Obfuscated Tiny C Compiler

What is it ?

The Obfuscated Tiny C Compiler (OTCC) is a very small C compiler I wrote in order to win the International Obfuscated C Code Contest (IOCCC) in 2002.

My goal was to write the smallest C compiler which is able to compile itself. I choose a subset of C which was general enough to write a small C compiler. Then I extended the C subset until I reached the maximum size authorized by the contest: 2048 bytes of C source excluding the ';', '{', '}' and space characters.

I choose to generate i386 code. The original OTCC code could only run on i386 Linux because it relied on endianness and unaligned access. It generated the program in memory and launched it directly. External symbols were resolved with `dlsym()`.

In order to have a portable version of OTCC, I made a variant called OTCCELF. It is only a little larger than OTCC, but it generates directly a *dynamically linked i386 ELF executable* from a C source without relying on any binutils tools! OTCCELF was tested succesfully on i386 Linux and on Sparc Solaris.

NOTE: My other project TinyCC which is a fully featured ISOC99 C compiler was written by starting from the source code of OTCC !

Download

- Original OTCC version (runs *only* on i386 Linux): otcc.c (link it with `-ldl`).
- OTCC with i386 ELF output (should be portable): otccelf.c.
- Example of C program that can be compiled: otccex.c.
- **[New]** The non-obfuscated versions are finally available: otccn.c and otccelfn.c. These non-obfuscated versions do not self compile. They are provided for documentation purpose.

Compilation:

```
gcc -O2 otcc.c -o otcc -ldl
gcc -O2 otccelf.c -o otccelf
```

Self-compilation:

```
./otccelf otccelf.c otccelf1
```

As a test, here are the executables generated by OTCCELF: otcc1, otccelf1, otccex1.

C Subset Definition

Read joint example otccex.c to have an example of C program.

- Expressions:
  - binary operators, by decreasing priority order: '*' '/' '%', '+' '-', '>>' '<<', '<' '<=' '>' '>=', '==' '!=', '&', '^', '|', '=', '&&', '||'.
  - '&&' and '||' have the same semantics as C : left to right evaluation and early exit.
  - Parenthesis are supported.
  - Unary operators: '&', '*' (pointer indirection), '-' (negation), '+', '!', '~', post fixed '++' and

'--'.
    - Pointer indirection ('*') only works with explicit cast to 'char *', 'int *' or 'int (*)()' (function pointer).
    - '++', '--', and unary '&' can only be used with variable lvalue (left value).
    - '=' can only be used with variable or '*' (pointer indirection) lvalue.
    - Function calls are supported with standard i386 calling convention. Function pointers are supported with explicit cast. Functions can be used before being declared.
- Types: only signed integer ('int') variables and functions can be declared. Variables cannot be initialized in declarations. Only old K&R function declarations are parsed (implicit integer return value and no types on arguments).
- Any function or variable from the libc can be used because OTCC uses the libc dynamic linker to resolve undefined symbols.
- Instructions: blocks ('{' '}') are supported as in C. 'if' and 'else' can be used for tests. The 'while' and 'for' C constructs are supported for loops. 'break' can be used to exit loops. 'return' is used for the return value of a function.
- Identifiers are parsed the same way as C. Local variables are handled, but there is no local name space (not a problem if different names are used for local and global variables).
- Numbers can be entered in decimal, hexadecimal ('0x' or '0X' prefix), or octal ('0' prefix).
- '#define' is supported without function like arguments. No macro recursion is tolerated. Other preprocessor directives are ignored.
- C Strings and C character constants are supported. Only '\n', '\"', '\'' and '\\' escapes are recognized.
- C Comments can be used (but no C++ comments).
- No error is displayed if an incorrect program is given.
- Memory: the code, data, and symbol sizes are limited to 100KB (it can be changed in the source code).

OTCC Invocation

You can use OTCC by typing:

```
otcc prog.c [args]...
```

or by giving the C source to its standard input. 'args' are given to the 'main' function of prog.c (argv[0] is prog.c).

Examples:

- Sample compilation and execution:

  ```
  otcc otccex.c 10
  ```

- Self compilation:

  ```
  otcc otcc.c otccex.c 10
  ```

- Self compilation iterated...

  ```
  otcc otcc.c otcc.c otccex.c 10
  ```

An alternate syntax is to use it as a script interpreter: you can put

```
#!/usr/local/bin/otcc
```

at the beginning of your C source if you installed otcc at this place.

OTCCELF Invocation

You can use OTCCELF by typing:

```
otccelf prog.c prog
chmod 755 prog
```

'prog' is the name of the ELF file you want to generate.

Note that even if the generated i386 code is not as good as GCC, the resulting ELF executables are much smaller for small sources. Try this program:

```
#include <stdio.h>

main()
{
    printf("Hello World\n");
    return 0;
}
```

Results:

Compiler	Executable size (in bytes)
OTCCELF	*424*
GCC (stripped)	*2448*

Links

- TinyCC, a tiny but complete C compiler.
- Tiny ELF programs from Brian Raiter.
- Linux assembly projects.

License

The obfuscated OTCC and OTCCELF are public domain. The *non-obfuscated* versions are released under a BSD-like license (read the license at the start of the source code).

Fabrice Bellard - http://bellard.org/ - http://www.tinycc.org/