ELFIO

Tutorial and User Manual

Table of Contents

[2 Introduction 2](#_Toc340883638)

[3 Getting Started With ELFIO 2](#_Toc340883639)

[3.1 ELF File Reader 2](#_Toc340883640)

# Introduction

ELFIO is a C++ library for reading and generating files in ELF binary format. This library is independent and does not require any other product. It is also cross-platform - the library uses standard ANSI C++ constructions and runs on wide variety of architectures.

While the library's implementation does make your work much easier: basic knowledge of the ELF binary format is required. Information about ELF format can be found widely on the web.

# Getting Started With ELFIO

## ELF File Reader

The ELFIO library is a header only library. No preparatory compilation steps are required. To make your application be aware about the ELFIO classes and types declarations, just include <elfio.hpp> header file. All ELFIO library declarations reside in ELFIO namespace. So, this tutorial code starts with the code:

#include <iostream>

#include <elfio.hpp> 

using namespace ELFIO 

int main( int argc, char\*\* argv )

{

if ( argc != 2 ) {

std::cout << "Usage: tutorial <elf\_file>" << std::endl;

return 1;

}

 - Include elfio.hpp header file

 - The ELFIO namespace usage

This chapter will explain how to work with the reader portion of the ELFIO library. The first step would be creation of the elfio class instance. The elfio constructor does not receive any parameters. After creation of a class object, we initialize the instance by invoking load function passing ELF file name as a parameter.

// Create elfio reader

elfio reader; 

// Load ELF data

if ( !reader.load( argv[1] ) ) { 

std::cout << "Can't find or process ELF file " << argv[1] << std::endl;

return 2;

}

 - Create elfio class instance

- Initialize the instance by loading ELF file. The function load returns ‘true’ if the ELF file was found and processed successfully. It returns ‘false’ otherwise



ELF header properties are available now. This makes it possible to request file parameters such as encoding, machine type, entry point, etc. To get the class and the encoding of the file use:

// Print ELF file properties

std::cout << "ELF file class : ";

if ( reader.get\_class() == ELFCLASS32 )

std::cout << "ELF32" << std::endl;

else

std::cout << "ELF64" << std::endl;

std::cout << "ELF file encoding : ";

if ( reader.get\_encoding() == ELFDATA2LSB )

std::cout << "Little endian" << std::endl;

else

std::cout << "Big endian" << std::endl;