```
**************************************
BORLAND C++: ANSWERS TO COMMON QUESTIONS
**************************************
```

This file contains information about the following issues:

Getting Started

Q. How do I install Borland C++?
A. For help on a wide range of common installation issues, see the on-line document, INSTALL.TXT.

Q. What is a configuration file?
A. A configuration file tells Borland C++ what options to use as defaults and where to look for its library and header files. BCC.EXE looks for a file named TURBOC.CFG, and BCC32.EXE looks for a file named BCC32.CFG. The integrated environment, BCW.EXE, looks for the following default configuration files: BCCONFIG.BCW, BCWDEF.BCW and BCWDEF.DSW.

Q. How do I create a configuration file?
A. The INSTALL program creates TURBOC.CFG and BCC32.CFG for you. These file are ASCII files you can change with any text editor. They contain path information for the library and header files used by BCC.EXE and BCC32.EXE, respectively. In the case of the IDE (BCW.EXE), configuration files are created the first time you start the IDE. They are saved automatically each time you exit. You can turn this autosave feature off by selecting Options|Environment|Preferences. You can also explicitly save your options using Options|Save. The options listed on this menu and the files they correspond to are as follows:

Menu Option	File in BIN directory
Environment	BCCONFIG.BCW
Desktop	BCWDEF.DSW
Project	BCWDEF.BCW

Q. What is project file or configuration file corruption?
A. Project file corruption occurs when an error occurs during program development and a "bad" portion of memory gets saved to disk. This can later cause problems in the IDE. You might suspect that your project files have become corrupt if you experience strange new behavior either when compiling or

linking or if you are running very simple programs which don't behave as you would expect them to.

If you suspect that your project files have become corrupt, exit the IDE and delete the default IDE configuration files that are listed above.

You'll also want to check the files that the IDE uses for your specific project. This is especially true if the trouble you're having seems to happen in one project only. First, note what files are included in your project and what settings they use. Next, make a backup copy of your project file (.IDE) so that you can refer to it if needed (use the DOS COPY command). Give the backup an extension other than .IDE or .PRJ (.BAK is a good extension). Also look in your project directory for files with the same base file name as your project and with the extension of .BCW or .DSW . Delete or back up these files as well.

Once you have deleted or renamed the default configuration files and the files associated with your particular project, rebuild your project and try running your program again. If you're still having problems contact Borland Technical Support.

Q. How can I prevent project file corruption?
A. One way to help minimize the chance of corruption is to turn off the autosave feature by selecting Options|Environment|Preferences and unchecking the autosave checkboxes. If you do this before you exit the IDE each time, you'll need to explicitly save your project and options using Options|Save. Don't save the files if you experience any problems in the course of development.

Exception Handling

Q. How do I prevent the debugger from catching exceptions?
A. Go to the local option of Module View (press Alt+F10 or right-click in a Module View window). Press 'X' for Exceptions. Press 'N' to turn off C++ exceptions and 'o' to turn off C exceptions. Note that both types of exceptions must be disabled in order to stop the debugger from trapping exceptions.

Q. How do I step into the catch block of an exception?
A. You don't have to do anything special if you step over the throw() statement with either F7 or F8. You won't be able to step directly into the catch block if you step over a function which contains a throw statement. What you will have to do is place a breakpoint in the desired catch block. Note that this is what happens when attempting to step through new--the new default behavior of new is to throw an exception, xalloc, when an allocation fails.

Q. Why is the new operator behaving differently?
A. The default behavior of new in BC 4.5 is to throw an exception, xalloc, if the allocation fails. A NULL

value is no longer returned. The current error checking for new should look something like this:

```
char *temp;
try
{
       temp = new char[20];
}
catch(xalloc)
{
// Do error handling here; possibly assign NULL to temp
}
```

Q. How do I change the default behavior of new?

A. Making a call to set_new_handler() with a parameter of 0:

```
#include <new.h>
#include <except.h>

int main(void)
{
 char *temp
  set_new_handler(0);  // Change to old behavior of new
  temp = new char[25]; // returns NULL on failure
  delete temp;
  return 0;
}
```

Q. How can I change the default behavior of new for global objects? For example, where a global instance of a class calls new within its constructor?

A. Global objects are created before execution gets to the main function. Set up a startup function which gets called before the global objects are created. This is done using the #pragma startup statement:

```
#include <new.h>
void set_new(void)
{
  set_new_handler(0) ;
}
// global objects are created with a priority of 32,
// so use 31 as the priority for the function so that
// it gets called before global objects are created.
#pragma startup set_new 31
```

Q. How do I catch exceptions that are not explicitly handled by a catch block so that abort isn't automatically called?

A. Add a catch block which has ellipses as its parameter to every try block:

```
try
{
  throw(1);
}
catch(...)
{
  cout << "All exceptions are caught" << endl;
}
```

This ensures that there are no unhandled exceptions.

Q. What are some of the predefined exceptions that can be thrown?
A. There are a number of exception which are thrown from Borland functions (most of which are described in Chapter 10 of the Library Reference):

```
xalloc     - thrown from new.
Bad_cast   - thrown from dynamic_cast
Bad_typeid - thrown when the operand of typeid
             is a dereferenced 0 pointer.
```

Q. How can I display information when an exception goes unhandled?
A. The exception handling mechanism calls the terminate() function before calling abort(). This function does nothing by default, but a new function can be set using the set_terminate() function. The set_terminate() function takes as a parameter a function which returns void and does not take any parameters (void). You can display error messages within this function:

```
#include <except.h>
#include <new.h>
#include <iostream.h>

void term_func(void)
{
cout << "This exception went unhandled:" << endl;
cout << "  Name: " << __throwExceptionName << endl;
  // The following information is only valid if the
  // -xp (Enable exception location Information)
  // compiler switch was set. Under Options|Project|
  // C++ Options in the IDE.
cout << "  FileName: " << __throwFileName << endl;
cout << "  LineNumber: " << __throwLineNumber << endl;
exit(3);
}

int main(void)
  {
  set_terminate(term_func);
  return 0;
  }
```

Note that you have to call exit() from the terminate function in order to prevent the abort() function from being called (which in turn displays the "Abnormal Program Termination" message).

Q. How can I get information about an exception?
A. The object you throw can contain information that can be accessed within the catch block. You can also get information about where an exception was thrown by the use of certain global variables: __throwExceptionName, __throwFileName, and __throwLineNumber. These variables are documented in

Chapter 4 of the Library Reference. Note that you have to have 'Enable Exception Location Information' set in order to get the FileName and LineNumber (-xp from the command line, Options|Project|C++ Options in the IDE.) The following shows a simple method of getting additional information from an xalloc exception (which is thrown by new):

```
#include <cstring.h>
#include <except.h>

 int main(void)
{
 char huge *temp;
 try
  {
  temp = new huge char[1000000Ul]; // This will fail in DOS
  delete temp;
  }
 catch(xalloc one)
 {
  cout << "Allocation failed in main" << endl;
  cout << one.why().c_str() << endl;
  cout << "  Name: " << __throwExceptionName << endl;
    // The following information is only valid if the
    // -xp (Enable exception location Information)
    // compiler switch was set. Under Options|Project|
    // C++ Options in the IDE.
  cout << "  FileName: " << __throwFileName << endl;
  cout << "  LineNumber: " << __throwLineNumber << endl;
  }
 catch(...)
 {
  cout << "An unhandled exception occurred" << endl;
 }
 return 0;
}
```

The xalloc class contains a member function, why() that returns a cstring object. The cstring class has a function, c_str(), which returns a char *.

Q. Why does my application abort and get an "Abnormal Program Termination" message?

A. When an exception is thrown within an application but isn't caught, first the terminate() and then the abort() functions are called. It's the abort() function that displays the "Abnormal Program Termination" message. You can use the set_terminate() function to replace the existing terminate function (the terminate() function does nothing by default, but can be modified to display a message). Note that you need to call exit() from within the terminate() function if you don't want the "Abnormal Program Termination" message displayed. For more information, go to the section "How do I display information when an exception goes unhandled?"

Note that the new operator aborts on failure if the

xalloc exception isn't caught.

Q. Why is my program aborting with the "Abnormal Program Termination" message after I have turned off exception handling (-x-) for my modules?
A. The behavior of new isn't changed by turning off exceptions in your modules - it will still throw an exception upon failure. When this exception isn't caught, abort() is called. Call set_new_handler(0) to modify the behavior of new.

Q. Why does my application terminate when throwing an exception from within a destructor?
A. What is most likely happening is that the exception is thrown while the compiler is cleaning up the stack from a previously thrown exception. The compiler automatically cleans up objects on the stack when an exception is thrown. These objects on the stack can have destructors, which are also automatically called by the exception mechanism. Throwing an exception at any time within this cleanup isn't allowed.

Q. How do I pass an exception up the chain if I have nest try...catch blocks?
A. Using throw without any parameters within a catch block throws the exception up the chain:

```
try
{
   try
   {
     throw(10);
   }
   catch(int i)
   {
      cout << "Caught exception: " << i << endl;
      throw;
   }
}
catch(int i)
{
  cout << "Caught Exception: " << i << " again." << endl;
}
```

Q. Where can I get more information regarding exceptions?
A. Exceptions are covered in the following books:
"The Borland C++ 4.5 Programmers Guide"
"The Annotated C++ Reference Manual" by Ellis and Stroustrup, from Addison-Wesley Publishing Company.
"The C++ Programming Language, Second Edition" by Bjarne Stroustrup, from Addison-Wesley publishing Company.

Other Common C++ Questions

Q. When linking C or Assembly language modules with C++ modules I get undefined symbol errors at link time. It appears that none of the C or Assembly public symbols can be found.

A. C++ is a strongly typed language. In order to support the language to its fullest, Borland C++ must attach information to the symbols generated for function names and variables. When this is done, the symbol will no longer match the standard C style function name. In order to link correctly, the compiler must be notified that the symbol is declared in an external module without type information tacked on to the symbol. This is done by prototyping the function as type extern "C". Here is a quick example:

```
extern "C" int normal_c_func(float, int, char);
                                // name not altered
void cplusplus_function(int);  // name altered
```

See related comments under Linker Errors. There is also more on extern "C" in the Programmer's Guide, Ch 1.

Q: How can I allocate a doubly-dimensioned array?

A: You can use either malloc() with C or C++, or the new operator with C++:

malloc(): to create a 2 by 3 character array

```
int i;
char** p;
p = (char **) malloc(2);
for (i=0; i<2; i++) p[i] = (char *) malloc(3);
```

new:

```
int j;
char** q;
q = new char* [2];
       for (j=0; j<2; j++) q[j] = new char [3];
```

Q. Classes with static data members are getting linker errors ("undefined").

A. This code is built into Turbo C++ 1.0 but not in later versions of Borland C++. In the 1.0 compiler, static members without definitions were given a default value of 0. This default definition will no longer be made in the compiler. The programmer must now give an explicit definition for each static member. Here is a quick example:

```
class A
{
 static int i;
};
```

A linker error saying that A::i isn't defined results unless the source also contains a line such as:

```
int A::i = 1;
 //i needs to defined but not necessarily initialized
```

In the case of a template class, you need to similarly define static data outside the class definition, and also include the actual type information for any type you plan to instantiate the template class with. For example:

```
template <class T>
```

```
class A
{
       static int i;
};
A<int>::i;  //provide definition for an integer type
A<Myclass>::i; //provide definition for a user-defined type
```

Q. What potential problems can arise from typecasting a base class pointer into a derived class pointer so that the derived class's member functions can be called?
A. Syntactically this is allowable. There is always the possibility of a base pointer actually pointing to a base class. If this is typecast to a derived type, the function being called might not exist in the base class. Therefore, you would be grabbing the address of a function that doesn't exist.

Q: What's the difference between the keywords STRUCT and CLASS?
A: The members of a STRUCT are PUBLIC by default, while in CLASS, they default to PRIVATE. They are otherwise functionally equivalent.

Q: I declared a derived class from a base class, but I can't access any of the base class members with the derived class function.
A: Derived classes DO NOT get access to private members of a base class. In order to access members of a base class, the base class members must be declared as either public or protected. If they are public, then any portion of the program can access them. If they are protected, they are accessible by the class members, friends, and any derived classes.

Q: I have a class that is derived from three base classes. Can I ensure that one base class constructor will be called before all other constructors?
A: If you declare the base class as a virtual base class, its constructor will be called before any non-virtual base class constructors. Otherwise the constructors are called in left-to-right order on the declaration line for the class.

Q: Are the standard library I/O functions still available for use with the C++ iostreams library?
A: Yes, using #include <stdio.h> functions such as printf() and scanf() are available. However, using them in conjunction with stream-oriented functions can lead to unpredictable behavior.

Q: When debugging my program in Turbo Debugger, I notice that none of my inline functions are expanded inline. Instead, they are all done as function calls.
A: Whenever you compile your program with debugging information included, no functions are expanded inline. To verify that your inline functions are indeed expanding inline, compile a module with the

-S option of the command-line compiler, then examine the .ASM file that is generated.

Q. In C++, given two variables of the same name, one local and one global, how do I access the global instance within the local scope?
A. Use the scope (::) operator. For example:

```
int x = 10;
for(int x=0; x < ::x; x++)
{
   cout << "Loop # " << x << "\n";
   // This will loop 10 times
}
```

Q. If I pass a character to a function which only accepts an int, what will the compiler do? Will it flag it as an error?
A. No. The compiler promotes the char to an int and uses the integer representation in the function instead of the character itself. The exception here is in the case of a template function where no implicit conversions or promotions take place.

Q. I was trying to allocate an array of function pointers using the new operator but I keep getting declaration syntax errors using the following syntax:
`new int(*[10])();` What's wrong?
A. The new operator is a unary operator and binds first to the int keyword producing the following:

```
(new int) (*[10])();
```

You need to put parentheses around the expression to produce the expected results:

```
new (int (*[10]()); //array of function pointers
```

Q. What are inline functions? What are their advantages? How are they declared?
A. An inline function is a function which gets textually inserted by the compiler, much like macros. The advantage is that execution time is shortened because linker overhead is minimized. They are declared by using the inline keyword when the function is declared:

```
inline void func(void) {cout << "printing inline function \n";}
```

or by including the function declaration and code body within a class:

```
class test
{
public:
void func(void) {cout << "inline function within a class.\n"}
};
```

Q. If I don't specify either public or private sections in a class, what is the default?
A. In a class, all members are private by default if neither public nor private sections are declared.

Q. If I don't specify either the public or private keyword when inheriting from a base class, what is the default?

A. In Borland C++ 2.0, the default was public inheritance,
but in versions 3.0 through 4.5, the default is private
inheritance.

Q. What does the _seg modifier do?
A. Using _seg causes a pointer to become a storage place for a
segment value, rather than an offset (or a segment/offset).
For instance, if "int _seg *x" contains the value 0x40,
then when you use "*x", the value pointed to will be at
segment 0x40, offset 0. If you add a value to the pointer,
the value is multiplied by the size of the pointer type. That
new value is used as an offset and is combined with the
segment value contained in the pointer. For example,

```
int _seg *x;
int value;

x = (int _seg *)0x40;
value = *(x + 20);
```

value is assigned the value of the integer at 0x40:0x28
(Remember, 20 * sizeof(int) = 40 = 0x28).
You can find a more detailed description of _seg in the
Borland C++ DOS Reference, Chapter 1.

Q. Can I statically allocate more than 64K of data in
a single module?
A. Yes. Far data items are now supported:

```
...
char far array1[60000L];
char far array2[60000L];
...
```

For arrays larger than 64k use:

```
char huge array3[100000L];
```

Q. What is a friend member function?
A. Declaring a friend gives non-members of a class access
to the non-public members of a class.

Q. Why do I get a "Type name expected" error on my definition
of a friend class in my new class?
A. You need to let the compiler know that the label you use
for your friend class is another class. If you don't
want to define your entire class, you can simply have
"class xxx", where xxx is your label. For example:

```
class Myclass1; //forward class declaration is required

class Myclass2{
      friend class Myclass1;
      ...
};
```

Q: How can I output hex values in uppercase using the
iostream libraries?
A: You need to set the state of the stream using setf().
For example,

```
#include <iostream.h>
```

```
int main(void)
{
  cout << hex;
  cout << "\nNot upper-case : " << 255;
  cout.setf(ios::upper-case);
  cout << "\nUppercase     : " << 255;
  return 0;
}
```

Q: The following program compiles/links successfully in C but not in C++. Why?

```
#include <stdlib.h>

int compare(const int *one, const int *two)
{
   if (*one > *two)
      return  -1;
   else
      return 1;
}
int a[3] = { 50, 10, 20 };

void main()
{
     qsort(a, 3, sizeof(a[0]), compare);
}
```

A: The fourth parameter to compare is the function pointer, and here's how it's declared in stdlib.h:

```
void qsort (void *__base, size_t __nelem, size_t __width,
        int _Cdecl (*__fcmp)(const void *, const void *));
```

However, the above program WILL NOT compile in C++, because of the strong typing features of the C++ language. The compiler refuses to convert the void parameters in the declaration to __fcmp function to int parameters. However, because C++ permits casting of function pointers, you can fix the call to QSORT in C++ like this:

```
qsort(a, 3, sizeof(a[0]),
      (int (*)(const void *,const void *))compare);
```

By casting the COMPARE function to be of the same type as the declaration in stdlib.h, C++ will accept and compile it.

Q. What is the "this" pointer?

A. "this" is a local variable in the body of a non-static member function. It is a pointer to the object for which the function was started. It cannot be used outside of a class member function body.

Q. Why does a binary member function only accept a single argument?

A. The first argument is defined implicitly.

Q. Looking through the class libraries there are definitions in classes which look like:

```
      class test {
          int funct(void) const;
      };
```
   What is the const keyword doing here?
A. There is a pointer to the object for which a function
  is called known as the 'this' pointer. By default the
  type of 'this' is X *const (a constant pointer). The
  const keyword changes the type to const X *const (a
  constant pointer to constant data).

Q: How can I use _new_handler and set_new_handler?
A: Borland C++ supports _new_handler and set_new_handler.
  You can find a discussion of them in Chapter 3 of the
  Borland C++ Programmer's Guide. The type of _new_handler
  is as follows.

```
     typedef void (*vfp)(void);
     vfp _new_handler;
     vfp set_new_handler(vfp);
```

Q: I would like to use C++ fstreams on a file opened
  in binary mode. How is this done?
A: Use ios::binary as the open mode for the file:

```
     #include <fstream.h>
     ifstream binfile;
     binfile.open("myfile.bin", ios::binary);
```

Q: How can I get at the DOS file handle associated
  with my iostream?
A: Using a combination of member functions fd() and
  rdbuf() you can get at the file handle.

```
     #include <fstream.h>
     #define fstrno(s)  (((s).rdbuf())->fd())
     ifstream test("test.txt");
     cout << "handle is " << fstrno(test) << '\n';
```

Q: How can I increase the number of FILES available to
  my program under Borland C++ 4.5?
A: Increasing the number of available files involves
  changing the following 3 files located in the Run-Time
  Library: _NFILE.H, FILES.C, and FILES2.C. For
  instructions on how to do this, download TI870 from
  the Borland fax line.

Q: When using the BIDS library, if I try to create an
  Array or Bag of integers, I get the error "multiple
  declaration for detach()...".
A: If you try to create a TI_ArrayAsVector<int>,
  TI_SArrayAsVector<int>, or TI_BagAsVector<unsigned>,
  the compiler yields an error message. This is because
  each of these templates provides a version of detach()
  that takes an argument whose type is the type for
  which the template is being instantiated, plus a
  version that takes an argument of type int or
  unsigned int, which lets you specify the index for the
  item to be detached. Therefore, when instantiated with
  an integer, these two versions have the same signature,
  which is what causes the error. Integer types aren't

allowed here due to resulting ambiguity between the type and the index of the Bag or Array.

Q: How come bioscom() doesn't work on my computer?
A: The bioscom() function uses DOS interrupt 0x14 directly, and thus bioscom()'s functionality is tied directly to the BIOS of your computer. MS-DOS support for the serial communications port might be inadequate in several respects for high-performance serial I/O applications. First, MS-DOS provides no portable way to test for the existence or status of a particular serial port in a system. If a program "opens" COM2 and writes data to it, and the physical COM2 adapter isn't present in the system, the program may simply hang. Similarly, if the serial port exists, but no character has been received and the program attempts to read a character, the program hangs until one is available. There is no traditional function call to check if a character is waiting. MS-DOS also provides no portable method to initialize the communication adapter to a particular baud rate, word length, and parity. An application must resort to ROM BIOS calls, manipulate the hardware directly, or rely on the user to configure the port properly with the MODE command before running the application that uses it. Because of all the problems mentioned above, we strongly recommend getting a third party communications package when attempting to do serial communications, or downloading the example program SERIAL.ZIP from our BBS at (408) 439-9096.

Common Windows and OWL Questions

Q: How do I convert my old OWL 1.0 to OWL 2.5?
A: You can use a utility called OWLCVT to help you do this. This utility is documented in the ObjectWindows for C++ Programmer's Guide, Appendix A.

Q: How do I get OWL 1.0 to work w/ BC 4.5?
A: The on-line file, COMPAT.TXT, provides information on doing this.

Q. Why isn't my DLL working correctly?
A. One possibility is that you are not linking in the correct 'cw' library. If you are building a small model DLL, you should be using cwc, not cws. If you are building a medium model DLL, you should be using cwl, not cwm. Compact and large models should use cwc and cwl respectively.

Q. Why isn't my program working correctly? I'm getting a message box from Windows saying "Unrecoverable Application Error".
A. One possible answer is that the program was not built correctly. For example, the linker didn't get the correct information to export functions. Diagnostic messages from

the linker could indicate this. To check that you have built your program on DLL as expected, review the section in Chapter 8 of the Programmer's Guide that deals with exports. This section has a table with 8 columns describing the possible combinations you might have used to build your program. If the setup of your program corresponds to one of the last three columns, chances are that your program was not built correctly (or, at least, as you intended). Column 5 corresponds to the 'classical' method of building Windows programs (that is, all exports are declared in the module definition file (the .def file)).

The columns that use -WE or -WDE will build 'better' code in the sense that the compiler won't make 'exportable' any functions that it doesn't actually export. However, it is here that many people run into problems. If you have any functions declared as exports in the .def file but the module is compiled with -WE or -WDE, then you probably have built the program incorrectly (the function will be exported only if it is preceded by _export in the source code).

Q. How do I use the _export key word?
A. Put the "_export" immediately before the function name in the function declaration to export that function. Here is a quick example:

```
long FAR PASCAL _export func(void) {...}
```

Q. I run BCW.EXE and get the error message:
Fatal: <filename>.def (<line #>): syntax error
A. Check your DATA statement on line number # in <filename>.DEF for the correct code (DATA PRELOAD).

Q. Why do I get a 'suspicious pointer conversion' warning or 'cannot convert' error (in C++ code) when I try to use the address of my window procedure or call back function?
A. Windows 3.1 has introduced a new type, WNDPROC, which takes the place of the FARPROC type in some cases, such as the data type of the lpfnWndProc member of the WNDCLASS structure. If you are getting a warning or error when setting up a WNDCLASS structure that is passed to RegisterClass(), use a WNDPROC cast to resolve the type mismatch. If you were previously using a FARPROC cast, simply change it to a WNDCLASS cast. For example,

```
void FAR PASCAL f(void);
WNDCLASS wcTemp;
wcTemp.lpfnWndProc = f;
            // Warning in C or error in C++
wcTemp.lpfnWndProc = (FARPROC)f;
            // Windows 3.0 style type cast
wcTemp.lpfnWndProc = (WNDPROC)f;
            // Windows 3.1 style type cast
```

Q. How can I use the features of Windows 3.1 in my

applications?
A. There are a number of examples that use the features of Windows 3.1, such as OLE, DDEML and Common Dialogs. There are also lengthy descriptions of programming techniques for the features of Windows 3.1 in the online Help files.

Q. Why don't some of the Windows 3.1 API functions, such as ChooseColor(), do anything when I call them?
A. The Windows 3.1 functions that take structures as parameters require that the size field of the structure be initialized to the size of the structure. This technique allows for backward compatibility in future versions of these functions. If the size field isn't set correctly the function won't do anything. For example,

```
CHOOSECOLOR ccTemp;                // Data structure
ccTemp.lStructSize=sizeof(ccTemp); // Set the size first!
if(ChooseColor(&ccTemp)!=0) etc... // Then call the function
```

Q. Why is my DDEML application crashing?
A. DDEML can crash in seemingly random ways if the conversation handle or application instance identifier is incorrect or corrupted. Use TDW to watch the value of the conversation handle and the application instance identifier. If it changes, is corrupted, or you inadvertently pass the DDE Management Library an invalid value, that particular call might not fail but DDEML might become unstable and crash at some time in the future. Also note that before DdeInitialize() is called for the first time, the application instance identifier argument MUST be set to 0.

Integrated Environment and Resource Workshop

Q: How do I use Source Pools?
A: A sample project called srcpool.ide in the \examples\ide\srcpool subdirectory shows you how.

Q: How can I create multitarget programs?
A: An example project called multitarg.ide in the \examples\ide\multitarg subdirectory show you how.

Q: How can I take advantage of Style Sheets?
A: A sample project called stylsht.ide in the \examples\ide\stylsht subdirectory shows you how.

Q. Why can't Borland C++ find any of my #include files?
A. The compiler searches for include files in the Include Directories path. You can specify this path by selecting Options|Project|Directories. The INSTALL program initially sets this path to the directory where it copied all the Borland C++ *.h files.

Q. How do I get Borland C++ to link in my own libraries or use multiple source files?
A. Borland C++'s Project Manager is designed to let you

work with multiple files.

Q. Why does Borland C++ report "Unable to open include file 'stdarg.h'" when I try to #include <stdio.h>?
A. The most probable reason is that you need to check to ensure you have correctly specified the path to your include directories by selecting Options|Project|Directories and examining the Source Directories|Include section.

One other possible reason is that you have exceeded the number of files that DOS can have open simultaneously. Add the line FILES=40 to your DOS CONFIG.SYS file. This lets DOS open up to 40 files at the same time. CONFIG.SYS will only be effective after you have rebooted your computer. See the IBM DOS Reference Manual for details on the CONFIG.SYS file.

Q. When I Make, Run, or Trace a program, Borland C++ sometimes goes through the compile and link process even when the object files are up-to-date.
A. Borland C++'s MAKE logic works solely on a file's date and time stamp. If one of your source files is marked with a date that's sometime in the future, the object files that are created from it will always be older than the source file, and Borland C++ will always try to rebuild the file. You can fix this by using TOUCH.COM to set the file to the current date and time. You should also make sure that your system's date and time are always properly set. TOUCH.COM is documented in the User's Guide chapter on MAKE.

Q. How can I convert my earlier project files to the new IDE format?
A. Using Project|Open, select "3.1 Project Files (*.prj)" into "List Files of Type" drop-down box. Select the 3.1 project you want to run--the project is automatically converted into *.ide format. The original *.prj file remains unchanged. Be sure to save the new *.ide file.

Q. How can I find out where my "null pointer assignment" is occurring?
A. Set a watch on the following expressions:

```
*(char *)0,4m
(char *)4
```

Step through the program. When the values change, the just-executed line is the one causing the problem.

Q. When I try to load a new file after editing a file, the first file remains on the screen. How do I close the first file?
A. Use Ctrl-F4 to close the current file.

Q. I'm doing a search and replace operation and the editor prompts me for each replacement. I've selected "Change All", but it still does it.

A. To disable the prompting, you must unselect the "Prompt on replace" option on the left side of the Search dialog box.

Q. When I try to use any of the pseudo registers, like _AX, I get the error message "Undefined symbol '_AX' in function..." when I compile. Why?
A. You are only allowed to use the pseudo registers in the Borland C++ and ANSI modes of the compiler. You can change this setting by selecting Options|Project|Compiler|Source.

Q: How do I stop all of the files I have ever edited from constantly being open when I start Borland C++?
A: By default, Borland C++ saves what is called the desktop configuration. This configuration is saved in a file with a .DSW extension. By deleting any files of this type (usually located in the current directory and/or the BC45\BIN directory), then entering Options|Environment|Preferences and unchecking the 'auto save desktop' option, you'll begin with a clean desktop each time you start Borland C++.

Q: How do I view 32-bit registers in the debugger?
A: Add the appropriate 32-bit register psuedovariable to the Watch window. For example, if you were interested in the value of EAX, you'd add _EAX to the Watch window.

Q: Why does Resource Workshop say "Cannot open file" on a header file I included in my .RC or .DLG file?
A: Either the file doesn't exist, or Resource Workshop doesn't know what directory to look for it. See if the file exists. If it does, then add the directory name to the Include Path under File|Preferences.

Q: Why is the Include Path disabled when I look under File|Preferences in Resource Workshop?
A: The Include Path is disabled when a project is opened. Close the project and then change the path.

Q: In Resource Workshop, why can I select only bold fonts when I'm in the dialog editor?
A: This is a limitation of Windows, not Resource Workshop. Currently, Windows allows only bold fonts in dialog templates. You can send WM_SETFONT messages while your program is running if you want other fonts or styles.

Q: Why does my dialog paint funny when I select Test mode? The menu and/or the minimize/maximize buttons don't draw correctly.
A: Again, this is a limitation of Windows. There are certain styles for a dialog box that shouldn't be combined. Namely, the Modal Frame style can cause painting problems when combined with the Min/Max buttons or menus.

Q: I did everything the documents suggested, but
  Resource Workshop still doesn't work or behaves
  strangely. Why?
A: Look for any .RWS files and delete them. These files
  are maintained by Resource Workshop and can cause
  unusual behavior in Resource Workshop if they become
  corrupted or out-of-sync with the .RC or .DLG files.

Command-Line Compilers

Q. Why can't Borland C++ find any of my #include files?
A. The compiler searches for include files in the Include
  Directories path. You specify this path with the -I
  option or with Options|Project|Directories in the IDE.
  The INSTALL program initially writes a configuration
  file (TURBOC.CFG for BCC.EXE, BCC32.CFG for BCC32.EXE)
  that sets this path to the directory where it copied all
  the Borland C++ *.h files. You can edit this file to
  change the default path, or create this file in your
  current working directory and place any relevant paths
  and other command line compiler options in it. A sample
  .cfg file might contain the following:

```
-ml
-IC:\BC45\INCLUDE;C:\BC45\CLASSLIB\INCLUDE
-LC:\BC45\LIB;C:\BC45\CLASSLIB\LIB
```

Q. Why does the linker tell me that all the graphics
  library routines are undefined?
A. BCC won't search the graphics library unless you
  tell it to. You should specify the graphics library
  on the command line. For example, to compile BGIDEMO,
  type BCC BGIDEMO.C GRAPHICS.LIB and press Enter.

Q. I run BCC.EXE and get the error message:
  Fatal: <filename>.def (<line #>): syntax error
A. Check your DATA statement on line number # in
  <filename>.def for the correct code (DATA PRELOAD).

General I/O

Q. The '\n' in cprintf() doesn't return the cursor to the
  beginning of the line--it only moves it down one line.
A. cprintf() interprets '\n' as a Line Feed. To force the
  cursor to the beginning of the line, manually insert a
  Carriage Return:

```
cprintf("\n\r");
```

Q. How do I print from a Borland C++ program?
A. Borland C++ uses a FILE pointer (stdprn) defined in the
  STDIO.H file. You do not need to open stdprn before
  using it:

```
#include <stdio.h>
int main(void)
{
    fprintf(stdprn, "Hello, printer!\n");
}
```

Note that if your printer is line-buffered, the output is flushed only after a '\n' is sent.

Q. I'm reading and writing binary files. My program is translating the Carriage Return (0x0D) and Line Feed (0x0A) characters. How do I prevent this from happening?
A. Files opened in text mode will translate these characters for DOS. To read a file in binary mode, open it in binary mode. For example,

```
#include <stdio.h>
int main(void)
{
  FILE *binary_fp;
  char buffer[100];

  binary_fp = fopen("MYFILE.BIN", "rb");
  fread(buffer, sizeof(char), 100, binary_fp);
              :
}
```

The default file mode is text.

Q. Why don't printf() and puts() print text in color?
A. Use the console I/O functions cprintf() and cputs() for color output.

```
#include <conio.h>
int main(void)
{
   textcolor(BLUE);
   cprintf("I'm blue.");
}
```

Q. How do I print a long integer?
A. Use the "%ld" format:

```
long int l = 70000L;
printf("%ld", l);
```

Q. How do I print a long double?
A. Use the "%Lf" format.

```
long double ldbl = 1E500;
printf("%Lf", ldbl);
```

Example Programs

Q. I've loaded and run one of the example programs and it doesn't work. What's going on?
A. The most common cause for this behavior is loading the .C or .CPP file for the example, which won't necessarily include everything the program needs to build and run correctly. The best method for building most of the examples is to use an .IDE file/MAKEFILE if one is provided for that example. The following paragraphs describe how to do this.

Q. How do I compile and link the examples from the IDE?
A. Use Project|Open to open the project file (*.IDE)

associated with the program(s) you want to run. Use Compile|Build All to build the project and Debug|Run to see the program output. See the Borland C++ User's Guide for more information on using the IDE.

Q. How do I compile and link the examples at the command line?
A. Go to the directory that contains the example(s) you want to run and type "MAKE". When make has completed building your program, go to the Windows Program Manager, select "RUN", and enter the path and file name of your program. (If it is a DOS program, it can be run from the DOS prompt as usual.)

Q. How can I change how a program is built when I use MAKE?
A. If you look in the MAKEFILE in the directory for the example you're working with, the beginning of this file frequently specifies what parameters you can pass to MAKE.EXE to control how the example is built. For example, you might see that using MAKE DEBUG=1 includes debugging information letting you step through the program, etc.

Q. I'd like to know more about the examples without building each one individually. How can I learn more about them?
A. A Windows help file that indexes the example programs is available on the Borland Download BBS at (408) 431-5096. The file and instructions for installing it are included in BC40EXAM.ZIP.

Graphics (DOS)

Q. Why do I get the error message:
BGI Error: graphics not initialized (use 'initgraph')
when I use a graphics function? My program has already called initgraph().
A. For some reason initgraph() failed. To find out why, check the return value of graphresult(). For example:

```
   #include <graphics.h>
   int main(void)
   {
     int gerr;   /* graphics error */
     int gdriver = DETECT, gmode;

  // Initialize graphics using auto-detection and look
  // for the .BGI and .CHR files in the C:\BC45\BGI
  // directory.
     initgraph(&gdriver, &gmode, "C:\\BC45\\BGI");

     if ((gerr = graphresult()) != grOk)
     {
         printf("Error : %s\n", grapherrormsg(gerr));
         exit(1);
     }
           :
   }
```

Math and Floating Point

Q. Why do I get incorrect results from all the math library functions like cos(), tan() and atof()?

A. You must #include <math.h> before you call any of the standard Borland C++ math functions. In general, Borland C++ assumes that a function that isn't declared returns an int. In the case of math functions, they usually return a double. For example

```
/* WRONG */                      /* RIGHT */
                                 #include <math.h>
int main(void)                   int main(void)
{                                {
 printf("%f", cos(0));            printf("%f", cos(0));
}                                }
```

Q. How do I "trap" a floating-point error?

A. See the signal() and matherr() functions in the Borland C++ Library Reference. The signal() function might be used to trap errors in the 80x87 or the 80x87 emulator. The matherr() function traps errors in the Math Library functions.

Linker Errors

Q. I am linking C functions with C++ functions. The linker reports that all of my C functions are undefined. Why?

A. Linking C++ modules with C modules requires the use of a linkage specification. Prototypes for C functions within C++ modules must be in one of the following forms:

```
extern "C" declaration
extern "C" { declarations }
```

For example, if a C module contains functions "char *SCopy(char*, char*);" and "void ClearScreen(void)", they must be declared in a C++ module in one of the following ways:

```
extern "C" char *SCopy(char*, char*);
extern "C" void ClearScreen(void);
```

or

```
extern "C" {
    char *SCopy(char*, char*)
    void ClearScreen(void);
}
```

For further examples, see the standard header files. For additional comment, see Common C++ Questions.

Q. Why do I get the message:
"Linker Error: Unable to open input file 'C0x.OBJ'"

A. The linker searches for Borland C++ start-up and library files in the Borland C++ Library Directories path. You can specify this path by selecting the Options|Directories. The INSTALL program initially sets this path to the directory where it copied the start-up and library files. Also be sure that you installed the memory model that the linker is looking for. The 'x' in the error message corresponds to the memory model, e.g. 's'

for small, 'l' for large, etc.

Q. Why do I get the message:
Linker Error: Undefined symbol '_main' in module C0
A. Every C program must contain a function called main(). This is the first function executed in your program. The function name must be all in lower case. If your program doesn't have one, create one. If you're using multiple source files, the file that contains the function main() must be one of the files in the Project. Note that an underscore character '_' is prefixed to all external Borland C++ symbols.

Q. Why does the linker tell me that all the graphics library routines are undefined?
A. See the "Integrated Environment" and "Command-line Compiler" sections above.

Q. What is a 'Fixup overflow'?
A. This usually means you're attempting to link object files that weren't all compiled under the same memory model. See the listing of TLINK error messages in the User's Guide. Publication TI1150, "Coping with Fixup Overflow Messages", provides more information. It's available as TI1150.ZIP on Borland's Download BBS (408)431-5096, and as document # 1150 on Borland Techfax (800) 822-4269.

Q. I'm linking my own assembly language functions with Borland C++. The linker reports that all of my functions are undefined.
A. Make sure that you have put an underbar character '_' in front of all assembly language function names to be called by Borland C++. Your assembly language program should be assembled with Case Sensitivity. If compiling as C++ (rather than C), see the "Common C++ Questions" section above for a discussion of extern "C".

Q: I'm getting an error out of the linker "segment group exceeds 64K : _text".
A: If you're using the BGIOBJ utility, the default segment into which the objects will be place is _text. You should try using BGIOBJ with the /f option to place the resultant objects into a separate segment. You'll then need to use the functions registerfarbgidriver and registerfarbgifont to register the objects for the graphics system. See UTILS.TXT for instructions on using these functions. In addition, publication TI703, "Resolving 'Segment or Group xxxx Exceeds 64K', provides more information on this issue. It is availble as TI703.ZIP on Borland's Download BBS (408)431-5096, and as document #703 on Borland Techfax (800) 822-4269.

Q: Why am I getting the error "printf: floating point formats not linked"?
A: You probably have your libraries out of order on the

TLINK command line. (See the User's Guide for TLINK syntax.) For example, if you are using the large memory model and BGI routines, the TLINK line might look like the following:

```
tlink /v c0l myobj,,,mylib graphics emu mathl cl
```

If the object file or library isn't in the current directory, the complete pathname must be supplied. Frequently this causes the command line to exceed 128 characters (you'll need to use a response file--See the User's Guide).

Q. I'm porting an application that uses communal variables to C++. I've set up the compiler to recognize them, but I still get linker errors: Error: <name> defined in module <a> is duplicated in module <b>

A. C++ doesn't support explicit COMDEFs; you must use static variables or switch to C.

Other Questions

Q: How can I use Paradox Engine the with Borland C++ 4.5?

A: For information about using the Paradox engine with Borland C++ 4.5, see the online document, COMPAT.TXT.

Q. How can I make use of my existing Turbo Vision 1.0 code with Borland C++ 4.5?

A: Information on using the Turbo Vision 1.0 with Borland C++ 4.5 is given in the online document, COMPAT.TXT. Turbo Vision 2.0 is now available as part of the Borland PowerPack for DOS. This version can be used to create 16-bit and 32-bit DPMI applications allowing you to break the 640K barrier by accessing extended memeory.

Q. I get a "floating point formats not linked" message when I run my program. What can I do about it?

A. Floating point formats (for scanf() and related functions) aren't always linked, for savings in executable size. To force their inclusion, put the following somewhere in your source files:

```
extern int _floatconvert;
#pragma extref _floatconvert
```

Q. How do I change the stack size?

A. In a DOS program, the size of the stack of a Borland C++ program is determined at run time by the global variable _stklen. For example, to change the size to 10,000 bytes, include the following line in your program:

```
extern unsigned _stklen = 10000;
```

This statement must not be inside any function definition. The default stack size is 4,096 bytes (4K), and you might increase the stack to 65519 (0xFFEF) or just under 64K in the compact, large, or huge memory models.

In a Windows Program, the size of the stack is controlled by the STACKSIZE line of the module definition (.DEF) file. See the Programmer's Guide and the User's Guide for

more information on module definition files.

Q. I'm getting a 'Stack Overflow!' message when I run my program. How can I work around this?

A. If you are using the compact, large, of huge memory models, you might increase the stack size by following the procedure above. In the smaller memory models, your only option is to decrease the amount of stack space or near heap space used in your program. Stack overflows are usually caused by a large amount of local data or recursive functions. You can decrease the amount of stack space used in several ways:

1) By declaring your local variables static (see the Programmer's Guide for the effects of using the "static" keyword):

```
int main(void)                int main(void)
{                             {
 char x[5000];     -->           static char x[5000];
    :                               :
}                             }
```

2) By making your variables global rather than local:

```
char x[5000];    //global allocation above main()
int main(void)
{
       :
}
```

3) By allocating your variables dynamically off the far heap:

```
#include <alloc.h>
int main(void)
{
 char far* x;
 x = (char far*)farmalloc(5000);  //dynamic allocation
 // or in the case of C++ you can use the new operator
 // x = new char[5000];
     :
}
```

Q. My program comes up with the message 'Null pointer assignment' after it terminates. What does this mean?

A. Before a small-data model Borland C++ program returns to DOS, it checks to see if the beginning of its data segment has been corrupted. This message warns you that you have used uninitialized pointers or that your program has corrupted memory in some other way.

Q. Why do I get "declaration syntax error" messages on dos.h?

A. You have set the "ANSI keywords only" option ON. Keep this option OFF when using any keywords specific to Borland C++. See the Programmer's Guide for a list of keywords.

Q. I get errors when compiling the windows.h header file. Why?

A. Be sure that you have "Borland Extensions" selected as your keywords option. This option can be toggled under Options|Project|Compiler|Source in the IDE. It is on by default in the command-line compilers (-AT or -A-).

Q. I have a working program that dynamically allocates memory using malloc() or calloc() in small data models (tiny, small, and medium). When I compile this program in large data models (compact, large, and huge), my program hangs.
A. Make sure that you have #include <alloc.h> in your program.

Q. I'm linking my own assembly language functions with Borland C++, but the linker reports that all of my functions are undefined. Why?
A. See answer above in the "Linker" section.

Q. My far pointers "wrap around" when they are incremented over 64K. How do I reference a data object that is greater than 64K?
A. Use huge pointers.

Q. How do I interface BC++ routines to a Turbo Pascal program?
A. See the example programs contained in CPASDEMO.ZIP on the Borland Download BBS (408) 431-5096.

Q. How do I get Clipper to link with Borland C++?
A. If you have trouble, contact Nantucket Technical Support.

Q. I'm trying to build an app based on one of Borland's libraries (ObjectWindows, Turbo Vision, the container classes in the CLASSLIB directory, or the Runtime Library), and I get linker errors, or it won't run right. What's going wrong?
A. You might be using a switch that affects linkage in your files that wasn't used when the library itself was compiled, or you need to change the library in question. Here are some examples:
- If you use far vtables (-Vf or Options|Project|Compiler|C++|Far virtual tables) to compile a file you developed which includes iostream.h, it won't build correctly until you rebuild the iostream library with the same option.
- If you use word alignment (-a or Options|Compiler|Code Generation|Word alignment) in building a Turbo Vision application, you must build the Turbo Vision library from source with the same option.
- If you opt to use the templates implementation of the container class library to build ObjectWindows applications, you must rebuild the necessary ObjectWindows libraries from source using the templates implementation of the class library (the BIDxxxx.LIB files.)

Q. I got a "bad call to intrinsic function" message when compiling one of my source files. What does this mean?
A. This message appeared because you tried to use an intrinsic function in a small model DLL. Either avoid using intrinsic functions in your DLL, or turn off the -Oi and -O2 switches (Options|Project|Optimizations Speed|Inline Intrinsic Functions).

Q: I open up a file in append mode and append some data

to the end of the file. When I look at the data in an ASCII editor, I can't see the appended data. Why?

A: The data is being appended after the End-of-File mark, and the ASCII editor isn't displaying the data after the EOF mark. To eliminate the EOF mark:

1) Get the file length with the filelength() function:

```
FILE *file_pointer = fopen("file.nam","a");
long length = filelength(fileno(file_pointer));
```

2) Use the chsize() function to change the file length to the current length-1:

```
chsize(fileno(file_pointer), (length -1));
```

3) Then write your appended data to the file.

Q: I run my program, allocate some memory, and check the amount of memory available with coreleft(). Then I free some memory and call coreleft() again. It reports the same number. Why?

A: Coreleft does NOT return the amount of memory available. It returns the total memory available above the highest block allocated. It does NOT return any amount of memory available in "holes" below the highest allocated block. The code for a function that returns the amount of total free memory is provided in the document TI1723. This document is available on Borland's Download BBS, (408)431-5096, as document # 1723, and on Borland Techfax, (800) 822-4269, as TI1723.ZIP.

----------END OF FILE HELPME.WRI----------