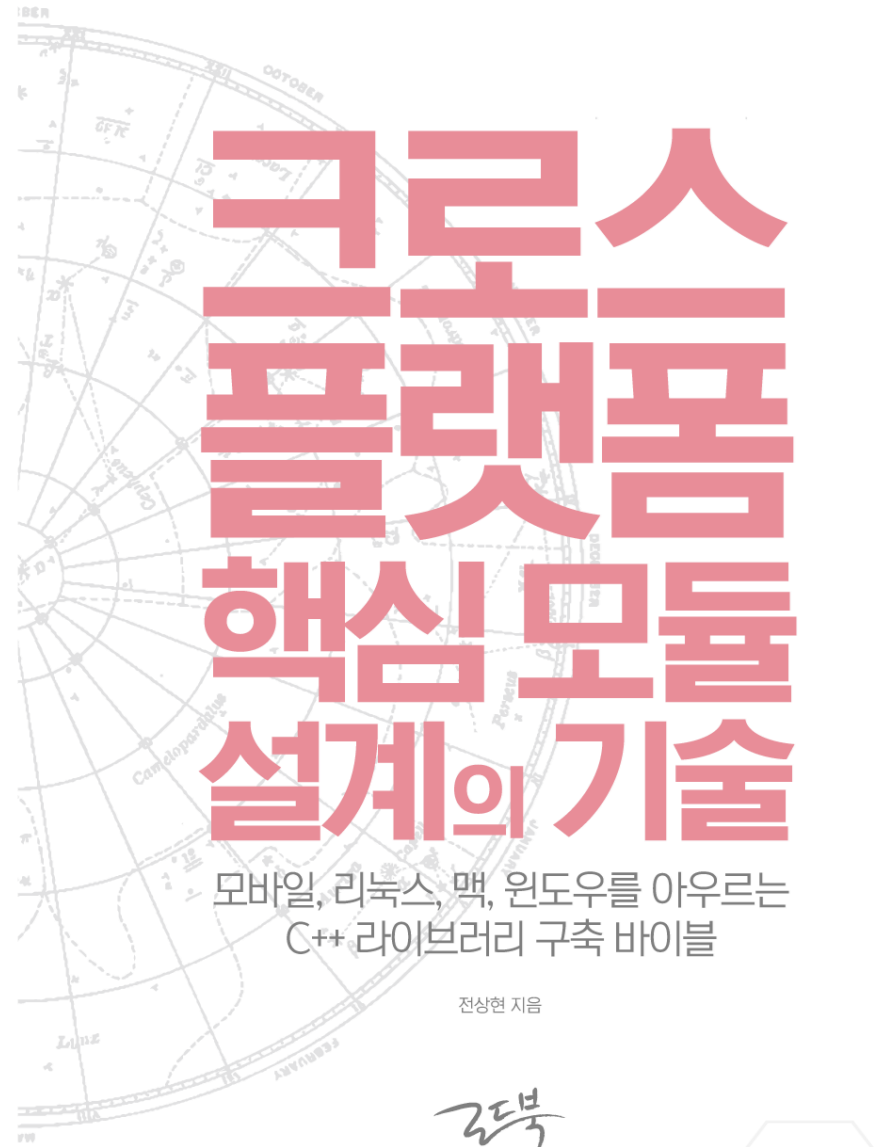


대표이사 전상현

크로스 플랫폼 핵심 모듈 설계의 기술

모바일, 리눅스, 맥, 윈도우를 아우르는 C++ 라이브러리 구축 바이블

전상현 지음



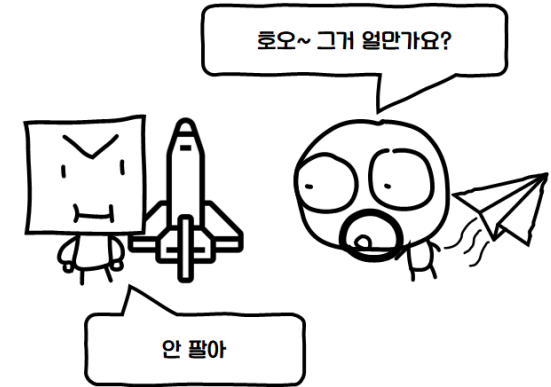
크로스 플랫폼 핵심 모듈 설계의 기술

모바일, 리눅스, 맥, 윈도우를 아우르는 C++ 라이브러리 구축 바이블

전상현 지음



과정명	시스템 프로그래밍		강사명	전상현
강의시간	24시간 (4일 x 6시간)			
강의목표	컴퓨터 시스템을 이해한다. 나아가 다양한 시스템을 제어하는 C++ 프로그래밍 기술을 익힌다.			
평가방식	시험 50%, 실습 50%			
강의내용				
시간(H)	제목	내용		
1H	오리엔테이션	강사 소개 후 간단한 퀴즈와 함께 실행파일의 구조를 이해한다.		
2H	변수형과 유니코드	시스템에 존재하는 모든 종류의 변수형을 알아본다. 다국어를 표현하는 문자체계인 유니코드를 이해한다.		
3H	개발환경 구축하기	윈도우/리눅스(wsl)에 빌드할 수 있는 환경을 구성한다. 기본적인 개발툴 사용법을 익힌다.		
2H	문자열 정복하기	C++과 친해지기 위한 코드를 작성해본다. VisualStudio 디버깅 기술을 배운다.		
2H	C++ 컴파일러/링커/디버거 정복하기	유니코드 상호 변환하는 함수를 이해한다. 문자열을 자유자재로 다루는 방법을 배운다.		
2H	파일시스템 정복하기	플랫폼별 파일/디렉토리 접근 함수를 이해한다. 파일 시스템을 자유자재로 다루는 방법을 배운다.		
2H	데이터 정복하기	STL 자료구조를 이해하고, 실무 응용 기술을 배운다. XML/JSON/INI 등을 자유자재로 다루는 방법을 배운다.		
2H	메모리 정복하기	C++의 꽃, 스택 메모리의 장단점을 알아본다. 4가지 영역의 메모리를 자유자재로 활용한다.		
2H	소켓 정복하기	UDP/TCP 통신, DATAGRAM/STREAM의 차이를 이해한다. 소켓 통신 시스템을 자유자재로 다루는 방법을 배운다.		
6H	최종실습	앞에서 배운 지식과 기술을 활용하여 미니 프로젝트를 진행한다.		



코딩 실력이 곧 우주선이다.
우주같이 광활한 컴퓨터 세계로 여행을 떠나자.



<참고서적>

크로스플랫폼 핵심모듈 설계의 기술(2018) – 전상현, 로드북
아무도 알려주지 않은 C++ 코딩의 기술 (2023) – 전상현, 로드북

긴 코드 작성의 기술

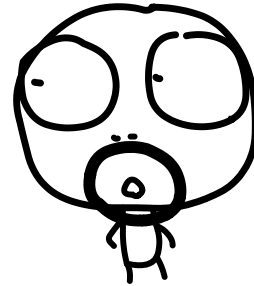
개발자들의 끊이지 않는 논쟁

- 함수의 시작을 새로운 라인에 쓸 것인가 아닌가

```
void SomeFunc(void) {  
    블라블라...  
}
```

```
void SomeFunc(void)  
{  
    블라블라...  
}
```

무조건 왼쪽 아닌 가유~?



퀴즈. 위 둘 중 어떤 것이 정답일까?

답은 생각보다 아주 쉽다.(수업시간에 공개)



다음 코드의 문제점이 보이시나요?

```
DWORD value;
value = -1;
If( value < 0 ) {
    중요한 로직...
}
```

C++은 헝가리안 표기법을 지키자.

- 변수 타입을 혼동하면 알고리즘이 무너진다.
- 나보다는 제3개발자(미래의 나 포함)를 위한 것이다.

변수타입을 놓쳤을 때의 문제 상황 예시

- 중요한 로직에 진입할 수 없다.

접두어	데이터 타입
b	byte, boolean
n	int, short
i	int, short (주로 인덱스로 사용)
c	int, short (주로 크기로 사용)
l	long
f	float
d, db	double
ld	long double
w	word
dw	double word
qw	quad word
ch	char
sz	NULL로 끝나는 문자열
str	C++ 문자열
arr	배열 (문자열 제외): 다른 접두어와 조합 가능
p	포인터 (16비트, 32비트): 다른 접두어와 조합 가능
lp	포인터 (32비트, 64비트): 다른 접두어와 조합 가능
psz	NULL로 끝나는 문자열을 가리키는 포인터 (16비트, 32비트)
lpsz	NULL로 끝나는 문자열을 가리키는 포인터 (32비트 ^[2] , 64비트)
fn	함수 타입
PFN	함수 포인터 (16비트, 32비트)
LPFN	함수 포인터 (64비트)



내 모니터로 한 번에 볼 수 있을 양이 가장 좋다.

- 함수가 길어져서 스크롤 해야만 끝까지 볼 수 있다면 버그 확률이 높아진다.

함수가 길어지면?

- 더 작은 함수들로 소분한다.
- Main 함수는 책의 목차처럼 한번에 전체 내용을 가늠할 수 있어야 한다.

```
int main(int argc, char* argv[])
{
    if (argc < 2)
    {
        printf("<USAGE> %s [config-file]\n", argv[0]);
        return -1;
    }

    ST_CONFIG config = LoadConfigureFrom(argv[1]);
    InitLog(config.logfile);
    StartUpServer(config.server.port, config.server.maxconnection);
    WaitForTerminateSignal();
    ShutDownServer();
    FinalizeLog();
    return 0;
}
```



사람의 눈은 위에서 아래로, 왼쪽에서 오른쪽으로 익숙해져 있다.

- 따라서 왼쪽과 같은 코드 보다는 오른쪽 방식이 읽기 더 편하다.
- 앞으로 부등호는 $<$, $<=$ 만 사용하는 것으로 약속하자.

```
if (a > 10 && a < 43)
{
}
```

```
if (10 < a && a < 43)
{
}
```

퀴즈, 영문표기 나는 어떤 부등호일까?

- 1) $<$
- 2) $>$
- 3) $<=$
- 4) $>=$



들여쓰기 끝없이 들어가는 코드가 보기 좋을까?

가독성을 위해 if(true) 보다는 if(false)에 주목해보자.

```
if (파일이 열렸으면) {  
    if (파일에 값을 쓰는데 성공하면) {  
        printf("Succeed!!\n")  
    }  
    else  
        printf("Failed to write File\n");  
}  
else  
    printf("Failed to open File\n");
```

```
if (파일이 안 열렸으면) {  
    printf("Failed to open File\n");  
    return;  
}  
  
if (파일에 값을 쓰지 못했다면) {  
    printf("Failed to write File\n");  
    return;  
}  
printf("Succeed!!\n")
```



1. 변수는 명사로 짓는다.(동명사 포함)
2. 함수는 동사(지시어)로 짓는다.
3. 클래스나 구조체는 기능이나 역할을 나타내는 명사로 짓는다.

```
struct ST_SCRAPPED_FILES
{
    std::vector<std::tstring> files;
};

int nMaxCount = 10;
ST_SCRAPPED_FILES scrapped = ScrapFiles(TEXT("d:/repo/*"), nMaxCount);
```

파일제어 함수군



fgetc	findnext	fputs	fseek
fgetchar	fopen	fread	fsetpos
fgetpos	fprintf	freopen	fstat
fgets	fputc	frexp	ftell
findfirst	fputchar	fscanf	ftime
			fwrite

```
FILE* fopen(_In_z_ const char * _Filename, _In_z_ const char * _Mode);
FILE* _wopen(_In_z_ const wchar_t * _Filename, _In_z_ const wchar_t * _
Mode);
```

여기서 잠깐, `_wopen` 함수의 필요성. 다음 코드의 실행 결과를 예측해봅시다.

```
wchar_t* pszUnicodeFilePath = L"c:\\한글경로\\한글파일.txt";
std::string strFilePath = MBSFromWCS(pszUnicodeFilePath);
FILE* pFile = fopen(strFilePath.c_str(), "rb");
```

한글 윈도우

영문 윈도우



어떤 함수가 더 쓰기에 편해 보입니까??

맥/리눅스 함수

```
int open(const char *pathname, int flags, mode_t mode);
```

윈도우 함수

```
HANDLE CreateFileA(  
    __in    LPCSTR lpFileName,  
    __in    DWORD dwDesiredAccess,  
    __in    DWORD dwShareMode,  
    __in_opt LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
    __in    DWORD dwCreationDisposition,  
    __in    DWORD dwFlagsAndAttributes,  
    __in_opt HANDLE hTemplateFile  
);
```



```
HANDLE CreateFileA(
    __in    LPCSTR lpFileName,
    __in    DWORD dwDesiredAccess,
    __in    DWORD dwShareMode,
    __in_opt LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    __in    DWORD dwCreationDisposition,
    __in    DWORD dwFlagsAndAttributes,
    __in_opt HANDLE hTemplateFile
);
```

dwDesiredAccess

- GENERIC_READ
- GENERIC_WRITE
- FILE_APPEND_DATA

dwShareMode

- FILE_SHARE_READ
- FILE_SHARE_WRITE
- FILE_SHARE_DELETE

dwCreationDisposition

옵션	새로 생성하는 경우	존재하는 파일인 경우
CREATE_NEW	성공	실패
CREATE_ALWAYS	성공	성공
OPEN_EXISTING	실패	성공
OPEN_ALWAYS	성공	성공
TRUNCATE_EXISTING	실패	성공

dwFlagsAndAttributes

- FILE_ATTRIBUTE_READONLY
- FILE_ATTRIBUTE_HIDDEN
- FILE_ATTRIBUTE_SYSTEM
- FILE_ATTRIBUTE_DIRECTORY
- FILE_ATTRIBUTE_ARCHIVE
- FILE_ATTRIBUTE_DEVICE
- FILE_ATTRIBUTE_NORMAL
- FILE_ATTRIBUTE_TEMPORARY
- FILE_ATTRIBUTE_SPARSE_FILE
- FILE_ATTRIBUTE_REPARSE_POINT
- FILE_ATTRIBUTE_COMPRESSED
- FILE_ATTRIBUTE_OFFLINE
- FILE_ATTRIBUTE_NOT_CONTENT_INDEXED
- FILE_ATTRIBUTE_ENCRYPTED
- FILE_ATTRIBUTE_VIRTUAL

오~ 이걸 만만해 보이는데요?
역시 리눅스가 짱이야~

```
int open(const char *pathname, int flags, mode_t mode);
```

멍충아, 전투력이 안 느껴져?
이게 더 어려운 거란다.
모르겠으면 다음 장을 보도록 해!





```
int open(const char *pathname, int flags, mode_t mode);
```

윈도우의 dwDesiredAccess 옵션	리눅스의 flags 옵션
GENERIC_READ	O_RDONLY
	GENERIC_WRITE
GENERIC_READ	GENERIC_WRITE
FILE_APPEND_DATA	O_APPEND O_WRONLY

윈도우의 nDisposition 옵션	리눅스의 flags 옵션	
CREATE_NEW	O_CREAT	O_EXCL
CREATE_ALWAYS	O_CREAT	O_TRUNC
OPEN_EXISTING	O_EXCL	
OPEN_ALWAYS	O_CREAT	
TRUNCATE_EXISTING	O_TRUNC	O_EXCL

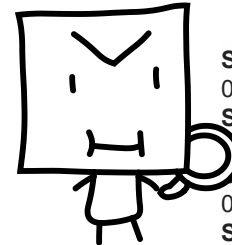
윈도우의 dwAttributes	리눅스의 flags
FILE_ATTRIBUTE_DIRECTORY	O_DIRECTORY
FILE_ATTRIBUTE_TEMPORARY	O_TMPFILE

리눅스의 flags 옵션	설명
O_CLOEXEC	fork 후 프로세스를 실행 시, 기존에 열어둔 파일 디스크립터들을 닫음
O_LARGEFILE	4GB 이상의 파일을 접근 가능하도록 함



파이썬 하겠습니다,
식빵 선생님~

mode



하나하나 잘 들여다 보면
이해할 수 있을 거란다!

- S_IRWXU**
00700 모드로 파일 소유자에게 읽기, 쓰기, 쓰기 실행권한을 준다.
- S_IRUSR**
0400 으로 사용자에게 읽기 권한을 준다.
- S_IWUSR**
00200 으로 사용자에게 쓰기 권한을 준다.
- S_IXUSR**
00100 으로 사용자에게 실행 권한을 준다.
- S_IRWXG**
00070 으로 그룹에게 실행 권한을 준다.
- S_IRWXO**
00007 으로 기타 사용자 에게 읽기, 쓰기, 실행 권한을 준다.
- S_IROTH**
00004 으로 기타 사용자 에게 읽기 권한을 준다.
- S_IWOTH**
00002 으로 기타 사용자 에게 쓰기 권한을 준다.
- S_IXOTH**
00001 으로 기타 사용자 에게 실행 권한을 준다.



```
HANDLE CreateFileA(const char* lpFileName, DWORD dwDesiredAccess, E_FILE_
DISPOSITION nDisposition, DWORD dwAttributes, DWORD dwMode = 0777, HANDLE
hTemplateFile = NULL);
HANDLE CreateFileW(const wchar_t* lpFileName, DWORD dwDesiredAccess, E_
FILE_DISPOSITION nDisposition, DWORD dwAttributes, DWORD dwMode = 0777,
HANDLE hTemplateFile = NULL);
```

```
enum E_FILE_DESIRED_ACCESS
```

```
{
    GENERIC_READ_           = 0x80000000,
    GENERIC_WRITE_          = 0x40000000,
    FILE_APPEND_DATA_       = 0x0004
};
```

```
enum E_FILE_DISPOSITION
```

```
{
    CREATE_NEW_              = 1,
    CREATE_ALWAYS_           = 2,
    OPEN_EXISTING_           = 3,
    OPEN_ALWAYS_             = 4,
    TRUNCATE_EXISTING_       = 5,
};
```

```
enum E_FILE_ATTRIBUTE
```

```
{
    FILE_ATTRIBUTE_READONLY_ = 0x00000001,
    FILE_ATTRIBUTE_HIDDEN_   = 0x00000002,
    FILE_ATTRIBUTE_SYSTEM_   = 0x00000004,
```

윈도우 정의와 충돌을 피하기 위해
마지막에 언더바를 넣음

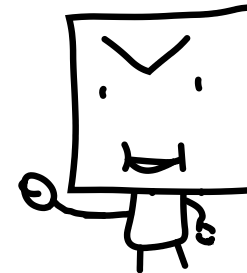
```
FILE_ATTRIBUTE_DIRECTORY_ = 0x00000010,
FILE_ATTRIBUTE_ARCHIVE_   = 0x00000020,
FILE_ATTRIBUTE_DEVICE_    = 0x00000040,
FILE_ATTRIBUTE_NORMAL_    = 0x00000080,
FILE_ATTRIBUTE_TEMPORARY_ = 0x00000100,
FILE_ATTRIBUTE_SPARSE_FILE_ = 0x00000200,
FILE_ATTRIBUTE_REPARSE_POINT_ = 0x00000400,
FILE_ATTRIBUTE_COMPRESSED_ = 0x00000800,
FILE_ATTRIBUTE_OFFLINE_   = 0x00001000,
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED_ = 0x00002000,
FILE_ATTRIBUTE_ENCRYPTED_  = 0x00004000,
FILE_ATTRIBUTE_VIRTUAL_   = 0x00010000,
};
```

```
enum E_FILE_MOVE_METHOD
{
    FILE_BEGIN_           = 0,
    FILE_CURRENT_        = 1,
    FILE_END_             = 2,
};
```

```
bool ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead,
LPDWORD lpNumberOfBytesRead);
bool WriteFile(HANDLE hFile, LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite,
LPDWORD lpNumberOfBytesWritten);
ECODE SetFilePointer(HANDLE hFile, int64_t nDistanceToMove, E_FILE_MOVE_
METHOD nMoveMethod, int64_t* pnNewFilePosition = NULL);
QWORD GetFileSize(HANDLE hFile);
void CloseFile(HANDLE hFile);
```

GetFilePointer가 없는 이유는
SetFilePointer 함수가
대체할 수 있기 때문이지!

```
int64_t nFilePosition = 0;
SetFilePointer(hFile, 0, FILE_CURRENT_, &nFilePosition);
```

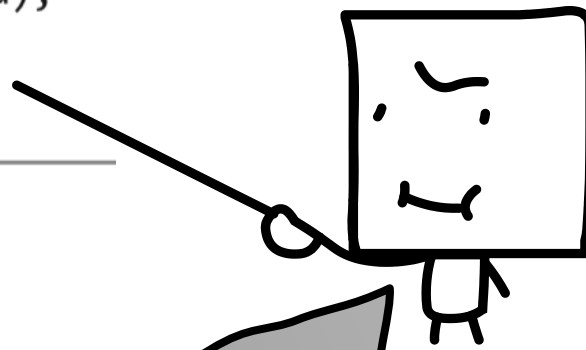


디렉토리 함수군



```
std::string GetCurrentDirectoryA(void);  
std::wstring GetCurrentDirectoryW(void);  
bool        SetCurrentDirectory(const char* pszNewPath);  
bool        SetCurrentDirectory(const wchar_t* pszNewPath);
```

```
std::string GetSystemDirectoryA(void);  
std::wstring GetSystemDirectoryW(void);  
std::string GetTempPathA(void);  
std::wstring GetTempPathW(void);
```



다른 함수들은 Directory인데, 왜 애만 Path이지?
이미 이렇게 세상에 나온 이상 어쩔 수 없습니다.
처음에 그렇게 만든 MS 개발자를 탓해야 합니다.
(cppcore는 윈도우 API를 그대로 다시 만든 것일 뿐!)



```
bool    CreateDirectory(const char* pszPath);
bool    CreateDirectory(const wchar_t* pszPath);
bool    RemoveDirectory(const char* pszPath);
bool    RemoveDirectory(const wchar_t* pszPath);

bool    PathFileExists(const char* pszExistFile);
bool    PathFileExists(const wchar_t* pszExistFile);
bool    CopyFile(const char* pszExistFile, const char* pszNewFile, BOOL
bFailIfExist);
bool    CopyFile(const wchar_t* pszExistFile, const wchar_t* pszNewFile,
BOOL bFailIfExist);
bool    MoveFile(const char* pszExistFile, const char* pszNewFile);
bool    MoveFile(const wchar_t* pszExistFile, const wchar_t* pszNewFile);
bool    DeleteFile(const char* pszFileName);
bool    DeleteFile(const wchar_t* pszFileName);
```

참고로 리눅스에는 CopyFile이 없습니다. 왜냐? 퀴즈예요.
MoveFile은 실패 가능성이 높으므로 신중히 사용합시다.
실패하면 복사 후 삭제로 2차 대비를 해둬야 해요.



```

HANDLE FindFirstFile(const char* pszFilePattern, LPWIN32_FIND_DATAA
pFindData);
HANDLE FindFirstFile(const wchar_t* pszFilePattern, LPWIN32_FIND_DATAW
pFindData);
bool FindNextFile(HANDLE hFindFile, LPWIN32_FIND_DATAA pFindData);
bool FindNextFile(HANDLE hFindFile, LPWIN32_FIND_DATAW pFindData);
void FindClose(HANDLE hFindFile);
    
```

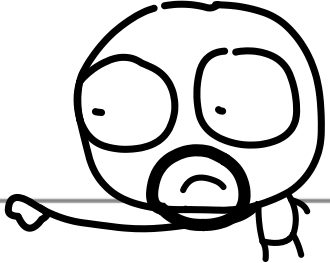
Do-while문으로 구현하기에
최적화된 API이다.
윈도우/맥/리눅스 공통!!

```

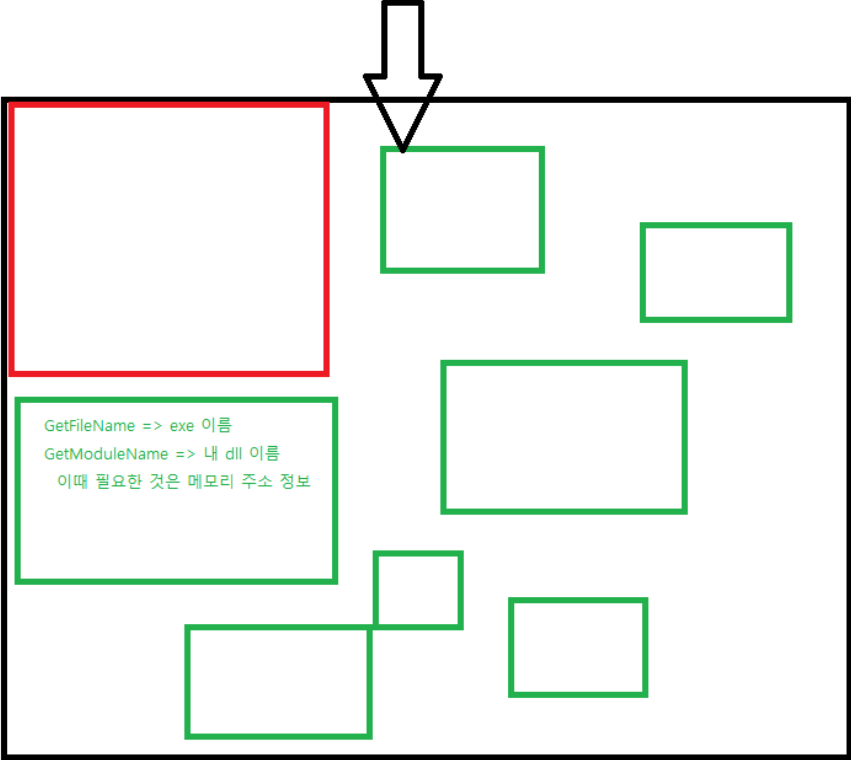
1 #include "stdafx.h"
2
3 void TraverseFileSystem(std::tstring strTargetDir, std::tstring strPattern)
4 {
5     std::tstring strTargetPattern = strTargetDir + TEXT("/") + strPattern;
6
7     ST_FILE_FINDDATA stFindData;
8     HANDLE hFile = FindFirstFile(strTargetPattern.c_str(), &stFindData);
9     if (NULL == hFile)
10        return;
11
12    do
13    {
14        if (stFindData.bIsDirectory && TEXT(".") == stFindData.strFileName)
15            continue;
16        if (stFindData.bIsDirectory && TEXT("..") == stFindData.strFileName)
17            continue;
18        if (stFindData.bIsDirectory)
19            TraverseFileSystem(strTargetDir + TEXT("/") + stFindData.strFileName, strPattern);
20
21        tprintf(TEXT("%s\n"), stFindData.strFileName.c_str());
22    } while (FindNextFile(hFile, &stFindData));
23    FindClose(hFile);
24 }
25
26
27 int main(void)
28 {
29     TraverseFileSystem(TEXT("/"), TEXT("*.dat"));
30     return 0;
31 }
    
```



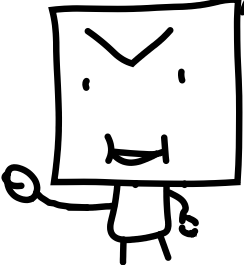
식빵선생님!
DLL 모듈명은 왜 핸들이 필요해유?



```
std::string GetFileNameA(void);
std::wstring GetFileNameW(void);
std::string GetModuleFileNameA(HANDLE hModule);
std::wstring GetModuleFileNameW(HANDLE hModule);
```



로드된 DLL 메모리 시작점이 있어야
어떤 DLL을 가리키는지
알 수 있기 때문이지!





Process Explorer - Sysinternals: www.sysinternals.com [WIN-QDFHA9A0RIE#profrog]

File Options View Process Find DLL Users Help

Process	CPU	Private Byt...	Working Set	PID	Description	Company Name
CrossEXService.exe	0,10	2,048 K	8,904 K	12492	CrossEX Service	iniLINE Co., Ltd.
TGitCache.exe	0,02	5,984 K	13,996 K	10836	TortoiseGit status cache	https://tortoisegit.org/
WINWORD.EXE		119,976 K	180,868 K	13236	Microsoft Word	Microsoft Corporation
devenv.exe	0,39	125,824 K	151,724 K	9280	Microsoft Visual Studio ...	Microsoft Corporation
SystemTestA.exe		2,152 K	7,100 K	9964	SystemTest 응용 프로그램	

Name	Description	Company Name	Path
advapi32.dll	고급 Windows 32 기반 API	Microsoft Corporation	C:\Windows\SysWOW64\advapi32.dll
bcryptprimitives.dll	Windows Cryptographic Primi...	Microsoft Corporation	C:\Windows\SysWOW64\bcryptprimitives.dll
combase.dll	Windows용 Microsoft COM	Microsoft Corporation	C:\Windows\SysWOW64\combase.dll
cryptbase.dll	Base cryptographic API DLL	Microsoft Corporation	C:\Windows\SysWOW64\cryptbase.dll
DummyDynamicL...			C:\GIT\cppcore\Build\Test\DummyDynamicLi...
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\gdi32.dll
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\gdi32full.dll
imm32.dll	Multi-User Windows IMM32 A...	Microsoft Corporation	C:\Windows\SysWOW64\imm32.dll
kernel32.dll	Windows NT 기반 API 클라이...	Microsoft Corporation	C:\Windows\SysWOW64\kernel32.dll
KernelBase.dll	Windows NT 기반 API 클라이...	Microsoft Corporation	C:\Windows\SysWOW64\KernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
msvcp_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\SysWOW64\msvcp_win.dll
msvcp90d.dll	Microsoft® C++ Runtime Library	Microsoft Corporation	C:\Windows\WinSxS\x86_microsoft.vc90.debug...
msvcr90d.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\WinSxS\x86_microsoft.vc90.debug...
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\SysWOW64\msvcrt.dll
ntdll.dll	NT 계층 DLL	Microsoft Corporation	C:\Windows\SysWOW64\ntdll.dll
ntdll.dll	NT 계층 DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
ole32.dll	Windows용 Microsoft OLE	Microsoft Corporation	C:\Windows\SysWOW64\ole32.dll
psapi.dll	Process Status Helper	Microsoft Corporation	C:\Windows\SysWOW64\psapi.dll
rpcrt4.dll	원격 프로시저 호출 런타임	Microsoft Corporation	C:\Windows\SysWOW64\rpcrt4.dll
sechost.dll	Host for SCM/SDDL/LSA Loo...	Microsoft Corporation	C:\Windows\SysWOW64\sechost.dll
shlwapi.dll	셸 표준 이하 유틸리티 라이브러리	Microsoft Corporation	C:\Windows\SysWOW64\shlwapi.dll
SortDefault.nls			C:\Windows\Globalization\Sorting\SortDefault...
sspicli.dll	Security Support Provider Int...	Microsoft Corporation	C:\Windows\SysWOW64\sspicli.dll
SystemTestA.exe	SystemTest 응용 프로그램		C:\GIT\cppcore\Build\Test\SystemTestA.exe
ucrtbase.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\SysWOW64\ucrtbase.dll
user32.dll	다중 사용자 Windows 사용자 A...	Microsoft Corporation	C:\Windows\SysWOW64\user32.dll
version.dll	Version Checking and File In...	Microsoft Corporation	C:\Windows\SysWOW64\version.dll
win32u.dll	Win32u	Microsoft Corporation	C:\Windows\SysWOW64\win32u.dll
wow64.dll	Win32 Emulation on NT64	Microsoft Corporation	C:\Windows\System32\wow64.dll
wow64cpu.dll	AMD64 Wow64 CPU	Microsoft Corporation	C:\Windows\System32\wow64cpu.dll
wow64win.dll	Wow64 Console and Win32 A...	Microsoft Corporation	C:\Windows\System32\wow64win.dll

CPU Usage: 22.53% Commit Charge: 43.46% Processes: 152 Physical Usage: 43.32%

▲ 하나의 프로세스에 로드된 많은 수의 동적 라이브러리



Process	CPU	Private Byt...	Working Set	PID	Description	Company Name
CrossEXService.exe	0,12	2,120 K	8,932 K	12492	CrossEX Service	iniLINE Co., Ltd.
TGitCache.exe		5,984 K	14,296 K	10836	TortoiseGit status cache	https://tortoisegit.org/
WINWORD.EXE	< 0,01	85,492 K	150,932 K	13236	Microsoft Word	Microsoft Corporation
devenv.exe	0,40	126,028 K	151,768 K	9280	Microsoft Visual Studio ...	Microsoft Corporation
SystemTestA.exe		2,152 K	7,100 K	9964	SystemTest 응용 프로그램	
conhost.exe		6,000 K	18,660 K	9558	Console Window Host	Microsoft Corporation
chrome.exe						
chrome.exe						
chrome.exe						
chrome.exe						
chrome.exe						
proccxp64.exe	1,30					
mspaint.exe						

Name	Description
advapi32.dll	고급 Windows 32 기반
bcryptprimitives...	Windows Cryptographi
combase.dll	Windows용 Microsoft C
cryptbase.dll	Base cryptographic AP
DummyDynamicL...	
gdi32.dll	GDI Client DLL
gdi32full.dll	GDI Client DLL
imm32.dll	Multi-User Windows IM
kernel32.dll	Windows NT 기반 API
KernelBase.dll	Windows NT 기반 API
locale.nls	
msvc_p_win.dll	Microsoft® C Runtime
msvc_p90d.dll	Microsoft® C++ Runtim
msvcr90d.dll	Microsoft® C Runtime
msvcrt.dll	Windows NT CRT DLL
ntdll.dll	NT 계층 DLL
ntdll.dll	NT 계층 DLL
ole32.dll	Windows용 Microsoft C

DummyDynamicLibrary.dll Properties

Image Strings

Description: n/a
 Company: n/a
 Version: n/a
 Build Time: Wed Aug 02 21:44:30 2017

Path (Image is probably packed):
 Explore

Autostart Location:
 Explore

Load Address: 0x5E9E0000 Verify

Mapped Size: 0x41000 Bytes

Mapping Type: Image

VirusTotal: Submit

Image: 32-bit

▲ 프로세스 정보 조회로 확인할 수 있는 HMODULE 값의 정체



```

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    g_hModule = hModule;
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}
    
```

▲ 동적 라이브러리 엔트리 함수로 전달받는 HMODULE의 정체

```

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    std::tstring strModuleFileName = core::GetModuleFileName(hModule);
    return TRUE;
}
    
```

▲ dll 모듈의 메모리 주소로 얻어온 dll 모듈명



```
ECODE CopyDirectory(std::string strDirFrom, std::string strDirTo);  
ECODE CopyDirectory(std::wstring strDirFrom, std::wstring strDirTo);  
ECODE CopyDirectory(const char* pszDirFrom, const char* pszDirTo);  
ECODE CopyDirectory(const wchar_t* pszDirFrom, const wchar_t* pszDirTo);
```

특정 디렉토리의 파일들을 모두 복사하는 응용함수입니다. 여러분은 구현할 수 있을까요?



```
ECODE ReadFileContents(std::string strFilePath, std::string& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContents(std::wstring strFilePath, std::wstring& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContentsA(std::string strFilePath, std::string& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContentsA(std::wstring strFilePath, std::string& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContentsW(std::string strFilePath, std::wstring& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContentsW(std::wstring strFilePath, std::wstring& strContents, E_BOM_TYPE nEncodeType = BOM_UNDEFINED);  
ECODE ReadFileContents(std::string strFilePath, std::vector<BYTE>& outContents);  
ECODE ReadFileContents(std::wstring strFilePath, std::vector<BYTE>& outContents);
```

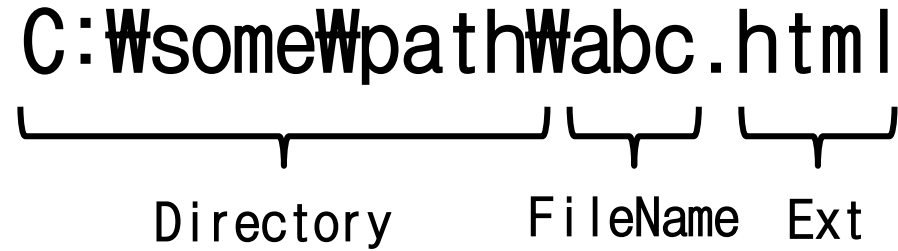
가장 많이 수행하는 파일 연산이라고 하면 “파일의 내용 읽어오기”라고 자신있게 말할 수 있습니다. 그렇기 때문에 매번 파일을 열기 위해서 CreateFile -> ReadFile -> CloseFile 을 반복해서 부르는 것이 상당히 번거롭고 자칫 실수하여 시스템 크래시 버그나 취약점을 유발할 가능성도 높습니다. 이것을 하나의 함수로 만들어두면 두고두고 편하게 사용할 수 있습니다.

특히 문자열을 읽을 때는 유니코드의 형식에 맞게 읽어와야 합니다. 문자열 반환함수들은 BOM을 추가 인자로 받는 이유가 그것입니다. 이 때 BOM_UNDEFINED라고 선언된 경우에는 내부에서 “알아서” 판단해 읽어오게 됩니다.



```
ECODE WriteFileContents(std::string strFilePath, const std::string strContents, bool bWithBOM = true);
ECODE WriteFileContents(std::wstring strFilePath, const std::wstring strContents, bool bWithBOM = true);
ECODE WriteFileContentsT(std::string strFilePath, const std::string strContents, bool bWithBOM = true);
ECODE WriteFileContentsT(std::wstring strFilePath, const std::wstring strContents, bool bWithBOM = true);
ECODE WriteFileContentsA(std::string strFilePath, const std::string strContents, bool bWithBOM = true);
ECODE WriteFileContentsA(std::wstring strFilePath, const std::string strContents, bool bWithBOM = true);
ECODE WriteFileContentsW(std::string strFilePath, const std::wstring strContents, bool bWithBOM = true);
ECODE WriteFileContentsW(std::wstring strFilePath, const std::wstring strContents, bool bWithBOM = true);
ECODE WriteFileContents(std::string strFilePath, const std::vector<BYTE>& vecContents);
ECODE WriteFileContents(std::wstring strFilePath, const std::vector<BYTE>& vecContents);
ECODE WriteFileContents(std::string strFilePath, const void* pContents, size_t tContentsSize);
ECODE WriteFileContents(std::wstring strFilePath, const void* pContents, size_t tContentsSize);
```

파일을 기록하는 경우도 마찬가지로입니다. 매우 흔히 쓰는 함수가 될 겁니다.



```
std::string ExtractDirectory(std::string strFullPath);  
std::wstring ExtractDirectory(std::wstring strFullPath);
```

```
std::string ExtractFileName(std::string strFullPath);  
std::wstring ExtractFileName(std::wstring strFullPath);
```

```
std::string ExtractFileNameWithoutExt(std::string strFilename);  
std::wstring ExtractFileNameWithoutExt(std::wstring strFilename);
```

```
std::string ExtractFileExt(std::string strFullPath);  
std::wstring ExtractFileExt(std::wstring strFullPath);
```



변경 전	변경 후
D:\some\path	D:/some/path
D:\some\path	
D:\some\////////////////////path	
D:\some\../some/path	
D:\some\../../..../some/path	

```
std::string& MakeFormalPath(std::string& strInformalPath);
std::wstring& MakeFormalPath(std::wstring& strInformalPath);
```

```
std::string MakeFormalPath(LPCSTR pszInformalPath);
std::wstring MakeFormalPath(LPCWSTR pszInformalPath);
```



```
ECODE GrepFiles(std::string strDir, std::string strPattern, std::vector<std::string>& outFileVec);
ECODE GrepFiles(std::wstring strDir, std::wstring strPattern, std::vector<std::wstring>& outFileVec);
ECODE GrepFilesRecursively(std::string strDir, std::string strPattern, std::vector<std::string>& outFileVec);
ECODE GrepFilesRecursively(std::wstring strDir, std::wstring strPattern, std::vector<std::wstring>& outFileVec);
ECODE GrepDirectories(std::string strDir, std::string strPattern, std::vector<std::string>& outFileVec);
ECODE GrepDirectories(std::wstring strDir, std::wstring strPattern, std::vector<std::wstring>& outFileVec);
ECODE GrepDirectoriesRecursively(std::string strDir, std::string strPattern, std::vector<std::string>& outFileVec);
ECODE GrepDirectoriesRecursively(std::wstring strDir, std::wstring strPattern, std::vector<std::wstring>& outFileVec);
```

특정 디렉토리로부터 파일 목록을 수집해주는 함수입니다.



고생 많으셨습니다!