Boboscript

Boris Martin

February 15, 2017

Contents

1 Introduction

The main goal of this document is to provide a formal, exact description of the programming language I'm working on, known as *Boboscript*. It will give indications about how to scan, parse, run or compile it, in a way that'll make the implementation almost obvious.

1.1 Purpose and main characteristics

Paradigm and data model Boboscript is meant to be a procedural, *C-like* language, which heavily encourage decoupling of *code* and *data*. Code is mainly represented through functions and data via POD structures. It supports primitive object-oriented programming, with garbage-collected objects that are the only way to have recursive data structures. No manual memory is allowed, excepted in low-level code (native C calls) : data is either stack-allocated or garbage-collected object.

Type system Boboscript is *statically* and *strongly typed* language, and favors *immutable by default* data. It is designed for compilation to C, and thus, use the C memory model.

Limitations Multi-thread support is not required, but could probably used through C native calls. Operation atomicity is not guaranteed in any case.

Modularity Every Boboscript file describes exactly one module, whose name must be declared in the beginning, with uppercase letters. It can be compiled to a binary format, with extension ".bobj". A full program consists of linked objects file, including one defining a module MAIN, which must contain the main() function.[1]
A module may have local functions declared with the *static* qualifier.

[1] In future versions, it could become possible to compile to a C library, with auto-generated headers.

2 Structure of a program