The Glow Object File Format Specification

21 May 2015

Nicolas Winkler

The Glow Object File Format

A Glow object file is a file containing a piece of Glow bytecode, which can either be run directly by a Glow virtual machine, or compiled using a Glow compiler tool. It can be viewed as the equivalent to *.class files for Java classes or *.obj/*.o files for C/C++ objects.

The Glow object file format uses little-endian encoding for all integers larger than one byte. The hexadecimal 32-bit constant `0xABCDEF01` would therefore be stored as the byte sequence {`0x01, 0xEF, 0xCD, 0xAB`}, which is perfectly counter-intuitive. But it's how the popular CPU architectures nowadays work.

Glow object files constist of

- header
- symbol table
- constant table
- bytecode

Header

The header is a fixed-size data structure at the beginning of every Glow object file.

Note that the data fields in the following structures are stored in the file in exactly the here defined order. No padding is used.

```
structure header {
    4 bytes: magic number;
    8 bytes: version;
    4 bytes: symbol table offset;
    4 bytes: symbol table length;
    4 bytes: constant table offset;
    4 bytes: constant table length;
    4 bytes: bytecode offset;
    4 bytes: bytecode length;
}
```

magic number These four bytes identify the Glow object file format. Their value is the ASCII representation of "GLOB" which corresponds to the hexadecimal value `0x424F4C47`.

version This entry specifies the version of the object file. The two most significant bytes (note: these are the last and the second last of the 8 bytes, say thanks to little-endian) stand for the main version, where the two following bytes can be increased on little changes. The remaining 4 bytes may be used in the future, or can be used for very very small changes if needed.

symbol table offset This value contains an offset pointing from the beginning of the file to the first entry of the symbol table, which for one thing holds references to the methods and classes in the current object file and for another thing also contains symbol names pointing to extern Glow object files.

symbol table length Specifies the number of entries in the symbol table. Since all entries in this table are the same size, the whole size of this table can be calculated from this value.

constant table offset This value contains an offset pointing from the beginning of the file to the first entry of the constant table, a table, which stores constants, which are too big to store directly in the bytecode (e.g strings).

constant table length Specifies the number of entries in the constant pool table. Note that the entries of this table may vary in size, so the absolute size of the table can't yet be decided.

bytecode offset Defines the offset from the beginning of the file to the start of the bytecode sequence.

bytecode length This value contains the number of bytes the bytecode part of the file takes up.

Symbol Table

This section constists of a table with entries for each class used and for each function called in the bytecode.

Each entry is of the following format:

```
structure symbol_table_entry {
    1 byte:  type;
    4 bytes: argument1;
    4 bytes: argument2;
}
```

type is always one of the following values:

`symbol_table_entry::type`	Value
`GLOB_CLASS_REFERENCE`	0x01
`GLOW_METHOD_REFERENCE`	0x02
`GLOB_CLASS_DEFINITION`	0x03
`GLOB_METHOD_DEFINITION`	0x04

argument1 The argument is a reference to the constant table entry containing the class name.

argument2 The argument is a reference to the constant table entry containing the method name (if needed).

Symbol Types

Depending on the value in `symbol_table_entry::type`, the meaning of the symbol entry differs.

An entry of type `GLOB_CLASS_REFERENCE` represents a class type. In this case, `symbol_table_entry::argument1` is an index pointing to an entry in the constant table. The value `symbol_table_entry::argument2` is ignored.

Several bytecode instructions will contain a link to an entry of this table:

- `GLOW_ALLOCATE_OBJECT = 0xB0` is followed by an index pointing to an entry in this table. This entry must therefore contain information about the class of which an instance should be allocated.
- `GLOW_CALL_`... like instructions are also followed by an index pointing to a function entry in this table.

Constant Table

This table stores big constant values like strings or class/method names.

Entries are of the following format:

```
structure constant_table_entry {
    1 byte:       type;
    4 bytes:      length;
    length bytes: data;
}
```

type identifies the type of the constant. This can be one of the following values:

`constant_table_entry::type`	Value
`GLOB_CONSTANT_STRING`	0x01

length specifies the size (in bytes) of the data part of the entry.

data is the actual data in the entry.

The entries follow each other directly i.e. there is no space between two entries. The offset to the next entry can therefore be computed using the formula: $5 +$ `length`.

Bytecode

This is the simplest part of the file. It just contains the plain bytecode content, ready for interpretation by an interpreter or further compilation to architecture-specific machine code.

For more information about the Glow bytecode, please read the Glow Byte-code Specification.