63.3 Module List for the Compiler Suite

B

This module is the introspective ("reflective" in Java terms) module, which allows a Perl program to inspect its innards. The back end modules all use this module to gain access to the compiled parse tree. You, the user of a back end module, will not need to interact with B.

O

This module is the front-end to the compiler's back ends. Normally called something like this:

```
  $ perl -MO=Deparse myperlprogram
```

This is like saying `use O 'Deparse'` in your Perl program.

B::Asmdata

This module is used by the B::Assembler module, which is in turn used by the B::Bytecode module, which stores a parse-tree as bytecode for later loading. It's not a back end itself, but rather a component of a back end.

B::Assembler

This module turns a parse-tree into data suitable for storing and later decoding back into a parse-tree. It's not a back end itself, but rather a component of a back end. It's used by the *assemble* program that produces bytecode.

B::Bblock

This module is used by the B::CC back end. It walks "basic blocks". A basic block is a series of operations which is known to execute from start to finish, with no possibility of branching or halting.

B::Bytecode

This module is a back end that generates bytecode from a program's parse tree. This bytecode is written to a file, from where it can later be reconstructed back into a parse tree. The goal is to do the expensive program compilation once, save the interpreter's state into a file, and then restore the state from the file when the program is to be executed. See §63.2.5 for details about usage.

B::C

This module writes out C code corresponding to the parse tree and other interpreter internal structures. You compile the corresponding C file, and get an executable file that will restore the internal structures and the Perl interpreter will begin running the program. See §63.2.4 for details about usage.

B::CC

This module writes out C code corresponding to your program's operations. Unlike the B::C module, which merely stores the interpreter and its state in a C program, the B::CC module makes a C program that does not involve the interpreter. As a consequence, programs translated into C by B::CC can execute faster than normal interpreted programs. See §63.2.6 for details about usage.

B::Concise

This module prints a concise (but complete) version of the Perl parse tree. Its output is more customizable than the one of B::Terse or B::Debug (and it can emulate them). This module useful for people who are writing their own back end, or who are learning about the Perl internals. It's not useful to the average programmer.

B::Debug

This module dumps the Perl parse tree in verbose detail to STDOUT. It's useful for people who are writing their own back end, or who are learning about the Perl internals. It's not useful to the average programmer.

B::Deparse

This module produces Perl source code from the compiled parse tree. It is useful in debugging and deconstructing other people's code, also as a pretty-printer for your own source. See §63.2.2 for details about usage.