Hanbit eBook

Realtime 88

C++ 개발자를 위한

WIN32 오픈소스 라이브러리 100

박주항 지음

한빛미디어 Hanbit Media, Inc.

C++ 개발자를 위한

WIN32 오픈소스 라이브러리 100

C++ 개발자를 위한 WIN32 오픈소스 라이브러리 100

초판발행 2014년 12월 30일

지은이 박주항 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-727-9 15000 / **정가** 15,000원

총괄 배용석 / **책임편집** 김창수 / **기획·편집** 정지연
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
영업 김형진, 김진불, 조유미 / **마케팅** 박상용

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

지은이_ **박주항**

2006년 클라이언트 프로그래머로 게임 회사에 입사하였지만, 회사 사정으로 서버 쪽 업무를 맡게 되면서 그 뒤로는 여러 게임의 온라인 플랫폼을 구축하고 제작해왔다. C++를 주 언어로 사용하지만, 최근에 모바일 플랫폼 관련 회사에서 일하면서 자바 언어를 다루게 되었고 C++ 언어와는 다른 자바만의 매력에 빠져 자바를 보조 언어로 사용하고 있다. 프로그래밍 자체를 좋아하여 운영체제 개발부터 파이썬, 루아 같은 스크립트 언어 활용까지 프로그래밍의 모든 영역에 관심을 두고 있다. 유용한 오픈소스를 자신의 프로젝트에 활용하는 것을 좋아하여 시간이 날 때마다 여러 오픈소스 공유 사이트에서 소스 코드를 내려받아 분석하는 것을 취미로 삼고 있다. 또한, 어드벤처 게임을 광적으로 좋아해서 '로라 보우 2 - 태양신의 단도', '스페이스 퀘스트 4' 등 시에라Sierra사 게임의 한글 패치를 제작하기도 했다. 현재 프리랜서로 일하는 중이다.

저자 서문

컴퓨터 산업이 발전한 이래로 수많은 오픈소스가 공개되었다. 초기 소프트웨어 시대에는 프로그래밍 언어의 종류가 많지 않아서 C 언어로 구현된 라이브러리가 대부분이었지만, 최근에는 수많은 언어로 제작된 오픈소스가 끊임없이 발표되고 있다. 특히 자바는 공개된 라이브러리가 너무 많아서 그 수를 헤아리기 힘들 정도다.

자바는 메이븐 같은 라이브러리 저장소 관리 시스템이 있고, C#에도 Nuget이라는 패키지 관리 툴이 있어서 필요한 라이브러리를 쉽게 프로젝트에 반영할 수 있다. 이렇게 라이브러리 관리 시스템이 정교하게 구축된 언어도 있지만, C++는 이러한 시스템이 매우 부실하다. 물론 리눅스 시스템은 네트워크로 라이브러리를 쉽게 설치할 수 있지만, 윈도우 환경에서는 이것마저도 불가능하다.

윈도우 Visual Studio에 C++ 라이브러리 관리를 위한 Nuget 시스템이 있지만, 관리되는 라이브러리 수도 많지 않고 라이브러리를 활용하려고 해도 Visual Studio 버전에 맞지 않아 그 기능을 제대로 활용할 수가 없다. WIN32 환경에서 C++로 프로그래밍할 때 오픈소스를 활용하려면 안타깝지만 해당 오픈소스를 내려받아서 직접 빌드하여 프로젝트에 반영하는 수밖에 없다.

오픈소스를 이렇듯 조금은 힘들게 프로젝트에 반영하는 문제 이외에도 WIN32 환경에서는 또 다른 난점이 존재한다. 오픈소스 대부분은 WIN32 환경을 주 대상으로 제작된 것이 아니라서 라이브러리를 활용하려면 WIN32 프로젝트를 별도로 생성해야 한다. 그런데 'WIN32 프로젝트를 생성하는 작업'이 생각보다 만만치 않다. 어떤 오픈소스는 CMake나 premake 등의 유틸리티를 사용해서 WIN32 프로젝트로 생성해야 하고, 자체 빌드 환경을 활용해서 WIN32 프로젝트를 생성해야 하는 경우도 있다. 이렇게 해서 변환된 WIN32 프로젝트가 제대로 빌드되면 좋겠

지만, 리눅스에서 사용하는 함수는 WIN32를 지원하지 않아서 포팅해야 하는 경우도 있다. 또한, 일부 오픈소스는 빌드하려면 다른 외부 라이브러리가 필요한데, 외부 라이브러리를 링킹하는 것은 매우 귀찮은 작업이고 개발자를 좌절하게 한다.

이 책에서는 WIN32 환경의 Visual Studio 2013에서 C++로 프로그래밍할 때 오픈소스를 쉽게 자신의 프로젝트에 적용하는 것을 목표로 삼고 있다. 아울러 수많은 오픈소스를 살펴보면서 필요한 기능의 라이브러리를 쉽게 찾을 수 있는 레퍼런스가 되길 바란다. Visual Studio 버전을 2013으로 한정했지만, 상위 버전이 나온다 하더라도 이 책의 내용을 참고한다면 쉽게 오픈소스를 마이그레이션할 수 있을 것이다.

부디 이 책을 통해 WIN32 Visual Studio 환경에서 쾌적한 프로그래밍을 할 수 있기를 바란다.

이 책을 읽기 전에

오픈소스의 필요성

오픈소스의 필요성에 대해서는 새삼 언급할 필요가 없을 것 같다. 프로젝트를 진행하다가 특정 기능이 필요할 때 그 기능을 직접 구현하는 것보다는 이미 완성된 라이브러리를 사용한다면 프로젝트 개발 시간을 크게 단축할 수 있고 프로젝트 본래의 기능 구현에 집중할 수 있다.

프로그래밍 시대 초창기에는 기반 환경이 조성되지 않아서 자신에게 필요한 기능을 직접 제작해야 했다. 하지만 요즘은 네트워크가 발전하고 수많은 오픈소스가 소셜 네트워크 서비스를 통해서 공유되므로 필요로 하는 기능을 지원하는 라이브러리를 찾아보면 어딘가에는 반드시 존재한다. 이제부터라도 밑바닥부터 차곡차곡 개발하는 것보다는 레고 조립 방식을 어느 정도 받아들여서 프로그래밍할 필요가 있다. 물론 어디까지나 레고 부품은 곁가지고 개발하는 프로젝트에 도움을 주는 존재이지 그 이상의 것은 아니다.

극단적으로 이야기해서 웹 서버를 구축한다고 가정해 보자. 웹 서비스를 구현하기 위해 아파치 서버를 개발하고 톰캣 서버를 개발할 수는 없지 않은가? 웹 서버 환경을 구축해 주는 기존의 툴을 사용하고 JSP 등의 코드를 작성하는 것만으로도 웹 서비스를 구축할 수 있다. 다시 말해, 우직하게 처음부터 끝까지 모든 기능을 구현할 필요는 없다.

WIN32 환경에서 C++로 프로그램을 개발할 때 이런 오픈소스의 필요성은 더욱더 커진다. C++가 훌륭한 언어이기는 하지만 다른 언어보다 생산성이 떨어지는 것은 사실이므로 이를 극복하기 위해서라도 오픈소스를 적극적으로 활용해서 개발 시간을 단축해야 한다고 생각한다.

WIN32 환경에서 오픈소스 빌드

오픈소스의 필요성을 앞에서 언급하기는 했지만, 오픈소스를 프로젝트에 실제 적용하는 일이 그리 녹록하지만은 않다. 가장 큰 이유는 오픈소스를 빌드하기가 쉽지 않기 때문이다. 물론 일부 오픈소스는 빌드의 불편함을 없애기 위해 미리 빌드된 바이너리 버전을 제공하는 경우가 많다. 일반적으로 이런 빌드된 버전은 프로젝트에 쉽게 적용할 수 있지만, 다음의 몇 가지 이유로 프로젝트에 제대로 적용되기가 어려워서 소스 코드를 직접 빌드해야 한다.

- 미리 빌드된 라이브러리 중 일부는 32비트만을 고려해서 빌드된 버전이 많다. 결국, 64비트 버전의 라이브러리를 얻으려면 소스 코드를 빌드해야 한다.
- Visual Studio 버전이 다르면 링크 에러가 발생하는 경우가 있다. 예를 들어, 미리 빌드된 라이브러리가 사용하는 런타임 라이브러리 버전과 이 라이브러리를 활용하는 프로젝트에서 사용하는 런타임 라이브러리 버전이 다르면 라이브러리 링킹 단계에서 라이브러리끼리 충돌하여 빌드되지 않는 상황이 발생한다.
- 정적 라이브러리와 동적 라이브러리 사용 시에도 문제가 발생한다. Visual Studio에서 모듈을 빌드할 때 코드 생성 옵션을 지정할 수 있는데, MT[Multi-thread] 옵션과 MD[Muilti-thread DLL] 옵션을 줄 수 있다. 이 옵션은 런타임 라이브러리를 정적으로 사용할 것인지 동적으로 사용할 것인지를 결정한다. 활용하려는 오픈소스 라이브러리가 MT 옵션으로 빌드되었는데, 프로젝트를 MD 옵션으로 컴파일하면 정상적으로 빌드되지 않는다. 따라서 오픈소스 라이브러리의 코드 생성 옵션을 최종 프로젝트의 코드 생성 옵션으로 변경해서 빌드해야 한다.

- 라이브러리의 소스 코드가 오래전에 제작되었다면 새로 빌드하는 것이 좋다. 최신 Visual Studio에서는 런타임 라이브러리가 멀티 스레드 세이프하지만 아주 오래된 라이브러리는 싱글 스레드만을 고려하였기 때문에 최종 애플리케이션이 멀티 스레드를 사용한다면 어떤 오동작을 일으킬지를 보장할 수가 없다. 따라서 소스 코드를 확보한 상태라면 새롭게 빌드하는 것이 필수다.

이외에도 몇 가지 이유로 미리 컴파일된 라이브러리를 활용하기보다는 소스 코드를 확보하여 빌드하는 것이 나중을 위해서도 좋다. 오픈소스는 유용하긴 하지만 잠재된 버그 또한 무시할 수 없으므로 문제가 생긴다면 바로 수정할 수 있도록 소스 코드를 반드시 장악해야 한다.

물론 미리 컴파일된 라이브러리가 있다면 특별한 문제가 없는 한 최종 애플리케이션에서 크게 말썽을 일으키지 않는다. 이런 경우는 그나마 괜찮지만, 오픈소스 대부분은 미리 컴파일된 라이브러리를 제공하지 않는다. 따라서 어쩔 수 없이 이런 오픈소스를 직접 빌드해야 한다. 또 다른 문제는 애초에 WIN32를 대상으로 개발된 라이브러리는 아무런 문제없이 빌드되지만, 크로스 플랫폼을 대상으로 하거나 유닉스 계열의 라이브러리인 경우에는 WIN32에서 제대로 빌드하기 어렵다는 것이다. 오픈소스를 빌드하려고 시도해 봤다면 어떤 라이브러리는 온종일 시도해도 빌드에 실패해서 좌절을 겪었던 적이 있을 것이다.

이런 이유로 이 책을 집필하게 되었다. 즉, WIN32 환경에서 프로그래밍할 때 효율적인 시간 사용을 위해 오픈소스를 최대한 활용하는 것이 좋은데, 이런 오픈소스를 WIN32 환경에서 활용하기가 매우 까다롭다. 이런 부분에 대한 비용을 줄여주는 수많은 유용한 오픈소스가 존재함에도 세상에 알려지지 않은 오픈소스가 많아서 이를 알리고자 이 책을 집필하였다.

이 책은 100여 가지의 오픈소스 라이브러리를 소개하고 이를 활용할 수 있는 방법을 설명한다. 그리고 이런 오픈소스를 최대한 불편 없이 사용할 수 있는 방법을 전달하고자 한다. 이 책을 통해 미리 오픈소스를 사용할 수 있는 빌드 환경을 구축한다면 새 프로젝트를 진행할 때 필요한 기능을 가진 라이브러리를 손쉽게 프로젝트에 적용할 수 있을 것이다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. 무료로 업데이트되는 전자책 전용 서비스입니다.

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위해 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

사전 준비사항

이 책에서 소개하는 라이브러리들은 각각의 내용이 독립적이므로 차례대로 읽을 필요는 없다. 하지만 라이브러리를 빌드하기 위해서는 여기서 설명하는 프로그래밍 도구를 설치하고, 환경 변수 등의 설정은 꼭 해줘야 한다. 따라서 사전 준비사항의 내용을 반드시 읽고 숙지해 주기 바란다.

필수 프로그램 설치

이 책에서 언급한 라이브러리를 빌드하고 활용하려면 다음 프로그램이 필요하다.

Visual Studio 2013

이 책에서 소개하는 모든 오픈소스 라이브러리 및 샘플 코드는 Visual Studio 2013에서 테스트하였다. 무료 버전인 Visual Studio 2013 Express 버전은 다음 링크에서 내려받을 수 있다.

http://www.visualstudio.com/downloads/download-visual-studio-vs

링크를 따라가면 DVD 이미지를 받아서 설치할 것인지 웹에서 내려받아 설치할 것인지를 묻는 화면이 나온다. 편의를 위해 wdexpress_full.exe 파일을 내려받아 설치한다.

최근 마이크로소프트에서 Visual Studio 2013 Community 버전을 무료로 배포했는데, 이 버전은 Visual Studio의 상용 기능을 대부분 지원하므로 가능하다면 이 버전을 내려받아 사용한다. 소스 코드는 Express 버전을 통해 테스트되었지만, Community 버전이 완성도가 더 높은 IDE이므로 되도록 Community 버전 사용을 추천한다.

CMake GUI

CMake는 Cross Platform Make의 줄임말로, 멀티 플랫폼에서 사용할 수 있는 Make 파일을 생성해 주는 도구다. CMake는 윈도우에서 Visual Studio용 솔루션 파일을 생성한다. 단지 프로젝트 빌드를 위한 파일만을 생성하므로 Meta Make라고도 불린다. 기존 유닉스 계열의 Make와 달리 빌드 규칙을 양식에 맞게 작성하면 유닉스 계열뿐만 아니라 윈도우 계열에서도 빌드할 수 있게 프로젝트를 생성해 준다. 즉, CMake는 멀티 플랫폼을 위한 빌드 지원 시스템이며 GUI 버전도 존재하므로 사용하기가 매우 쉽다. CMake는 다음 링크에서 내려받아 설치한다. 집필 시 사용한 버전은 3.0.2지만, 버전이 업데이트되었다면 최신 버전을 사용한다.

http://www.cmake.org/download/

수많은 오픈소스가 프로젝트 자동 생성을 위해 CMake를 사용하므로 사용방법을 간단하게나마 숙지해야 한다. CMake 사용의 좋은 예제로 이 책에서 소개한 zlib 라이브러리를 먼저 살펴보길 바란다.[01]

TortoiseSVN

TortoiseSVN은 소스 코드 관리를 위한 프로그램이다. 오픈소스 대부분은 SVN 시스템을 차용하고 있어서 오픈소스를 내려받으려면 소스 코드 관리 도구인 TortoiseSVN 을 설치해야 한다. 32비트와 64비트 버전이 있는데, 운영체제 환경에 맞는 버전을 설치한다. 다음 링크에서 내려받을 수 있다.

http://tortoisesvn.net/downloads.html

01 이 책에서 CMake를 사용해서 프로젝트를 생성하는 라이브러리 중 특별한 설정이 필요없는 라이브러리는 CMake에 대한 설명을 생략한 경우가 많으므로 꼭 zlib 라이브러리의 빌드 방법을 먼저 살펴보길 권한다.

Python 2.7

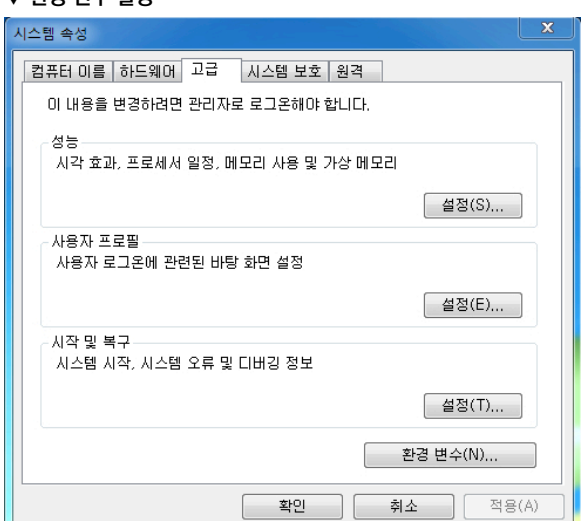
프로젝트 파일을 생성하기 위해 파이썬의 스크립트를 사용하는 오픈소스 라이브러리가 있다. 이 오픈소스의 프로젝트 파일을 생성하려면 파이썬을 설치해야 한다. 현재 파이썬은 2.x 버전과 3.x 버전으로 나뉘는데, 두 버전을 모두 설치하면 좋다. 다음 링크에서 파이썬을 내려받아 설치한다. 집필 시 2.x 버전은 2.7.8, 3.x 버전은 3.4.2다.

https://www.python.org/downloads/

두 버전을 모두 설치했지만 빌드 스크립트를 사용하기 위해 2.7.8 버전을 사용한다. C:\Python27 경로에 파이썬을 설치했다면 다음 작업으로 환경 변수를 확인하자. 여기서는 Windows 7을 기준으로 설명한다.

'컴퓨터 → 속성 → 고급 시스템 설정'을 선택해서 대화창이 뜨면 환경 변수 버튼을 누른다.

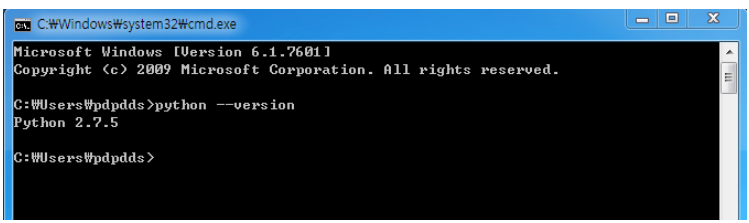
▼ 환경 변수 설정



환경 변수 창이 뜨면 시스템 변수의 Path 항목을 클릭하여 Python 경로가 추가되었는지 확인한다(C:\Python27\;C:\Python27\Scripts). 이 값이 추가되지 않았으면 Path 변수에 추가한다. 기존 Path 값을 잘못 건드리면 문제가 생길 수 있으므로 조심해서 경로를 덧붙인다.

파이썬 설정이 제대로 되었는지는 콘솔 창을 실행해서 확인할 수 있다. '윈도우키 + R' 키를 누르면 실행 창이 나오는데, 에디트 박스에 CMD를 입력하고 확인을 누르면 콘솔 창이 뜬다. 여기에 `python --version`을 입력해서 버전 값이 제대로 나오면 파이썬 설정이 제대로 된 것이다(화면의 컴퓨터에 설치된 파이썬 버전은 2.7.5다).

▼ 파이썬 환경 변수 설정 확인



서드 파티 준비

오픈소스를 빌드하려면 미리 준비해야 할 서드 파티가 있다.

부스트 라이브러리 설치

부스트 라이브러리 공식 사이트[02]에 접속하여 메인 페이지 오른쪽의 Version 1.55.0을 누른다.[03] 부스트를 빌드하는 데 시간이 오래 걸리므로 소스 코드를 내려받는 대신 Other DownLoads의 Windows binaries를 클릭한다. 이 페이지에서

02 http://www.boost.org/

03 부스트는 버전 업데이트가 빠른 오픈소스의 하나다. 집필 당시 버전은 1.55.0였다. 이 버전보다 최신 버전이 있다면 최신 버전을 설치해 주는 것도 좋지만, 구 버전에서 아무 문제가 없던 부분이 최신 버전에서는 문제가 생길 수 있으므로 될 수 있으면 책에서 언급한 버전의 부스트를 설치한다.

boost_1_55_0_msvc-12.0-64.exe와 boost_1_55_0_msvc-12.0-32.exe 두 파일을 내려받고 로컬에 설치한다.

DirectX SDK 2010 설치

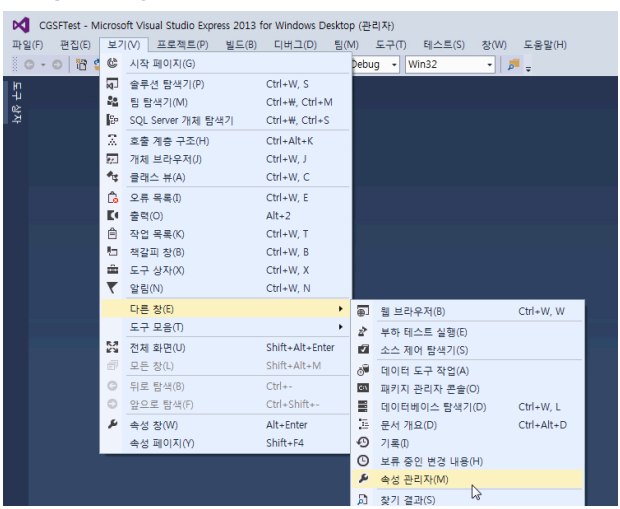
마이크로소프트 홈페이지[04]에서 Download 버튼을 눌러서 파일을 내려받은 후 로컬에 설치한다(집필 시 DirectX SDK 2010 6월 버전). DirectX SDK에는 32비트와 64비트 버전의 라이브러리가 모두 포함되어 있다.

서드 파티 경로 설정

부스트 라이브러리와 DirectX SDK가 설치된 곳을 지정해서 두 라이브러리가 모든 프로젝트에 적용되도록 전역으로 설정한다.

먼저 '보기 → 다른 창 → 속성 관리자'를 선택하여 속성 관리자 창을 연다.

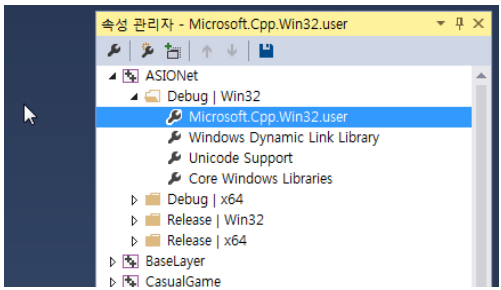
▼ 속성 관리자 창



04 http://www.microsoft.com/en-us/download/details.aspx?id=6812

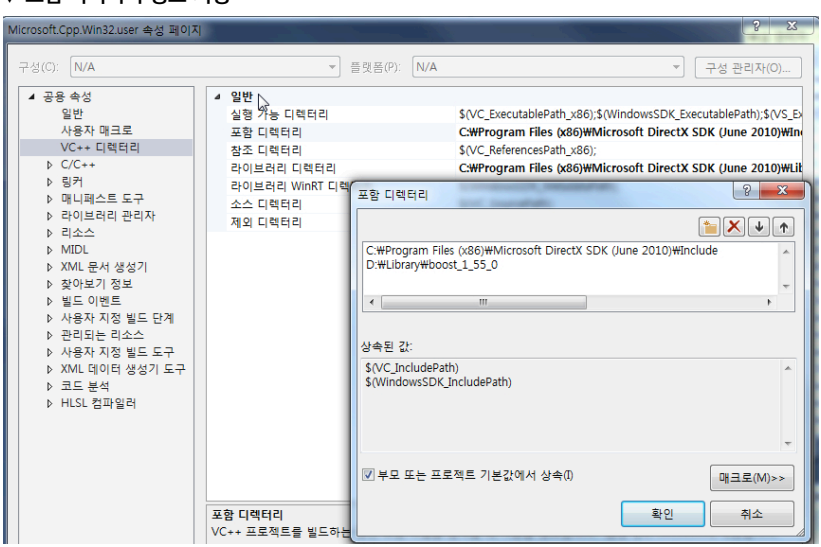
속성 관리자 창에서 아무 프로젝트나 선택한 다음 Microsoft.Cpp.Win32.user를 더블 클릭한다.

▼ Microsoft.Cpp.Win32.user



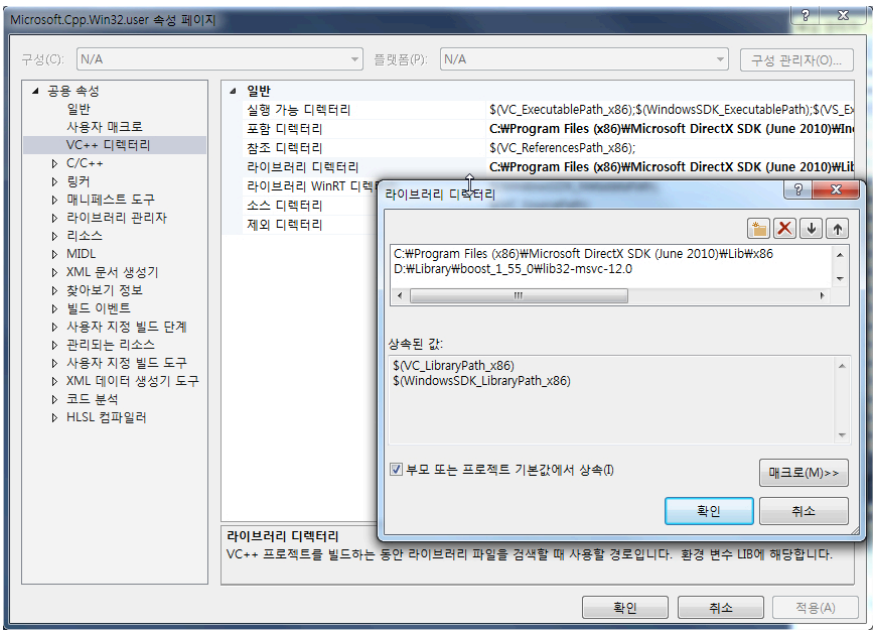
'공용 속성 → VC++ 디렉터리 → 포함 디렉터리'를 편집해서 부스트와 DirectX 경로를 설정한다(해당 폴더 위치는 각자 설치한 경로 위치로 지정한다). 64비트도 동일하게 적용한다.

▼ 포함 디렉터리 경로 지정



'공용 속성 → VC++ 디렉터리 → 라이브러리 디렉터리'를 편집해서 부스트와 DirectX 라이브러리 경로를 설정한다. 64비트는 64비트 라이브러리 경로로 설정한다.

▼ 라이브러리 디렉터리 경로 지정



이렇게 설정하면 이후 DirectX, 부스트 라이브러리의 포함 경로와 라이브러리 경로는 모든 프로젝트에 전역으로 적용된다.

기타 도구

오픈소스를 빌드하기 위해 필요한 기타 도구를 살펴보자.

VS2013 x64 네이티브 도구 명령 프롬프트

콘솔 창과는 다른 프롬프트로 nmake라는 빌드 도구를 사용하려면 해당 콘솔 창을 실행해야 한다. 64비트 라이브러리를 빌드하기 위해 'VS2013 x64 네이티브 도구 명령 프롬프트'를 실행하자. '모든 프로그램 → Visual Studio 2013 → Visual Studio Tools' 항목에서 확인할 수 있다.

NASM

NASM은 Netwide Assembler의 약자로, 어셈블리 언어로 작성된 코드를 컴파일하는 데 필요한 윈도우용 도구로 최신 버전은 2.11.05다. 다음 링크에서 내려받을 수 있다.

http://www.nasm.us/pub/nasm/releasebuilds/2.11.05/win32/

nasm-2.11.05-win32.zip 파일을 내려받아 C:\nasm 경로에 압축을 푼다. 인스톨러 버전으로 설치했다면 환경 변수의 PATH에 해당 프로그램의 경로가 자동으로 등록된다.

ActivePerl

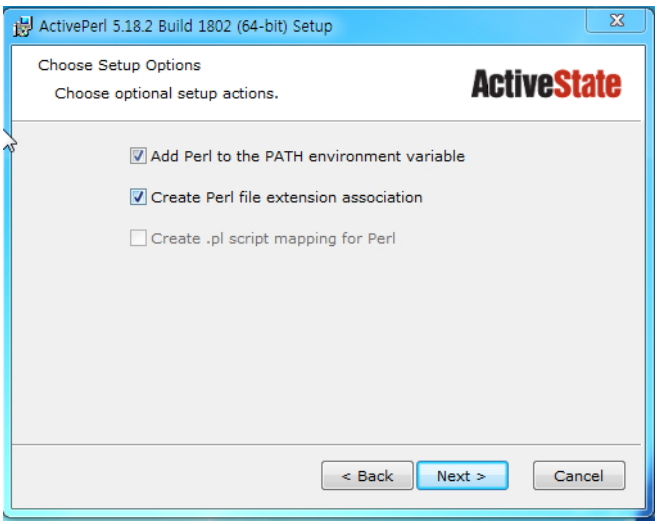
ActiverPerl은 윈도우 환경에서 Perl 언어를 사용하기 위해 제공되는 도구로, 다음 링크에서 내려받아 설치한다. 사용한 버전은 5.18.2.18이다.

http://www.activestate.com/activeperl

환경에 따라 32비트/64비트 설치 프로그램 중 하나를 선택해서 설치한다.

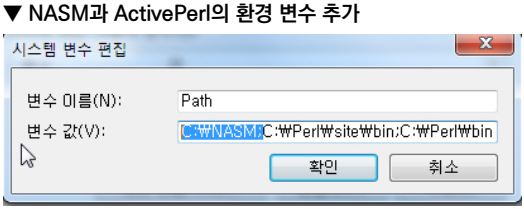
ActivePerl이 환경 변수에 등록되도록 체크 박스는 해제하지 않는다.

▼ ActivePerl 설치 화면



NASM과 ActivePerl을 설치했으면 두 프로그램이 명령창의 어느 위치에서도 실행될 수 있도록 환경 변수를 추가한다. 다음 그림은 C:\Nasm, C:\Perl에 설치된 두 프로그램이 시스템 변수의 Path 변수에 추가된 것을 보여준다. ActivePerl은 프로그램 설치 시 자동으로 환경 변수에 등록이 되지만, NASM은 그렇지 않으므로 환경 변수에 등록되었는지 꼭 확인하자.

▼ NASM과 ActivePerl의 환경 변수 추가



git

다음 링크에서 윈도우용 git을 내려받는다.

http://git-scm.com/download/win

설명 규칙

이 책은 라이브러리를 소개하고 빌드할 때 다음 규칙을 따른다.

원본 소스의 빌드

원본 소스는 원칙적으로 32비트 디버그/릴리스 라이브러리 생성에 중점을 두었으며 64비트 모듈의 생성에 대해서는 언급하지 않는다. 64비트 모듈을 생성할 수 있으나 번거로움을 줄이기 위해 32비트 라이브러리 생성에만 집중하였다. 64비트 모듈 생성 시 딱히 어려운 점이 있는 것은 아니다.

빌드된 라이브러리의 유지 보수

빌드된 라이브러리는 다른 라이브러리의 외부 의존 라이브러리가 될 수 있다. 따라서 이후 다른 라이브러리가 쉽게 해당 라이브러리를 링크할 수 있도록 prebuilt 폴더에 오픈소스의 헤더 파일과 미리 빌드된 라이브러리를 등록한다. 예를 들어, zlib 라이브러리는 prebuilt 폴더에 다음과 같이 저장된다.

- prebuilt\zlib\include: 포함 경로
- prebuilt\zlib\lib\x86\debug: 32비트 디버그 라이브러리 경로
- prebuilt\zlib\lib\x86\release: 32비트 릴리스 라이브러리 경로

릴리스 모드로만 라이브러리를 배포한다면 해당 경로는 다음과 같다.

- prebuilt\zlib\lib\x86

라이브러리 경로

이 책에 소개된 라이브러리의 빌드 방법을 설명하기 위해 하드 드라이브 경로는 D:나 E:로, 라이브러리 폴더는 D:\Library나 E:\Library로 설정한다. 라이브러리 파일의 크기가 크면 직접 해당 오픈소스에서 라이브러리를 내려받은 후 빌드해서 prebuilt 폴더에 넣어준다. 책을 살펴볼 때에는 이 부분을 고려하여 자신의 컴퓨터 환경에 맞게 오픈소스를 빌드한다.

오픈소스 버전 문제

이 책을 집필하는 중에도 오픈소스의 버전은 계속 업데이트되었고, 책이 출간된 이후에도 계속 업데이트될 것이다. 이 책에서 라이브러리 버전은 필자가 테스트한 버전을 표기한다. 물론 업데이트 버전을 사용하더라도 책에서 언급한 빌드 절차를 사용하는 데에는 별로 문제가 없을 것이다. 단, 최신 버전을 사용해서 빌드하는 데 문제가 발생하면 책에서 언급한 버전을 사용해서 빌드하길 바란다.

샘플 프로젝트

오픈소스를 빌드하는 방법도 중요하지만, 해당 오픈소스를 적용한 결과를 미리 살펴보는 것도 중요하다고 판단되어 오픈소스 라이브러리를 활용한 샘플 프로젝트를 구축하였다. 단, 오픈소스에서 별도로 라이브러리를 활용한 예제를 포함하는 경우는 샘플 프로젝트를 만들지 않았다. 오픈소스를 활용한 샘플 프로젝트는 다음 링크에서 내려받을 수 있다.

https://github.com/pdpdds/Win32OpenSourceSample

해당 프로젝트는 미리 컴파일된 라이브러리가 필요하므로 소스를 내려받은 루트 폴더에 prebuilt 폴더를 만들고 이 안에 관련 라이브러리를 추가해야 한다. 이 폴더에는 책의 내용을 참조하면서 직접 빌드한 라이브러리의 헤더와 라이브러리가

추가되지만, 오픈소스 라이브러리를 활용한 결과를 먼저 확인하고 싶다면 미리 컴파일한 라이브러리 모음 파일을 내려받아 설치한다.

https://drive.google.com/folderview?id=0B4zZS0o3dor0UW44N0tBRFloM0U&usp=sharing

이 링크에서 prebuilt.rar 파일을 내려받는다. 소스 코드가 D:\Dev\Win32OpenSourceSample 폴더에 있다면 컴파일된 라이브러리 모음 파일은 D:\Dev\Prebuilt 폴더에 있어야 한다. 솔루션 파일을 실행해서 프로젝트가 정상적으로 빌드되는지 확인한다. 편의를 위해 32비트 디버그 모드에서만 결과를 확인할 수 있도록 정리하였다. 또한, 샘플 실행 시 필요한 리소스도 앞의 링크에서 내려받을 수 있다(Debug.rar). 내려받은 다음 Win32OpenSourceSample\Debug 폴더에 압축을 푼다.

PART 1
데이터 처리

1 | 문서 작업

PDF 파일이나 엑셀 파일을 프로그램에서 제어할 수 있다면 매우 유용할 것이다. 이번 장에서는 이런 문서 파일 제어와 관련된 라이브러리를 소개한다.

- libHaru
- PoDoFo
- xlsLib
- libxls
- audio_ostream

라이브러리명	libHaru
분류	문서 작업 – PDF 문서 생성
설명	PDF 문서 제작을 할 수 있게 하는 라이브러리로, 이 라이브러리를 사용하면 PDF 문서에 글자를 추가하거나 이미지를 추가하여 PDF 문서로 저장할 수 있다.
다운로드	http://libharu.org/
버전	2.3.ORC1

소스 빌드

해당 소스는 CMake로 빌드할 수 있고, 특별한 설정 없이 솔루션 파일이 생성된다. 하지만 zlib를 참조하므로 libHaru.sln을 실행해서 libhpdf 프로젝트와 libhpdfs 프로젝트를 선택한 다음 '속성 → C/C++ → 추가 포함 디렉터리'에 zlib 모듈의 헤더 경로를 추가한다. 그리고 libhpdf 프로젝트의 '속성 → 링커 → 일반 → 추가 라이브러리 디렉터리'에 zlib 모듈의 라이브러리 경로를 추가하고 '속성 → 링커 → 입력 → 추가종속성'에 zlibd.lib를 추가한다.

NOTE_

환경 설정을 하고 빌드하더라도 정상적으로 빌드되지 않는데, 이는 컴파일러 오류 C2373 때문이다. 해당 오류에 관해서는 다음 링크를 참고하자.

http://msdn.microsoft.com/ko-kr/library/k6z2ykx4(v=vs.90).aspx

이 오류를 해결하려면 hpdf_image_ccitt.c 파일의 HPDF_Image_LoadRaw1 BitImage FromMem 함수 윗부분을 다음과 같이 수정한다.

```
#ifdef HPDF_DLL_MAKE
HPDF_Image __stdcall
#else
HPDF_Image
#endif
```

이처럼 수정하고 빌드하면 문제없이 라이브러리가 빌드된다. 라이브러리는 \build\src\Debug\Window 폴더에 생성된다. 릴리스용 라이브러리도 생성한다.

샘플 코드

libHaru 라이브러리를 활용하는 예제를 살펴보자. 샘플 예제는 libharuEx 프로젝트에서 확인할 수 있으며 libHaru 동적 라이브러리를 사용했다. libHaru 동적 라이브러리는 zlib DLL 파일을 참조하므로 zlibd.dll 파일도 디버깅 경로에 복사하고, 전처리기에 HPDF_DLL 매크로도 추가한다.

다음은 여러 서체를 사용해서 글자를 PDF 문서에 출력하는 예제다.

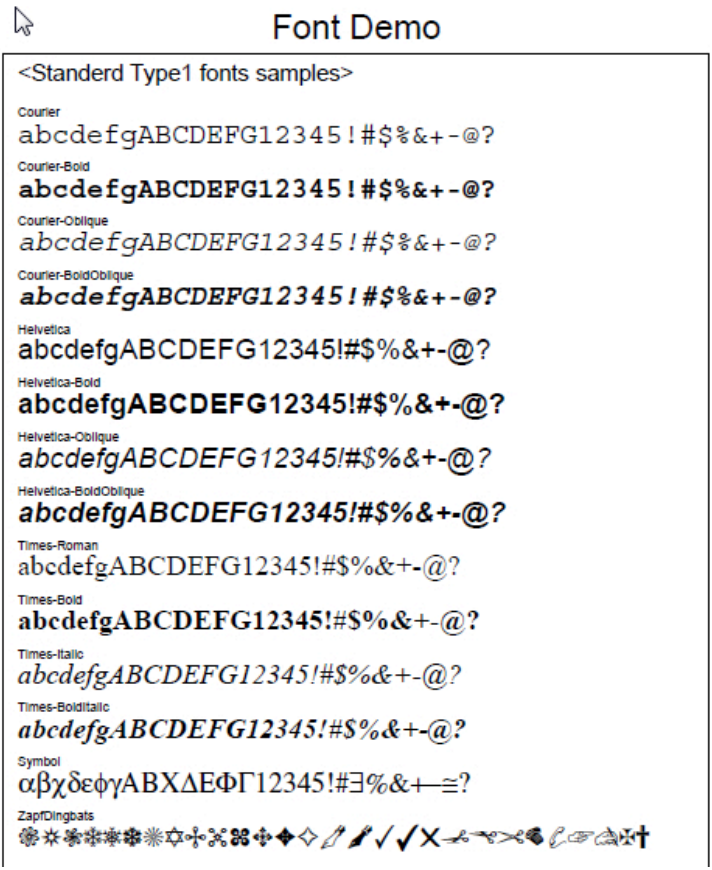
[PDF 문서로 문자열 출력하기]

```
int main(int argc, char **argv)
{
      ……
      //PDF 객체를 생성한다.
      pdf = HPDF_New(error_handler, NULL);
      if (!pdf) {
         printf( " error: cannot create PdfDoc object\n " );
         return 1;
      }

      if (setjmp(env)) {
         HPDF_Free(pdf);
         return 1;
      }

      //페이지를 추가한다.
      page = HPDF_AddPage(pdf);

      //페이지의 높이와 너비를 구한다.
```

```
    height = HPDF_Page_GetHeight(page);
    width = HPDF_Page_GetWidth(page);

    //선의 두께를 지정한다.
    HPDF_Page_SetLineWidth(page, 1);

    //사각형을 그린다.
    HPDF_Page_Rectangle(page, 50, 50, width - 100, height - 110);
    HPDF_Page_Stroke(page);

    //폰트를 얻어와서 설정한다. 크기는 24로 설정한다.
    def_font = HPDF_GetFont(pdf, "Helvetica", NULL);
    HPDF_Page_SetFontAndSize(page, def_font, 24);

    //화면 상단 중간에 타이틀 글자를 출력한다.
    tw = HPDF_Page_TextWidth(page, page_title);
    HPDF_Page_BeginText(page);
    HPDF_Page_TextOut(page, (width - tw) / 2, height - 50, page_title);
    HPDF_Page_EndText(page);

    HPDF_Page_BeginText(page);
    HPDF_Page_SetFontAndSize(page, def_font, 16);
    HPDF_Page_TextOut(page, 60, height - 80, "<Standerd Type1 fonts
samples>");
    HPDF_Page_EndText(page);

    HPDF_Page_BeginText(page);
    HPDF_Page_MoveTextPos(page, 60, height - 105);

    //박스 안에 여러 폰트를 사용해서 텍스트를 출력한다.
    i = 0;
    while (font_list[i]) {
```

```
        const char* samp_text = " abcdefgABCDEFG12345!#$%&+-@? " ;
        HPDF_Font font = HPDF_GetFont(pdf, font_list[i], NULL);

        /* print a label of text */
        HPDF_Page_SetFontAndSize(page, def_font, 9);
        HPDF_Page_ShowText(page, font_list[i]);
        HPDF_Page_MoveTextPos(page, 0, -18);

        /* print a sample text. */
        HPDF_Page_SetFontAndSize(page, font, 20);
        HPDF_Page_ShowText(page, samp_text);
        HPDF_Page_MoveTextPos(page, 0, -20);

        i++;
    }
    HPDF_Page_EndText(page);

    //PDF 파일로 저장한다.
    HPDF_SaveToFile(pdf, fname);

    /* clean up */
    HPDF_Free(pdf);

    return 0;
}
```

▼ 출력 결과

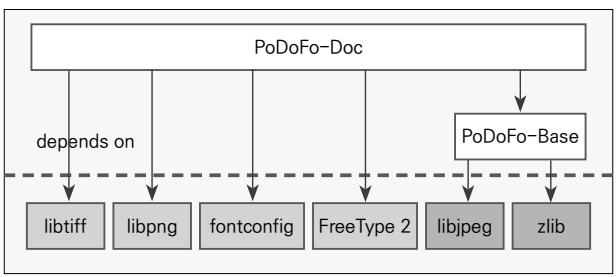


libHaru 라이브러리의 demo 폴더에는 데모 예제가 많이 있으니 해당 데모를 분석하면 라이브러리 활용 방법을 더 자세히 알 수 있다. libHaru 라이브러리를 활용하면 직접 워드 프로세서를 만들 수 있으므로 관심 있는 분은 도전해 보기 바란다.

라이브러리명	PoDoFo
분류	문서 작업 – PDF 문서 생성
설명	PoDoFo는 Portable Document Format의 약어로, PDF 파일을 생성하거나 내용을 파싱해서 메모리에서 데이터를 수정할 수 있게 하는 라이브러리다. 코드는 C++로 구현되었고 유닉스, OS X에서 컴파일할 수 있으며 윈도우에서도 빌드할 수 있다.
다운로드	http://podofo.sourceforge.net/download.html
버전	0.9.3

소스 빌드

▼ PoDoFo 라이브러리 구조



PoDoFo 라이브러리는 그림에서 보듯이 많은 외부 라이브러리를 참조하지만, 필수 외부 라이브러리는 libjpeg와 zlib다. CMake로 빌드할 수 있는데, GUI 버전을 사용하면 빌드하기가 매우 번거로우므로 콘솔 창을 사용해서 빌드한다. 먼저 PoDoFo 라이브러리 소스가 있는 경로로 이동해서 build.cmd라는 배치 파일을 만든다(라이브러리 모음이 E:\Library\prebuilt 경로에 있다고 가정한다).

```
del cmakecache.txt
set FTDIR=E:\Library\prebuilt\freetype
set FTLIBDIR=E:\Library\prebuilt\freetype\lib\x86
set JPEGDIR= E:\Library\jpeg-9a
set ZLIBDIR= E:\Library\zlib-1.2.8
set PNG_LIBRARY=D:\Library\lpng1613\build\Release
```

```
cmake -G "Visual Studio 12" ./
-DCMAKE_INCLUDE_PATH="%FTDIR%\include;%PTHREADDIR%;%JPEGDIR%;%JPEGDIR%;%ZLIB
DIR%"
-DCMAKE_LIBRARY_PATH="%FTLIBDIR%;%FTDIR%\lib;%JPEGDIR%\Release;%JPEGDIR%;%PN
G_LIBRARY%;%ZLIBDIR%\build\Release" -DPODOFO_BUILD_SHARED:BOOL=FALSE -DFREET
YPE_LIBRARY_NAMES_DEBUG=freetype253_D
-DFREETYPE_LIBRARY_NAMES_RELEASE=freetype253
```

앞의 내용을 입력해서 배치 파일을 생성하고 build.cmd 파일을 실행하면 솔루션 파일이 생성된다. FreeType, libjpeg, zlib, libpng 라이브러리의 헤더 경로와 라이브러리 경로를 제대로 맞춰줘야 하고, 플랫폼 타깃은 Visual Studio 2013이므로 'Visual Studio 12'를 선택한다. Podofo.sln을 실행해서 빌드하면 정상적으로 빌드되지 않는 프로젝트가 존재하는데, 이는 일본어 관련 스트링 문제 때문이다. 문자열을 제거하고 'aaa' 식으로 더미값을 넣어 빌드하면 정상적으로 컴파일된다.

샘플 코드

해당 솔루션에는 다양한 샘플이 존재하므로 샘플 프로젝트를 살펴보기 바란다. 여기서는 podofoimg2pdf 프로젝트를 살펴본다.

[이미지 파일을 pdf 문서에 포함하기]

```
int main( int argc, char* argv[] )
{
      char*    pszOutput;
      //출력 파일 이미지 사이즈 사용 여부 이미지 파일
      //output.pdf -useimgsize snap071.jpg
      if( argc < 3 )
      {
         print_help();
         exit( -1 );
```

```
    }

    pszOutput = argv[1];
    printf( " Output filename: %s\n " , pszOutput);
    //컨버터에 출력파일 이름을 설정하고 이미지 소스를 더한다.
    ImageConverter converter;
    converter.SetOutputFilename( pszOutput );
    for( int i=2;i<argc;i++ )
    {
       std::string sOption = argv[i];
       if( sOption == " -useimgsize " )
       {
          converter.SetUseImageSize( true );
       }
       else
       {
          printf( " Adding image: %s\n " , argv[i]);
          converter.AddImage( argv[i] );
       }
    }

    try {
       //등록된 이미지 리소스를 가지고 PDF 문서를 생성한다.
       converter.Work();
    } catch( PoDoFo::PdfError & e ) {
       fprintf( stderr, " Error: An error %i ocurred during processing the pdf
file.\n " , e.GetError() );
       e.PrintErrorMsg();
       return e.GetError();
    }

    printf( " Wrote PDF successfully: %s.\n " , pszOutput );
    return 0;
}
```

라이브러리명	xlsLib
분류	문서 작업 – 엑셀 라이브러리
설명	마이크로소프트의 엑셀 프로그램에서 생성한 문서는 확장자 xls를 가진 파일로 저장된다. 이 xlsLib 라이브러리를 사용하면 프로그래밍으로 쉽게 xls 파일을 생성할 수 있다.
다운로드	http://sourceforge.net/projects/xlslib/
버전	2.4.0

소스 빌드

소스 빌드 솔루션 파일이 존재하므로 소스는 쉽게 빌드된다. xlslib\build\msvc2008 경로로 이동해서 솔루션 파일을 실행한 다음 Visual Studio 2013 버전으로 마이그레이션한다. 디버그 모드에서는 정상적으로 빌드되지만 릴리스 모드에서는 추가 포함 디렉터리 경로 문제로 제대로 빌드되지 않는다. 이때 디버그 모드의 추가 포함 디렉터리 경로를 릴리스 모드의 추가 포함 디렉터리 경로에 그대로 복사하면 프로젝트가 정상적으로 빌드된다.

샘플 코드

샘플 예제의 xlsLibEx 프로젝트는 TSV 포맷의 파일을 읽어 데이터를 워크시트에 기록한 다음 엑셀 파일로 생성하는 방법을 보여준다.

```
int main() {
      //통합문서를 생성하고 워크 쉬트를 추가한다.
      workbook wb;
      worksheet* sh = wb.sheet( " sheet " );
      //test.tsv 파일을 읽어 들인다.
      vector< vector<string> > lines;
      ifstream input;
      input.open( " test.tsv " );
      if (!input) {
```

```
        cout << " Open error\n " ;
        return 1;
    }

    string line;
    while (getline(input, line)) {
        lines.push_back(splitLine(line));
    }
    cout << time(NULL) << " readed\n " ;
    //읽어 들인 문자열을 쉬트에 기록한다.
    for (int i = 0, l = lines.size(); i<l; i++) {
        for (int j = 0, s = lines[i].size(); j<s; j++) {
            sh→label(i, j, lines[i][j]);
        }
    }
    //통합 문서에 기록된 내용을 엑셀 파일로 저장한다.
    wb.Dump( " Real.xls " );
    return 0;
}
```

라이브러리명	libxls
분류	문서 작업 – 엑셀 라이브러리
설명	libxls 라이브러리도 엑셀 파일을 조작할 수 있다. 단, 이 라이브러리는 엑셀 파일을 읽을 수만 있고 쓰기 기능은 구현되어 있지 않다.
다운로드	http://libxls.sourceforge.net/
버전	0.2.0

소스 빌드

소스 파일이 많지 않으므로 수동으로 라이브러리 프로젝트를 생성해서 빌드한다.

샘플 코드

샘플 예제인 libxlsEx 프로젝트는 test.xls 파일을 읽는 방법을 보여준다.

```
int main(int argc, char *argv[])
{
      //통합 문서
      xlsWorkBook* pWB;
      //워크 시트
      xlsWorkSheet* pWS;
      //행 데이터
      st_row_data* row;
      WORD t, tt;

      //액셀 문서를 연다.
      pWB = xls_open( " test.xls " , " UTF-8 " );

      if (pWB != NULL)
      {
         //워크 시트 갯수만큼 루프를 돈다.
         for (int i = 0; i<pWB→sheets.count; i++)
```

```
        {
            printf( " Sheet[%i] (%s) pos=%i\n " , i, pWB→sheets.sheet[i].name,
pWB→sheets.sheet[i].filepos);

        //워크 시트를 얻어온 뒤
        //데이터를 파싱한다.
            pWS = xls_getWorkSheet(pWB, i);
            xls_parseWorkSheet(pWS);

            //워크 시트의 내용을 출력한다.
            printf( " Count of rows: %i\n " , pWS→rows.lastrow + 1);
            printf( " Max col: %i\n " , pWS→rows.lastcol);
            //각 행 정보를 출력한다.
            for (t = 0; t <= pWS→rows.lastrow; t++)
            {
               row = (st_row_data*)&pWS→rows.row[t];
               //마지막 열까지 루프를 돈다.
               for (tt = 0; tt <= pWS→rows.lastcol; tt++)
               {
                  //셀 값을 출력한다.
                  st_cell_data cell = row→cells.cell[tt];
                  printf( " cell=%s\n " , cell.str);
               }
            }
        }
    }
    return 0;
}
```

라이브러리명	audio_ostream(A Text-to-Speech ostream)
분류	문서 작업 – 텍스트의 오디오 변환
설명	audio_ostream 라이브러리는 텍스트를 음성으로 변환해 준다.
다운로드	http://www.codeproject.com/Articles/17897/audio-ostream-A-Text-to-Speech-ostream
버전	–

소스 빌드

마이크로소프트의 Speech SDK[01]를 설치해야 한다. 헤더 파일만 제공되므로 최종 애플리케이션에 같이 포함해서 빌드한다. 빌드 시 Speech SDK가 설치된 폴더의 헤더 경로를 설정하고 STLSoft C++ 라이브러리[02]를 내려받는다. 이 라이브러리도 헤더 파일로만 구성되어 있으므로 프로젝트에 이 헤더 파일 경로를 추가한다.

NOTE_

빌드하면 library_discriminator.hpp 파일에서 에러가 발생하는데, 이는 Visual Studio 2013 버전을 인식하지 못하기 때문이다. 해당 파일에 다음 내용을 추가한다.

```
#  elif _CPPLIB_VER <= 540
  /* Version 11.0 */
#   ifdef STLSOFT_COMPILE_VERBOSE
#    pragma message( " Dinkumware version 11.0 " )
#   endif /* STLSOFT_COMPILE_VERBOSE */
#   define STLSOFT_CF_STD_LIBRARY_DINKUMWARE_VC_VERSION
STLSOFT_CF_DINKUMWARE_VC_VERSION_11_0
#  elif _CPPLIB_VER <= 610
/* Version 12.0 */
```

01 http://www.microsoft.com/en-us/download/details.aspx?id=10121
02 http://sourceforge.net/projects/stlsoft/

```
#   ifdef STLSOFT_COMPILE_VERBOSE
#    pragma message( "  Dinkumware version 12.0 " )
#   endif /* STLSOFT_COMPILE_VERBOSE */
#   define STLSOFT_CF_STD_LIBRARY_DINKUMWARE_VC_VERSION
STLSOFT_CF_DINKUMWARE_VC_VERSION_11_0
```

샘플 코드

샘플 예제 중 TextToSpeechEx 프로젝트를 살펴보자. 프로그램을 실행하면 "Hello World"라는 음성이 들리는 걸 확인할 수 있다.

[Hello World 음성 출력]

```
int _tmain(int argc, _TCHAR* argv[])
{
      audio_ostream aout;
      aout << " Hello World! " << endl;

      getchar();
      return 0;
}
```