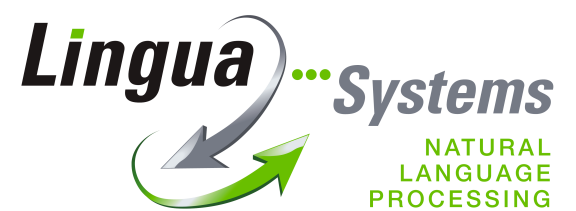


Software Specification

for

lid

Covers version 3.1.0



1 Introduction

This document provides the software specification for `lid`, version 3.1.0.

`lid` is a shared C/C++ library that identifies language and character encoding of textual input.

The specification serves as a general introduction to the software on the one hand and as a detailed description on the library's scope of service on the other hand.

Hint on conventions used:

At several points it is necessary to make a distinction between strings that may not contain embedded NUL characters and those which may, due to their special character encoding, potentially contain NUL characters. The former are called "Strings" while the latter are called "Byte Strings".

2 Overview on the Software

`lid` is a shared library that identifies language and character encoding of textual input. The input has to be plain text and can be passed to `lid` either as a file or a string. The results are returned as a data structure.

2.1 User's Requirements

Every user with basic knowledge of C/C++ programming and library usage is able to use `lid` right away.

`lid`'s field of application is generally every field that benefits from knowing the language and/or character encoding of its relevant documents.

2.2 Operating Environment

`lid` is a library suited for Unix-like operating systems, but is available with the same functionality for different versions of Windows, too. Its standard version is currently available for the following operating systems:

- Debian GNU/Linux (x86): *Etch, Lenny*
- Ubuntu GNU/Linux (x86/IA-32): *LTS (10.04)*
- Solaris (Sparc): *Solaris 10*
- FreeBSD (x86): *6, 7, 8*
- Microsoft Windows (x86/IA-32): *XP, Server 2003, Vista, Server 2008, 7*
- Microsoft Windows (x86_64/IA-64): *7, Server 2008 R2*

The software may very well work on other versions and/or distributions without modification although `lid` is only supported in the environments specified above.

Versions for other distributions and/or operating systems may be made available upon request as well.

2.3 Dependencies

`lid` does not require any additional software (libraries) to be installed except for the standard C and thread library shipped with the particular system.

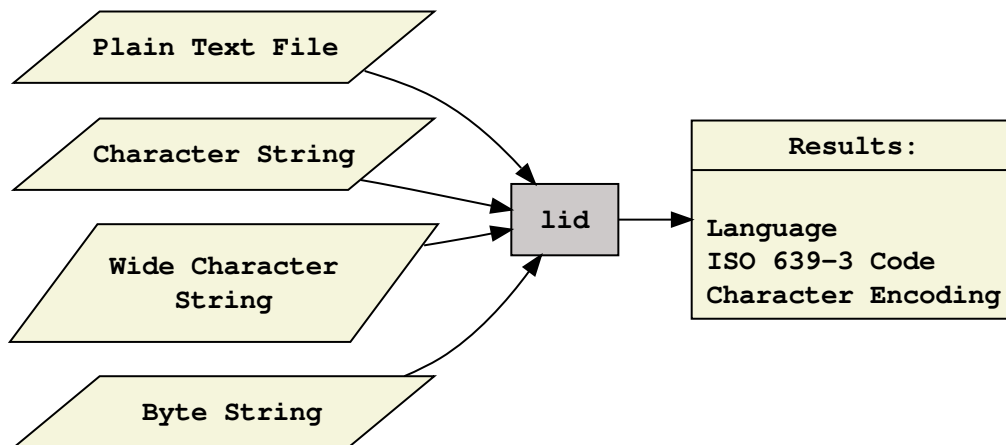
2.4 Use of Resources

`lid` computes the results very efficiently and requires in average – independent from the input's size – not more than 285 KiB of memory.

For each thread using the provided functionality, `lid` allocates two variables on Thread-Local Storage (TLS). These are used for error handling purposes.

3 Scope of Service

The `lid` library supports 26 languages and 35 character encodings. Additionally 10 languages are identified in transliterated form as well. The input may be given in a variety of different formats. The result covers the identified language, its ISO 639-3 code and the identified character encoding.



3.1 Input

3.1.1 Supported Input Formats

Only input in plain text format can be processed. The input can be passed to `lid` as:

→ Files

→ Strings

- Character String (`char *`)
- Wide Character String (`wchar_t *`)
- Byte String (`char *`)

The input's length is only limited by the respective data types.

3.1.2 Supported Languages and Character Encodings

Currently, 26 languages can be identified. The 35 supported encodings cover commonly used encodings as well as deprecated ones.

The byte-order of any UTF-16 and UTF-32 input is determined additionally. In detail, these encodings are determined as either "UTF-16BE", "UTF-16LE", "UTF-32BE" or "UTF-32LE".

Language	ISO 639-3 Code	Character Encoding
Bulgarian	bul	UTF-32, UTF-16, UTF-8, ISO-8859-5, Windows-1251, MacCyrillic, CP 855, CP 866, KOI8-R
Czech	ces	UTF-32, UTF-16, UTF-8, ISO-8859-2, Windows-1250, MacCentralEurope, CP 852
Danish	dan	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Dutch	nld	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
English	eng	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Estonian	est	UTF-32, UTF-16, UTF-8, ISO-8859-4, Windows-1257, MacCentralEurope, CP 775, ASCII
Finnish	fin	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
French	fra	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
German	deu	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Greek	ell	UTF-32, UTF-16, UTF-8, ISO-8859-7, Windows-1253, MacGreek, CP 737
Hungarian	hun	UTF-32, UTF-16, UTF-8, ISO-8859-2, Windows-1250, MacCentralEurope, CP 852
Irish (Gaelic)	gle	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Italian	ita	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Lithuanian	lit	UTF-32, UTF-16, UTF-8, ISO-8859-4, Windows-1257, MacCentralEurope, CP 775, ASCII
Latvian	lav	UTF-32, UTF-16, UTF-8, ISO-8859-4, Windows-1257, MacCentralEurope, CP 775, ASCII
Maltese	mlt	UTF-32, UTF-16, UTF-8, ISO-8859-3
Mandarin (Chinese)	cmn	UTF-32, UTF-16, UTF-8, Big5, GB2312
Polish	pol	UTF-32, UTF-16, UTF-8, ISO-8859-2, Windows-1250, MacCentralEurope, CP 852
Portuguese	por	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Romanian	ron	UTF-32, UTF-16, UTF-8, ISO-8859-2, Windows-1250, MacRomanian, CP 852
Russian	rus	UTF-32, UTF-16, UTF-8, ISO-8859-5, Windows-1251, MacCyrillic, CP 855, CP 866, KOI8-R
Swedish	swe	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Slovak	slk	UTF-32, UTF-16, UTF-8, ISO-8859-2, Windows-1250, MacCentralEurope, CP 852

Language	ISO 639-3 Code	Character Encoding
Slovenian	slv	UTF-32, UTF-16, UTF-8, ISO-8859-2, ISO-8859-16, Windows-1250, MacCentralEurope, CP 852, ASCII
Spanish	spa	UTF-32, UTF-16, UTF-8, ISO-8859-1, ISO-8859-15, Windows-1252, MacRoman, CP 850, ASCII
Ukrainian	ukr	UTF-32, UTF-16, UTF-8, Windows-1251, MacUkrainian, KOI8-U

In addition to the languages, 10 languages can also be identified in transliterated form. The supported 12 transliterations are given here. "Common" denotes transliterations that are not based on a standard but are commonly used nevertheless.

Language	Transliteration	Character Encodings
Bulgarian	ISO 9	UTF-32, UTF-16, UTF-8, ASCII
	DIN 1460	UTF-32, UTF-16, UTF-8, ASCII, Windows-1250
	Streamlined System	UTF-32, UTF-16, UTF-8, ASCII, Windows-1250
Czech	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
German	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Greek	ISO 843	UTF-32, UTF-16, UTF-8, ASCII
	DIN 31634	UTF-32, UTF-16, UTF-8, ASCII
	Greeklsh	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Polish	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Romanian	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Russian	ISO 9	UTF-32, UTF-16, UTF-8
	DIN 1460	UTF-32, UTF-16, UTF-8
Slovak	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Slovenian	common	UTF-32, UTF-16, UTF-8, ASCII, ISO-8859-1
Ukrainian	ISO 9	UTF-32, UTF-16, UTF-8
	DIN 1460	UTF-32, UTF-16, UTF-8

3.2 Return Value

3.2.1 Results

The following information is made available as a result:

- The identified language
- The ISO 639-3 Code
- The identified character encoding

3.2.2 Format

The results are returned as a data structure. The structure is formally defined as follows:

```
typedef struct lid {
    char *language;      /* i.e. "Bulgarian" */
    char *encoding;     /* i.e. "Windows-1252" */
    char *isocode;     /* i.e. "bul" */
} lid_t;
```

3.3 User Interface

Functions are provided to access the library's functionality. There is one function each for identifying language and character encoding depending on the input's format.

Additionally a function for error handling and freeing memory is provided.

All the functions with their parameters and return values are explained in detail in the User Manual to this software.

3.4 Error Handling

`lid` provides thread-safe error handling facilities. Each thread has its own error indicator that stores numeric error codes. Given these error codes, a natural language error message may be generated by a library function. For convenience, a named constant is defined for each error code, so that error codes can be referenced by name.

3.5 Security Considerations

`lid` was designed and implemented with security in mind and has no single known safety defect. Besides that, the application does not create any temporary files or evaluate any environment variables.

Memory allocated internally by `lid` is always freed. This includes all known error paths.

3.6 Restrictions

- At the present state of the art, an identification of language and/or character encoding cannot be guaranteed to be accurate in any case without restrictions. There may be documents that lead to wrong identification results.
- An input can only be processed accurately if it is available in plain text or has been preprocessed to this format before. Any such preprocessing is not part of `lid`.
- Only supported languages and character encodings can be identified.
- Unsupported languages and character encodings will be identified as the most similar language and/or character encoding.
- Multilingual documents cannot be processed unless they contain no more than a few passages in another language.

- Any input should contain at least 25 characters of textual data that form different words.
- The software itself is provided in English only.
- The manual is only provided as a PDF document, in both an English and German version.
- There is no general guarantee for downward compatibility to previous versions.

4 Deliveries

A complete installation of the software consists of the compiled library, all associated header files, man pages and the user manual in an English and German version. The installation requires about 1.5 MiB of disk space.

5 References

- Text format: MIME-Type: text/plain, RFC 2046 [3]
- Language Codes : ISO 639-3:2007 *Codes for the representation of names of languages*
- User Manual for [lid](#) version 3.1.0