

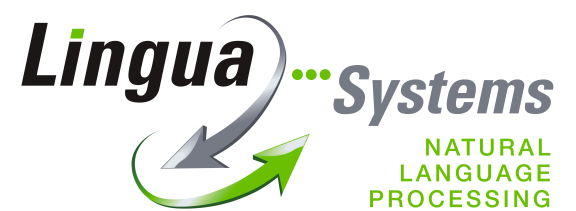
# User Manual

for

lidc

An application to identify language and character encoding

Covers version 1.3.0



lidc User Manual published February 1, 2011.

Copyright © 2009 - 2011, Lingua-Systems Software GmbH.

Lingua-Systems Software GmbH, Wiesenstraße 34, 44653 Herne, Germany, *info@lingua-systems.com*

All rights reserved, especially changing or publishing parts of this manual needs prior written permission of the copyright owner.

The rights to reproduce and publish unchanged copies in any form, to translate or to present the manual are granted.

Mentioned hard- and software as well as companies may be trademarks of their respective owners. Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark. A missing annotation of the trademark may not lead to the assumption, that no trademark is claimed and may thus be used freely.

Great effort has been made in writing this manual. However, faults cannot be excluded in general. For any loss or damages caused or alleged to be caused directly or indirectly by errors or omissions in this manual, the authors and the publisher assume no responsibility and cannot be held liable. Neither can the authors or the publisher be held liable for the content or changes of content concerning the linked websites. The links have been carefully chosen and proved at the preparation of the manual.

If you have problems using the links or get aware of any faults, feel free to give a brief hint on it via *support@lingua-systems.com*.

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Supported Input Types</b>	<b>5</b>
<b>3</b>	<b>Supported Languages and Character Encodings</b>	<b>6</b>
<b>4</b>	<b>Installation</b>	<b>8</b>
4.1	Requirements . . . . .	8
4.2	What will be installed . . . . .	8
4.3	Installing the Software . . . . .	8
4.3.1	Installing the Software on Debian GNU/Linux . . . . .	8
4.3.2	Installing the Software on Solaris . . . . .	8
4.3.3	Installing the Software on FreeBSD . . . . .	9
4.3.4	Installing the Software on Mac OS X . . . . .	9
4.4	Deinstalling the Software . . . . .	10
4.4.1	Deinstalling the Software on Debian GNU/Linux . . . . .	10
4.4.2	Deinstalling the Software on Solaris . . . . .	10
4.4.3	Deinstalling the Software on FreeBSD . . . . .	10
4.4.4	Deinstalling the Software on Mac OS X . . . . .	10
<b>5</b>	<b>Using lidc</b>	<b>11</b>
5.1	Reference . . . . .	12
5.2	Options . . . . .	12
5.3	Customizing the Output Format String . . . . .	13
5.3.1	Examples . . . . .	14
5.4	Interpreting Character Encoding Results . . . . .	15
5.4.1	Identified Character Encoding . . . . .	15
5.4.2	Declared Character Encoding . . . . .	15
5.5	Avoiding Errors . . . . .	16
<b>A</b>	<b>References</b>	<b>17</b>

## About this manual

---

This manual addresses users with experience in using command line applications.

The manual provides a short introduction to the application, its supported languages and character encodings as well as the supported input types. Instructions are given how to install the `lidc` software package. Afterwards the complete usage is introduced covering all options and the possibility to customize the output. Finally, a short primer on avoiding errors is given.

For a quickstart have a look at the reference (chapter 5.1 on page 12).

Administrators who want to install the software get all necessary information in chapter 4, page 8.

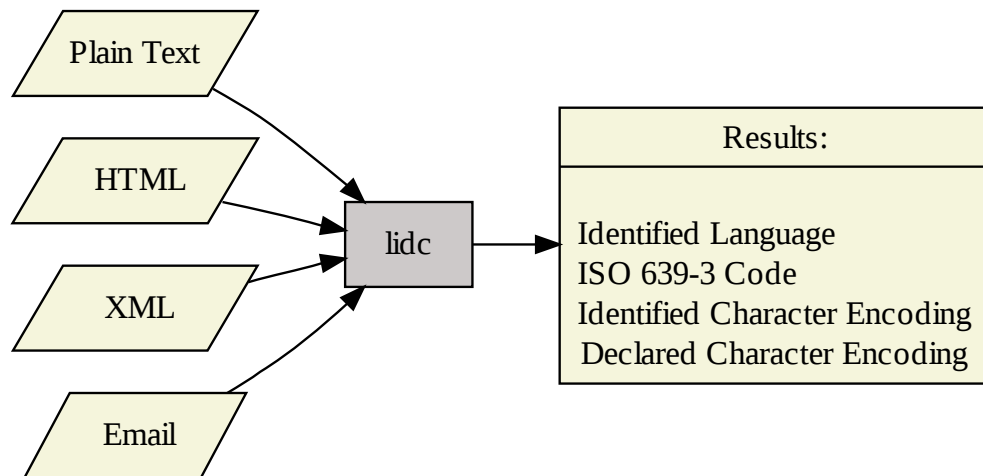
# 1 Introduction

---

`lidc` is a command line application that identifies the language and character encoding of textual input.

The input may be plain text, *HTML* or *XML* documents and emails. `lidc` reads its input either from file or from standard input (`stdin`). Therefore it may be used as part of a command chain (pipe).

The results returned cover the identified language, its ISO 639-3 code, the identified character encoding and - if available - the declared character encoding. The output can be customized at will with a format string.



`lidc` is based on the `lid` library which supports 28 languages and 35 encodings. Additionally 10 languages are supported in transliterated forms.

The application computes the results fast and reliable and has no software dependencies except for the standard C and thread library. Therefore `lidc` can be easily integrated on all supported platforms and works efficiently even on dated hardware.

## 2 Supported Input Types

---

The application can process plain text, *HTML*, *XML* documents and emails. The following listing provides an overview on the different supported input types, especially concerning email.

1. Plain Text
2. HTML: HTML (4.0, ...), X-HTML
3. XML
4. Email (RFC 822)
5. Email: text/plain, text/html, multipart/mixed, multipart/alternative, multipart/digest, message/rfc822, multipart/parallel (RFC 2045-2049: MIME)
6. Email: multipart/related (RFC 2387)
7. Email: multipart/report (RFC 3462)
8. Email: multipart/signed (RFC 1847)

### 3 Supported Languages and Character Encodings

Currently, 28 languages can be identified. The supported encodings cover commonly used encodings as well as traditional ones.



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

These encodings are not supported for the input type *email*.

Language	ISO 639-3 Code	Character Encoding
Bokmål (Norwegian)	nob	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
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, ISO-8859-15, 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, ISO-8859-15, Windows-1252, MacRoman, CP 850, ASCII
French	fra	UTF-32, UTF-16, UTF-8, ISO-8859-1, ISO-8859-15, Windows-1252, MacRoman, CP 850, ASCII
German	deu	UTF-32, UTF-16, UTF-8, ISO-8859-1, ISO-8859-15, 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, ISO-8859-16, 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, ISO-8859-16, 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

Language	ISO 639-3 Code	Character Encoding
Nynorsk (Norwegian)	nno	UTF-32, UTF-16, UTF-8, ISO-8859-1, Windows-1252, MacRoman, CP 850, ASCII
Polish	pol	UTF-32, UTF-16, UTF-8, ISO-8859-2, ISO-8859-16, Windows-1250, MacCentralEurope, CP 852
Portuguese	por	UTF-32, UTF-16, UTF-8, ISO-8859-1, ISO-8859-15, 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
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 be recognized in transliterated forms. The supported transliterations cover official norms and commonly used transliterations as for example found in emails.

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
	Greeklisk	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

Only the supported languages and encodings can be identified. If the input is in an unsupported language or encoding, no error is indicated. `lidc` will determine the most similar language and encoding and return this as a result.

## 4 Installation

---

### 4.1 Requirements

`lidc` merely uses the system's standard C and thread library and has no other dependencies.

Demands on the underlying hardware are minimal. Depending on the type and size of the input, the application usually requires between 50 and 200 KiB of RAM and computes its results very fast.

### 4.2 What will be installed

The `lidc` package contains the application itself, its man page and this manual in an English and German version.

The following shows an installation tree on a Debian GNU/Linux system:

```
/opt/ls/  
|-- bin  
|   '-- lidc  
'-- share  
    |-- doc  
    |   '-- lidc  
    |       |-- copyright  
    |       |-- changelog.gz  
    |       |-- manual_deu.pdf  
    |       '-- manual_eng.pdf  
    '-- man  
        '-- man1  
            '-- lidc.1.gz
```

### 4.3 Installing the Software

The software package is provided in the native format for all supported operating systems and distributions. An installation can thus be achieved by the system's administrator with the commonly used standard utilities.

#### 4.3.1 Installing the Software on Debian GNU/Linux

The package can as usual be installed with administrative privileges using `dpkg(1)`:

```
debian-box% dpkg --install lidc_1.3.0-1_i386.deb
```

The package complies to the *Filesystem Hierarchy Standard Version 2.3* and installs all files beneath `/opt/ls`.

#### 4.3.2 Installing the Software on Solaris

An installation of the `lidc` package on Solaris can be done by using `pkgadd(1)` with administrative privileges as follows:

```
solaris-box% gzip -d LSlidc-1.3.0-sol10-sparc-opt.pkg.gz  
solaris-box% pkgadd -d LSlidc-1.3.0-sol10-sparc-opt.pkg
```

```
The following packages are available:
 1  LSlidc      lidc (/opt)
      (sparc) 1.3.0

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1

Processing package instance <LSlidc> from \
  <LSlidc-1.3.0-sol10-sparc-opt.pkg>

lidc (/opt)(sparc) 1.3.0
Lingua-Systems Software GmbH

...

Installation of <LSlidc> was successful.
```

The software will be installed beneath `/opt/ls/`.

### 4.3.3 Installing the Software on FreeBSD

An installation of the `lidc` package on FreeBSD can be done by using `pkg_add(1)` with administrative privileges as follows:

```
freebsd-box% pkg_add lidc-1.3.0_1.tbz
```

The software will be installed beneath `/usr/local/`.

### 4.3.4 Installing the Software on Mac OS X

`lidc` is provided as a Disk Image (`.dmg`) that can as usual be mounted by double-clicking on its icon. Once mounted, the software is immediately usable and can be installed by copying the image's content to a local directory, for example `/opt/local/`.

## 4.4 Deinstalling the Software

The software package is provided in the native format for all supported operating systems and distributions. A deinstallation can thus be achieved by the system's administrator using the commonly used standard utilities.

### 4.4.1 Deinstalling the Software on Debian GNU/Linux

The `lidc` package can as usual be deinstalled with administrative privileges using `dpkg(1)`:

```
debian-box% dpkg --remove lidc
```

### 4.4.2 Deinstalling the Software on Solaris

The `lidc` package can be removed using the `pkgrm(1)` command. Administrative privileges are required.

```
solaris-box% pkgrm LSlidc

The following package is currently installed:
  LSlidc  lidc (/opt)
          (sparc) 1.3.0

Do you want to remove this package? [y,n,?,q] y

...

Removal of <LSlidc> was successful.
```

### 4.4.3 Deinstalling the Software on FreeBSD

The `lidc` package can as usual be removed using `pkg_delete(1)`. Administrative privileges are required.

```
freebsd-box% pkg_delete lidc-1.3.0_1
```

### 4.4.4 Deinstalling the Software on Mac OS X

If the Disk Image's content has been copied to a local directory, for example `/opt/local/`, removing the installed files is sufficient to completely deinstall `lidc`.

## 5 Using `lidc`

---

The `lidc` application has to be invoked on the command line and takes an optional set of options that allows to suite it to your needs. For example, you can set the input file or customize the output by specifying a format string. For a detailed explanation of all available options have a look at the chapter "Options" (5.2, page 12).

`lidc` reads input either from file or standard input (`stdin`) and can thus be used within a pipe. The following example shows an invocation using a file as an input.

```
$ lidc -i website.html -t html -f "%l\n"
```

In the example above an *HTML* document is passed to `lidc` and the input type is set as appropriate. The format string is set to output only the identified language, followed by a newline. The result could be as follows:

```
Danish
```

`lidc` provides four distinct results: the identified language, its ISO 639-3 code, the identified character encoding and the declared character encoding. Due to the complete lack of markup and meta information, the declared encoding is not provided when processing plain text documents. For more information on the meaning and interpretation of character encoding results have a look at chapter 5.4.

Result	Description	Example
language	language name	"German"
isocode	ISO 639-3 language code	"deu"
identified encoding	name of the identified character encoding	"UTF-8"
declared encoding	name of the declared character encoding	"utf-8"

*Figure 1: Results*

The output may be customized by replacing the default output format string. All available flags are explained in "Customizing the Output Format String" (5.3, page 13).

In order to fulfill its purpose, `lidc` needs an input of at least 25 characters in length that form different words and thus provide some degree of variance. As a result, a character string that consists only of the word "house" is not a valid input, as well as a string that contains nothing but the repetition of the same word, although the minimum input length criteria may be met in the latter case.

Please note: When processing multipart *MIME* emails, only the first textual message body is evaluated.

## 5.1 Reference

Option	Parameter	Explanation
-i	input file's path	input document
-t	type [txt html xml email]	input type
-f	output format string	output format
	%l	identified language
	%i	ISO 639-3 language code
	%e	identified character encoding
	%d	declared character encoding
	%f	input file's path
	\n, \r, \t, \a	escape sequences
-v	-	<b>lidc</b> version
-h	-	help text

## 5.2 Options

### -i PATH

This option allows to set the path to an input file. If this option is not specified or set to "-", standard input (`stdin`) is used as an input.

### -t TYPE

This option allows to set the input file's type. According to the supported types (see chapter 2) the set of valid arguments is as follows:

- `txt`  
plain text documents without any markup
- `html`  
any HTML document (e.g. X-HTML, HTML 4.0)
- `xml`  
any XML document
- `email`  
emails conforming to RFC 822 or MIME 1.0

If **lidc** is invoked with a file argument and no type is set, **lidc** attempts to automatically determine the correct type by evaluating the file's extension. The commonly used file extensions (`.txt`, `.html`, `.xml`, `.eml`) are recognized as well as all *Maildir* extensions and additions as introduced by the *Dovecot* IMAP server.

If no input type is set and the input type cannot be determined, `txt` is used as a default.

### -f FMT\_STR

This option allows to customize the output as needed by setting a format string. The set of available flags is described in detail in the next chapter. The output is send to standard output (`stdout`).

If this option is omitted, the default format string will be used to output the results: "%l,%i,%e\n" (identified language, ISO 639-3 code, identified character encoding).

#### **-v**

Given this option, `lidc` prints information about its version and the underlying `lid` library to `stdout` and terminates.

#### **-h**

Given this option, `lidc` displays a short help text and terminates.

### **5.3 Customizing the Output Format String**

The output can be customized by specifying a format string. As a result, it is possible to suite the output to your needs easily. For example, generating results in a *CSV* or *XML* format can be achieved by setting the format string to an appropriate value.

Flags are provided for all results and will be replaced with their corresponding values in the output.

Please note: the exclusive usage of the input file's path flag (%f) is not allowed, because it does not represent any computed result.

#### **%l**

The %l flag is replaced by the identified language, i.e. "German", "French" or "Swedish".

#### **%i**

The %i flag is replaced by the ISO 639-3 code of the identified language, i.e. "deu", "fra" or "swe".

#### **%e**

The %e flag is replaced by the identified character encoding, i.e. "ISO-8859-1", "Windows-1252" or "UTF-8".

#### **%d**

The %d flag is replaced by the declared character encoding in lowercase letters, i.e. "iso-8859-1", "windows-1252" or "utf-8".

Please note: not all available input types support the declaration of a character encoding or just allow it as an optional addition. If no such declaration is provided within the input document, the flag expands to "none".

#### **%f**

The %f flag is replaced by the input file's path or, in case the input is read from standard input, by "stdin".

## Escape Sequences

Additionally, the following common escape sequences are supported to customize the output: "\n", "\r", "\t" and "\a".

### 5.3.1 Examples

An invocation of `lidc` using a simple format string may look like this:

```
$ lidc -i myfile.xml -f "%f: %l, %e\n"
myfile.xml: Bulgarian, UTF-8
```

The following example shows an *XML* output:

```
$ lidc -i german.eml -f \
"<email>\n\t<lang>%l</lang>\n\t<enc>%e</enc>\n</email>\n"
<email>
  <lang>German</lang>
  <enc>ISO-8859-1</enc>
</email>
```

The following more complex example utilizes a pipe to first convert a PDF file to plain text using `pdftotext(1)` (part of *xpdf*) and process the result with `lidc(1)` afterwards:

```
$ pdftotext manual_deu.pdf - | lidc -f "%l, %e, %d\n"
German, ISO-8859-1, none
```

## 5.4 Interpreting Character Encoding Results

### 5.4.1 Identified Character Encoding

The meaning of the identified character encoding (%e flag) depends on the input type and has to be interpreted accordingly.

Whenever a plain text, *HTML* or *XML* document is processed, the identified character encoding denotes the *input file's character encoding*.

If an email is processed, this interpretation may still be valid, primarily if the email is given in a 7 or 8 bit transfer-encoding. However, in case the email was encoded using *Quoted-Printable* or *Base64 (MIME)*, `lidc` has to decode the message body to obtain the original natural language text in order to identify its language and character encoding. Whenever an email is processed, the meaning of the identified character encoding should thus be interpreted as denoting the *character encoding of the decoded message body*, as it may differ from the file's encoding.

Type	identified encoding refers to...
txt	file
html	file
xml	file
email	first decoded message body

Figure 2: interpretation of the identified character encoding

### 5.4.2 Declared Character Encoding

A declared character encoding can only be determined for input types that support this kind of meta information. This is true for `html`, `xml` and `email (MIME)`. The declaration is, however, optional for all of these types.

Any declared character encoding will be converted to lowercase letters.

`lidc` may be used to determine defective character encoding declarations. This can be achieved by comparing the results of the identified (%e) and the declared character encoding (%d). However, not any reported difference between these values indicates a defective declaration - this is especially the case, if the declared character encoding is a superset of the identified one, as with UTF-8 and ASCII or ISO-8859-1 and ASCII.

## 5.5 Avoiding Errors

In case of an error that makes it impossible to process the input, the application terminates and an error message is given that describes the cause of the error. The most probable errors are named here and hints are given to avoid or interpret them.

In order to process an input reliably, it has to be of a minimum size. Clues for a sufficient length are  
⇒ the text has more than 25 characters and  
⇒ more than two words and  
⇒ is formed of different words.

Furthermore there has to be text at all. This need not be apparent on the first sight, for example if an email or *XML* document is processed.

Emails lack of input that can be processed if

⇒ it does not contain a text part or  
⇒ is encoded in UTF-16 or UTF-32 or  
⇒ no part is of one of the supported types (e.g. multipart/encrypted).

*XML* documents may consist of tags only that do not contain textual input.

If the text is too short or no text can be extracted at all, one of the following error messages will be triggered, depending on the circumstances:

```
lidc: Insufficient input length
lidc: No text extracted
```

If by accident a picture or an application is used as input, the following error is thrown:

```
lidc: Binary Data
```

When invoking *lidc*, every option can only be defined once. If an option is set more than once the application provides this message:

```
lidc: [Parameter] redefined
```

Only the supported input types (*txt*, *html*, *xml* and *email*) can be processed. If an unsupported input type is set, there will be an error as well:

```
lidc: Unsupported type
```

Further errors are mainly triggered if the input is defective in any way, for example if an email is encoded incorrectly or an UTF-16 or UTF-32 encoding is faulty. The respective error messages give hints to the cause of the error. In most cases these errors cannot be headed off.

If the input can be computed although there was an error, an appropriate warning is sent to `stderr`. Nevertheless the input is processed and a result is returned.

## A References

---

- ISO 639-3 Standard, <http://www.sil.org/iso639-3/>
- RFC 822, "Standard for the Format of ARPA Internet Text Messages"
- RFC 2045, 2046, 2047, 2049: "Multipurpose Internet Mail Extensions" (MIME)
- RFC 2387: "The MIME Multipart/Related Content-type"
- RFC 1847: "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted"
- RFC 3462: "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages"
- Maildir Standard, <http://cr.yp.to/proto/maildir.html>
- Dovecot IMAP Server, <http://www.dovecot.org/>
- Lingua-Systems' `lidc` product website, <http://www.lingua-systems.com/language-identifier/lidc-application/>

# Index

---

<b>C</b>	
character encoding .....	15
declared .....	15
identified .....	15
interpreting the results .....	15
supported .....	6
customizing the output .....	13
<b>D</b>	
defective character encoding declaration .....	15
deinstalling the software .....	10
Debian GNU/Linux .....	10
FreeBSD .....	10
Mac OS X .....	10
Solaris .....	10
dependencies .....	8
dpkg (Linux) .....	8, 10
<b>E</b>	
error messages .....	16
example .....	11, 14
format string .....	14
format string (XML) .....	14
invocation .....	11
PDF .....	14
<b>F</b>	
format string .....	13
<b>I</b>	
input types .....	5, 12, 15
email .....	5, 12, 15
html .....	5, 12, 15
txt .....	5, 12, 15
xml .....	5, 12, 15
installed files .....	8
installing the software .....	8
Debian GNU/Linux .....	8
FreeBSD .....	9
Mac OS X .....	9
Solaris .....	8
invocation .....	11
<b>L</b>	
languages .....	6
<b>M</b>	
minimum input length .....	11, 16
<b>O</b>	
options .....	12
-f FMT_STR .....	12
-h .....	13
-i PATH .....	12
-t TYPE .....	12
-v .....	13
output .....	13, 15
declared character encoding .....	11, 15
format string .....	13
identified character encoding .....	11, 15
identified language .....	11
ISO 639-3 code .....	11
<b>P</b>	
parameters .....	<i>see options</i>
pkg_add (FreeBSD) .....	9
pkg_delete (FreeBSD) .....	10
pkgadd (Solaris) .....	8
pkgrm (Solaris) .....	10
<b>R</b>	
requirements .....	8
result .....	<i>see output</i>
<b>T</b>	
transliteration .....	7
<b>W</b>	
warning .....	16