embarcadero

Using the Borland 5.5 Compiler and command-line tools

By: John Ray Thomas

Abstract: *How do I install the Borland 5.5 Compiler and command-line tools? This article takes a look at what's contained in the free download and shows how you can start building programs.*

Using the Borland 5.5 Compiler and command-line tools

The free Borland 5.5 Compiler and command-line tools has, so far, been a great success. At this time of writing,less that a month after we made it available, we have had hundreds of thousands of downloads.
I have also received hundreds of emails asking me how to install and use these tools but there is not really an install, per se. Rather, simply unzip the contents of the package and you are **almost** ready to go.

First, let's look at the directory structure. The root, by default is called BCC55. Under this directory you will find:

- Bin
- Examples
- Help
- Include
- Lib

Bin
Bin is short for binaries. Under this directory you will find all of the command-line tools (as well as RTL and STL dynamic libraries). These are 32-bit Windows exectuable, command-line programs, which means if you double click on one of them from Windows Explorer you are likely to see a flashing DOS box, that comes up and immediately goes away. These applications are meant to be run from within a DOS shell. Meaning, you need physically move to the Bin directory and type the name of the program that you want to run (unless this directory is first in your path).
For example, if I were to run the compiler bcc32.exe without any arguments I would get a dump of it's version and command-line options as shown.

```
[f:\borland\bcc55\bin]bcc32
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file[s] * = default; -x- = turn switch x off
```

-3	* 80386 Instructions	-4	80486 Instructions
-5	Pentium Instructions	-6	Pentium Pro Instructions
-Ax	Disable extensions	-B	Compile via assembly
-C	Allow nested comments	-Dxxx	Define macro
-Exxx	Alternate Assembler name	-Hxxx	Use pre-compiled headers
-Ixxx	Include files directory	-K	Default char is unsigned
-Lxxx	Libraries directory	-M	Generate link map
-N	Check stack overflow	-Ox	Optimizations
-P	Force C++ compile	-R	Produce browser info
-RT	* Generate RTTI	-S	Produce assembly output
-Txxx	Set assembler option	-Uxxx	Undefine macro
-Vx	Virtual table control	-X	Suppress autodep. output
-aN	Align on N bytes	-b	* Treat enums as integers
-c	Compile only	-d	Merge duplicate strings
-exxx	Executable file name	-fxx	Floating point options
-gN	Stop after N warnings	-iN	Max. identifier length
-jN	Stop after N errors	-k	* Standard stack frame
-lx	Set linker option	-nxxx	Output file directory
-oxxx	Object file name	-p	Pascal calls
-tWxxx	Create Windows app	-u	* Underscores on externs
-v	Source level debugging	-wxxx	Warning control
-xxxx	Exception handling	-y	Produce line number info
-zxxx	Set segment names		

Examples
The examples directory contains one directory called stdlib. In that directory are a number of source files that use various classes and algorithms available in the STL. Also, there is a makefile that builds all of the examples. More on that later.
Help
There is one Windows help file. It is called bcb5tool.hlp. You can double-click on it from Windows Explorer to read it. The following is a snapshot of one of the entries in the help file which explains what each of the command-line tools do.

File	Description
BCC32.EXE	C++ compiler (32-bit), command-line version
BPR2MAK.EXE	Converts project file from XML to MAKE file format for use with command-line tools
BRC32.EXE	Resource compiler (32-bit), command-line version
BRCC32.EXE	Resource shell (32-bit)
CPP32.EXE	C preprocessor (32-bit), command-line version
GREP.EXE	File search utility
ILINK32.EXE	Incremental linker (32-bit), command-line version
IMPDEF.EXE	Utility used when building apps with Libraries
IMPLIB.EXE	Utility used when building apps with Libraries
MAKE.EXE	Make utility
RLINK32.DLL	Resource linker (32-bit)
TDUMP.EXE	File dump utility
TLIB.EXE	Utility for maintaining static-link libraries
TLIBIMP.EXE	(Type Library Import) tool. Takes existing type libraries and creates C++Builder Interface files. Incorporated into the Type Library editor.
TOUCH.EXE	Change files stamps to current date/time
TRIGRAPH.EXE	Converts 3-character trigraph sequences into single characters

VCTOBPR.EXE Converts Microsoft Visual C++ 5.0 and 6.0 project (.DSP) and workspace (.DSW) files to their equivalent Borland C++Builder files

Include
This directory contains all of the header files for the Borland RTL, the STL (RW) and the Windows SDK.

Lib
This directory contains all of the static and import library files and startup code modules.

Putting it all together
So, now that you are armed with all this information you are probably wondering "How do I turn my source code into a program?" We will start with the simplest case of a single source file, console program. Here is the source code of a file called simple.cpp that I wrote in the text editor, notepad:

```
#include <stdio.h>  int main(void) {   printf("Output from running program");  return 0; }
```

To build this into a program we only need to call the compiler. However, the compiler needs to know what source file to build and where to find the header files and the library files. We pass these to the compiler as command line options as shown:
bcc32 -If:\Borland\bcc55\include -Lf:\Borland\bcc55\Lib simple.cpp

The resulting program is called simple.exe and can be run by typing *simple* at the command-line.

Now, let's look at the case of a console program with two source modules. simple.cpp will contain our entry point main and will call a function defined in the other module, funcs.cpp.

simple.cpp

```
#include "funcs.h"  int main(void) {   return PrintSomeOutput("Output from running program"); }
```

funcs.h

```
#include <stdio.h>  int PrintSomeOutput(char* output);
```

and funcs.cpp

```
#include <stdio.h>  int PrintSomeOutput(char* output) {        printf(output);         return 0; }
```

To build this, simply add funcs.cpp to the previous compiler command-line as such:
bcc32 -If:\Borland\bcc55\include -Lf:\Borland\bcc55\Lib simple.cpp funcs.cpp

So what happens if you have a bunch of different include and library directories. Or hundreds of source files. As you can imagine the command-line for this would be huge. You have two choices. Wrap up all of these commands into a batch file or use a makefile. Makefiles are prefferred, and next week we will delve into this. In the meantime, take a look at the makefile in the Examples\StdLib directory as we will be dissecting it.

Stay Tuned,
//jt

Published on: 3/3/2000 12:00:00 AM

Server Response from: ETNASC04