Speech Daemon — Easy access to speech synthesis

Mastering the Babylon of TTS'
for Speech Daemon 0.0

Tomáš Cerha `cerha@brailcom.org`
Hynek Hanke `hanke@volny.cz`
Milan Zamazal `pdm@brailcom.org`

This manual documents Speech Daemon, version 0.0.

Table of Contents

1 How to read this manual

`->` there should be a simple map of the manual and where to search for different concepts

2 Introduction

2.1 Why and how

Speech Daemon project comes to provide a device independent layer for speech synthesis. It should provide a simple interface for client applications (applications, that want to speak) as well as for device driver modules (for particular speech synthesis).

High quality speech synthesis has been available for a long time and now it's usable even by ordinary users on their home PC's. It comes sometimes as a necessity, sometimes as a good feature for programs to provide speech output. There is a wide field of possible uses from educational software, through specialized systems (hospitals, laboratories, telephony servers). For visually impaired users it is one of the two essential ways of getting the output from computer (the second one is Braille display). That's also where Speech Daemon comes from.

There are different speech synthesizers with different capabilities. Some of them are hardware, some of them are software. Some of them are Free Software and are are available on the Internet. However, none of them is pre-installed in one of the widely used GNU/Linux distributions. Programmers have really hard times when they want to make their program speak because they need to find some suitable synthesizer (long hours of experiments and so on...) and then make it work with their program. They often need to write output device drivers for these programs or hardware devices and are doing it again and again. You can imagine it all fails when an innocent user executes two programs with speech output at once — if they even start both (what I doubt), they will be shouting one over the other. This makes it very hard for programmers to implement speech support to their programs (for blind users or simply to make a better user interface) and it's one of the reasons we still don't fully exploit what speech synthesis technology offers.

In an ideal world, programmers could use similar commands for speech synthesis as they do for normal text output (printf, puts, ...). In an ideal world, there would be some speech_printf() that would take care of saying your message in the right time without interrupting others, without you being obligated to take care of how exactly the communication with speech synthesizer is implemented and without you having to worry about what synthesizer to use and if it's available. In an ideal world, there would be some speech synthesizer in each GNU/Linux distribution and some speech daemon taking care of all applications that want to speak, allowing user to configure speech parameters and providing simple interface (as speech_printf()) through some shared library for programmers. It will be a long way until we achieve this state of things, but with Speech Daemon, we are taking the first steps...

2.2 Current state

Today, the development of programs and new technologies connected with speech synthesis under GNU/Linux is centered around two main points: visually impaired people and pure development. Although some fields are beginning to use synthesis for different purposes, like telephony servers, these are still like drops of water in the ocean. Here is a short (definitely not exhaustive list) software synthesizers, hardware synthesizers and applications known to work under GNU/Linux.

1. Speech Synthesizers
    - Hardware synthesizers

      Hardware synthesizers are the devices, which may be connected to PC. Mostly they are external, connected via serial or parallel port. There are also some internal devices for ISA bus or USB. Application may send textual data to the port and the device converts it to spoken letters and words. Data may contain also several control sequences in the form of escaped characters as commands. The problem, we are facing, is that each of these devices uses its own communication protocol.
    - Software synthesizers
        - Festival

          Festival is a multi-lingual Free Software text to speech synthesizer with high quality speech databases available. One of it's problems is that some of the most important databases are not free. (e.g. the database for British English is non-free). Other problem is that Festival is intended rather as a platform for research and development than as an end-user product and therefore is big and not-so-easy to install. The problem we face as Speech Daemon Developers is that it's too slow to be really useful for most applications.
        - Flite

          Flite stands for Festival Lite and it is a light fully free English speech software synthesizer with good quality of sound, developed by the authors of festival as an end-user product. It's very fast, however, we currently don't know how to configure it (it seems it is not possible yet) and it seems that the developers have some problems with importing the voices from Festival. Speech Daemon currently uses Flite as it's primary output module for English.
        - Odmluva

          Odmluva is a simple (and very light) Czech speech synthesizer available under the terms of GNU GPL. We are working on it's support in Speech Daemon.
        - Epos

          Epos is Czech synthesis. It is an academic project and it already gives quite good results, but some parts are covered by a proprietary license.
        - Free TTS

          Free TTS is some JAVA-based text-to-speech system. We didn't checked it yet.
        - IBM ViaVoice

          ViaVoice is a multi-lingual software synthesizer available for GNU/Linux. The main problem is that ViaVoice is not free (as in freedom). Until IBM changes its license, we can't use it in Free World / Free Operating System and therefore it's not and will not be supported in Speech Daemon.
        - MBROLA

          MBROLA is a multi-lingual software synthesis available for GNU/Linux. MBROLA is not free as in freedom, although it's gratis. The same problems as with IBM ViaVoice prevents us to include it in Speech Daemon.

2. Speaking applications
   - Emacspeak

     The Emacspeak (by T. V. Raman <raman@cs.cornell.edu>) software package provides speech output for Emacs, and includes ,,speech servers" for the Dectalk speech synthesizers.

     The Emacspeak speech servers package provides servers for several additional synthesizers. None of these programs are normally run by the user directly. Instead, they are run by Emacs. That is: Emacs runs the Emacspeak code, which executes Tcl, which interprets the server code. This approach is too closely ,,wired" to usage with Emacspeak, so it can't be used for our general purposes.

     This does not mean, that these servers are completely a bad idea and we can not use them. Thanks to the author Jim Van Zandt <jrv@vanzandt.mv.com>, we can learn from the sources and write the output driver modules for Speech Daemon (emacspeak-ss is GPL).
   - GTK+ (Gnome Accessibility project)

     GNOME windowing toolkit library.
   - wxWindows

     Windowing toolkit library.
   - Java AWT

     Windowing toolkit library.
   - FOX toolkit

     Windowing toolkit library.
   - Speakup

     Speakup is a kernel patch that provides low level speech output for visually impaired, so it works even if there is some problem in configuration and you can't run Emacspeak.
   - Brltty

     Brltty is mainly a driver for different Braille displays, but also supports some kind of software synthesis.

We hope to be able to integrate Speech Daemon into these projects in the future.

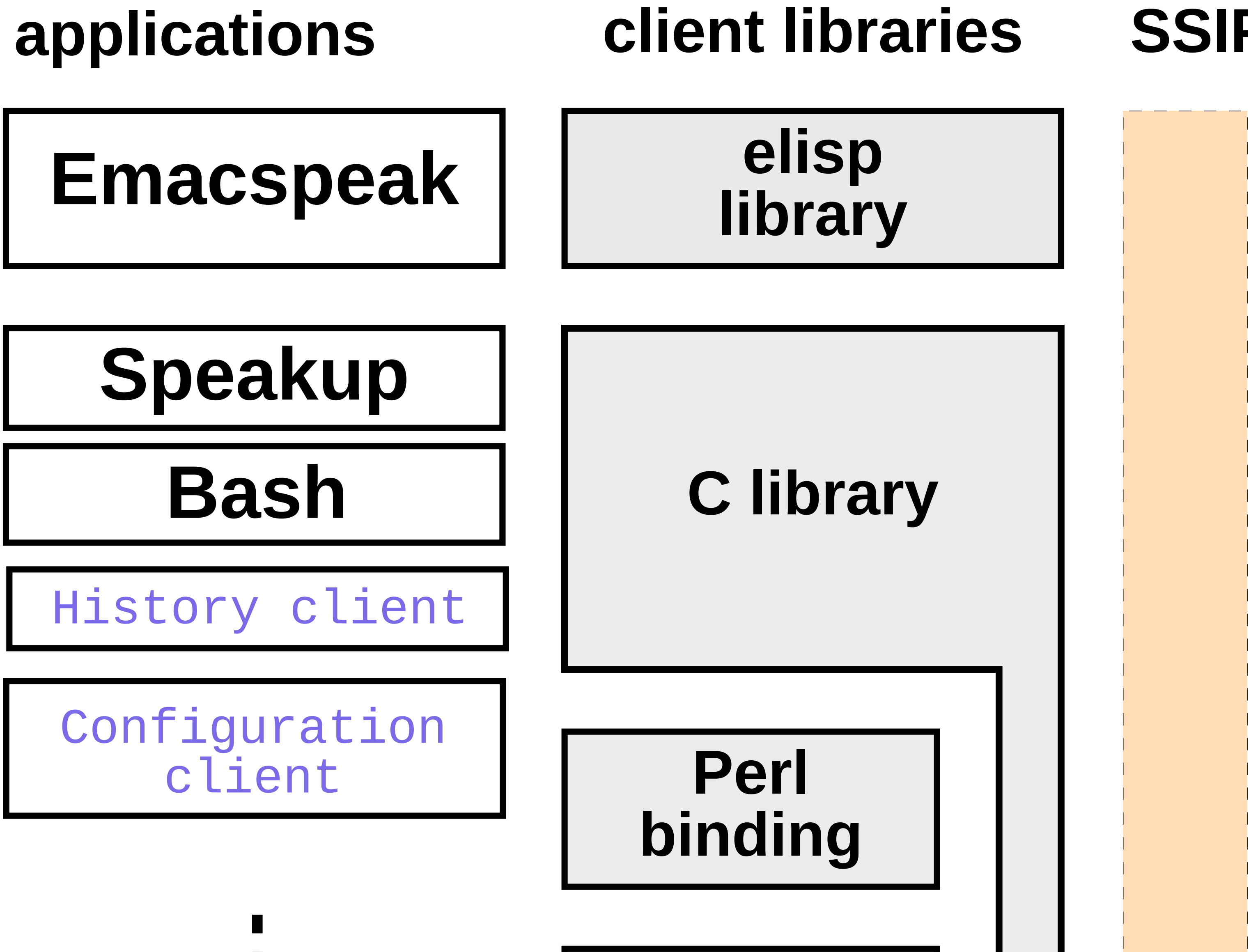
2.3 Design

The communication between all these applications and synthesizers is a great mess. For this purpose, we wanted Speech Daemon to be a layer separating applications and synthesizers so that applications wouldn't have to care about synthesizers and synthesizers wouldn't have to care about interaction with applications.

We decided we would implement Speech Daemon as a server receiving commands from applications over a protocol called `SSIP`, parsing them and if it's necessary and calling appropriate functions of output modules communicating with the different synthesizers. These output modules are implemented as plug-ins, so that the user can just load a new module if he wants to use new synthesizer.

Each client (application that wants to speaks) opens a socket connection to Speech Daemon and calls functions like spd_say(), spd_stop(), spd_pause() provided by the shared library. This shared library is still on the client side and sends Speech Daemon SSIP commands over the socket. When these arrive at Speech Daemon, it parses them, reads the text that should be said and put it in a queues according to the priority of this message and other criteria. It then decides when, with which parameters (set up by the client and the user) and on which synthesizer it will say the message. These requests are handled by the output plug-ins (output modules) for different hardware and software synthesizers and then said aloud.

See this figure:



See also the detailed description of SSIP, public API and module API.

2.4 User's point of view

In this section we will try to describe what can Speech Daemon offer to common users. But every programmer interested in this program should also read this because it's very important to understand.

Sketch:

- easy configuration of different speaking applications, central maintenance
- the ability to freely choose which synthesizer with which application
- less time devoted to configuration and tuning different applications and synthesis
- history of said messages for visually impaired

2.5 Programmer's point of view

Sketch:

- easy way to make your applications speak
- no time spent on configuration/debugging interface with different synthesizers
- no need to take care about configuration of voice
- easy way to make the application accessible to visually impaired people
- different facilities like the one providing a command line functionality

3 Invoking

3.1 Verbosity

There are 6 different verbosity levels of Speech Daemon logging. 0 means there is no output, while 5 means that nearly all the information about Speech Daemon working is written to standard output.

3.1.1 Level 0

No information.

3.1.2 Level 1

- Information about loading and exiting.

3.1.3 Level 2

- Information about errors that occurred.
- Allocating and freeing resources on start and exit.

3.1.4 Level 3

- Information about accepting/rejecting/closing clients' connections.
- Information about invalid client commands.

3.1.5 Level 4

- Every received command is output.
- Information about proceeding the command output
- Information about queueing/allocating messages.
- Information about the function of history, sound icons and other facilities.
- Information about the work of the speak() thread.

3.1.6 Level 5

This is only for debugging purposes and can output really *much* data. Use with caution.

- Also received data (messages etc.) is output.

4 Internal structure

4.1 Definitions

Server side is the side where Speech Daemon operates. It means server core, output modules and partly SSIP which is the layer for communication between server side and client side.

Client side is where particular applications wanting to speak are, where the shared library implementing public API is located and partly SSIP which is the layer for communication between server side and client side.

Client means an application that wants to speak or an application that is used to control Speech Daemon. (Of course different combinations are possible.)

Server core is the central part of Speech Daemon composed of two threads. One is listening on the user socket, parsing and proceeding incoming commands, and saving incoming text to queues. The other thread takes messages from queues and sends them to appropriate synthesizers.

Output module is a backend of Speech Daemon in the form of plug-in. It takes care of communication with the particular synthesizer and provides only abstract functions to the server core.

Shared library or *Public API* is a front-end of Speech Daemon that provides polished functions programmers should use to send commands to the server.

SSIP is the layer (communication protocol) between server side (server core) and client side (shared library). It stands for Speech Synthesis Internet Protocol. Client programs should never use it directly.

Socket or *File descriptor* represents the particular connection between a client and server. In C, it's and integer variable.

4.2 Server core

see sources, I'll try to write this section soon

4.3 Output modules

Output modules for Speech Daemon have the form of a glib plug-ins located in src/modules/. Each output module is a data structure composed of some parameters and pointers to it's functions.

```
typedef struct {
  gchar    *name;
  gchar    *description;
  gchar    *filename;
  gint     (*write)    (const gchar *, gint, void*);
  gint     (*stop)     (void);
```

```
    gint     (*is_speaking) (void);
    gint     (*close)    (void);
} OutputModule;
```

This structure is defined in '`intl/modules.h`' and therefore this header must be included in every plug-in source code.

```
#include "modules.h"
```

Also one other file called '`intl/fdset.h`' where the FDSetElement structure is defined must be included to be able to handle the different speech synthesis settings.

```
#include "fdset.h"
```

Each output module has associated a module_init function that is called at the starting of Speech Daemon. After doing the necessary initialization, it must return a filled structure of the type OutputModule (defined above).

```
OutputModule *module_init(void){
        ...
        return &module_flite;
}
```

Now what are the 4 functions: flite_write, flite_stop, flite_is_speaking and flite_close? This is the core of every output module and you have to define their bodies in the source code of your plug-in.

gint *synthesizer_write const gchar *data, gint len, TFDSetElement* set* [Output module functions]

This is the function where actual speech output is produced. It is called every time Speech Daemon decides to send a message to synthesis. The data of length *len* are passed in *data*. Additionally, the structure containing settings associated to this particular message is passed, however only few options are important for output modules.

Each output module should take care of setting the output device to these parameters (the other ones are handled independently in other parts of Speech Daemon):

- (signed int) set->speed
- (signed int) set->pitch
- (char*) set->language
- (int) set->voice_type

Speed and pitch are values between -100 and 100 included. 0 is the default value that represents normal speech flow. So -100 is the slowest (or lowest) and +100 is the fastest (or highest) speech.

(We should establish a constant scale referred to some text, standard speeds and different associated times. This will probably be the work of the person who will program the first real output module. We need to chose some longer text, decide what speed of reading we consider 0 and what we consider, say, +-50, measure the times needed to read it at these speeds and put it there in documentation as our standard scale.)

The language parameter is given as a null-terminated string containing the name of the language in English in lowercase (e.g. "english", "czech", "spanish").

voice_type is used only when the output module supports more types of voices for this particular language. The values represent (from '`intl/fdset.h`')

```
typedef enum {
    MALE = 0,
    FEMALE = 1,
    CHILD_MALE = 2,
    CHILD_FEMALE = 3
}EVoiceType;
```

We can consider also other voice types.

This function should return 0 if it fails and some non-0 value if the delivery to the synthesis is successful. Formerly we thought that it should return the number of bytes written, but it's still not clear how to handle messages that have to be divided in more parts (for example if the output device has a finite size buffer).

gint synthesizer_stop *void* [Output module functions]

This function should stop the synthesis of the currently spoken message immediately and throw away the rest of the message.

It should return 0 on success, -1 otherwise.

gint synthesizer_is_speaking *void* [Output module functions]

This function is very important to let Speech Daemon know how to regulate the speech flow between different queues, programs and even other synthesizers. On calling it, the output module must decide whether there is currently any output being produced in the speakers.

This can be a very hard problem and it's not clear how to do it with different synthesizers. If it's not possible to return an exact value, at least some estimate should be calculated. But such an inaccurate value can highly reduce the usefulness of an even otherwise very good plug-in. To some degree, this is still an open question.

It should return 0 if the synthesis is silent, 1 if it's speaking.

gint synthesizer_close *void* [Output module functions]

This function is called when Speech Daemon terminates. There are no special requirements on what the output plug-in should do.

It should return 0 on success, -1 otherwise.

5 Public API

6 Speech Synthesis Internet Protocol (SSIP)

Clients communicate with Speech Daemon via the Speech Synthesis Internet Protocol (SSIP). The protocol is the actual interface to Speech Daemon.

Usually, you don't need to use SSIP directly, you can use one of the programming interfaces, see Chapter 5 [Public API], page 12, wrapping SSIP with programming library calls. This is a recommended way of communication with Speech Daemon. However, in case your programming environment is not supported by any of the provided interfaces or you prefer to communicate with Speech Daemon directly for any reason, you can find the complete SSIP description here.

6.1 General rules

SSIP communicates with the clients through a defined set of text commands, in the way usual in common Internet protocols. The characters sent to and from Speech Daemon are encoded using the UTF-8 encoding.

Each SSIP command, unless specified otherwise, consists of exactly one line. The line is sent in the following format:

```
command arg ...
```

where *command* is a case insensitive command name and *arg*s are its arguments separated by spaces. The command arguments which come from a defined set of values are case insensitive as well. The number of arguments is dependent on the particular command and there can be commands having no arguments.

All lines of SSIP input and output must be ended with the pair of carriage return and line feed characters, in this order.

When you connect to Speech Daemon, you should at least set your client name, through the `SET CLIENT_NAME` command, Section 6.2.3 [Parameter setting commands], page 16. This is important to get a proper identification of your client — to allow managing it from the control center application and to identify it in a message history browser. You might want to set other connection parameters as well, look for more details in Section 6.2.3 [Parameter setting commands], page 16.

Connection to Speech Daemon is preferably closed by issuing the `QUIT` command, see Section 6.2.6 [Other commands], page 23.

SSIP is a synchronous protocol — you send commands and only after a complete response from SSIP arrives back you are allowed to send the next command. Usually, the connection to Speech Daemon remains open during the whole run of the particular client application. If you close the connection and open it again, you must set all the previously set parameters again, Speech Daemon doesn't store session parameters between connections.

The protocol allows you to perform commands regarding other currently connected or previously connected clients. This allows you to write a control application managing or browsing all the messages received by the current Speech Daemon process. The mechanism is completely relaxed, there are no restrictions on accessing messages of other clients and users and managing some aspects of their sound output.

SSIP replies of Speech Daemon are of the following format:

```
ccc-line 1
ccc-line 2
...
ccc-line n-1
ddd line n
```

where *n* is a positive integer, and *ccc* and *ddd* are three-digit long numeric codes identifying the result of the command. The last line determines the overall result of the command, the result code is followed by an English message describing the result of the action in a human readable form.

6.2 SSIP commands

Commands recognized by SSIP can be divided into several groups: Speech synthesis and sound output commands, speech control commands, parameter setting commands, commands retrieving information about current client and server settings, command handling the message history, and other commands. Each of these command groups is described in one of the following sections.

In the command descriptions, the command is written together with its arguments. Optional arguments are enclosed by square brackets (`[` and `]`), alternatives are separated by the vertical rule (`|`) and are grouped within braces (`{` and `}`) or square brackets for mandatory or optional arguments respectively, literal arguments values are typeset in lower letters (they are case insensitive), and variable arguments are typeset *like this*. Ellipsis denoted by three dots (`...`) means repetition (zero or more times) of all the arguments within the current brackets.

6.2.1 Speech synthesis and sound output

These commands invoke Speech Daemon mechanisms transforming given data and parameters into an audio sample and sending it onto an audio device. The particular way how the message is handled is defined by the Speech Daemon configuration mechanism (see Chapter 11 [Configuration], page 31) and are out of scope of SSIP.

`SPEAK` Start receiving a text message and synthesize it. After sending a reply to the command, Speech Daemon waits for the text of the message. The text can spread over any number of lines and is finished by an end of line marker followed by the line containing the single character `.` (dot). Thus the complete character sequence closing the input text is `CR LF . CR LF`. If any line within the sent text starts with a dot, an extra dot is prepended before it.

During reception of the text message, Speech Daemon doesn't send response to the particular lines sent. The response line is sent only immediately after the `SPEAK` command and after receiving the closing dot line.

Speech Daemon can start speech synthesis as soon as a sufficient amount of the text arrives, it generally needn't (but may) wait until the end of data marker is received.

There is no explicit upper limit on the size of the text, but the server administrator may set one in the configuration or the limit can be enforced by available system resources. If the limit is exceeded, the whole text is accepted, but its exceeding part is ignored and an error response code is returned after processing the final dot line.

This command, unlike all other commands, stores the received text into the message history.

`CHAR` *`char`*

Speak letter *char*. *char* can be any character representable by the UTF-8 encoding.

This command is intended to be used for speaking single letters, e.g. when reading a character under cursor or when spelling words.

`KEY` *`key-name`*

Speak key identified by *key-name*. The command is intended to be used for speaking keys pressed by the user.

key-name is a case sensitive symbolic key name. It is composed of a key name, optionally prepended with one or more prefixes, each containing an auxiliary key name and the underscore character.

Key name may contain any character excluding control characters (the characters in the range 0 to 31 in the ASCII table, characters in the range 128 to 159 in the Latin-* tables and other “invisible” characters), spaces, underscores, and double quotes.

The recognized key names are:

Any single UTF-8 character, excluding the exceptions defined above.

Any of the symbolic key names defined in Appendix A [Key names], page 32.

Examples of valid key names:

```
a
A
shift_a
shift_A
ú
$
enter
shift_kp-enter
control_alt_delete
control
```

`SOUND_ICON` *`icon-name`*

Send a sound identified by *icon-name* to the audio output. *icon-name* is a symbolic name of the given sound from the standard set listed in Appendix B [Standard sound icons], page 35, or another name from the particular Speech Daemon sound icon configuration.

6.2.2 Controlling speech output

These commands can stop or resume speech or audio output. They all affect only the synthesis process and output to a sound device, they do not affect the message history.

`STOP { id | all | self }`

Immediately stop outputting the current message (whatever it is — text, letter, key, or sound icon) from the identified client, if any is being output. If the command argument is `self`, last message from the current client connection is stopped. If it is `all`, stop currently output message or messages from all the clients. Otherwise, argument *id* must be given as an positive integer and the currently processed message from the client connection identified by *id* is stopped; if there is none such, do nothing.

`CANCEL { id | all | self }`

This command is the same as `SPEAK`, with the exception that it stops not yet output messages as well. All currently queued messages are stored into the message history without being sent to the audio output device.

`PAUSE { id | all | self }`

Stop audio output immediately, but do not discard anything. All the currently output and currently or later queued messages are postponed and saved for later processing, until the corresponding `RESUME` command is received.

The meaning of the command arguments is the same as in the `STOP` command.

`RESUME { id | all | self }`

Cancel the effect of the previously issued `PAUSE` command. Note that messages of the priority 3 received during the pause are not output (but they remain stored in the message history).

It is an error to send the `RESUME` command when the output corresponding to the given argument is not paused by a previous invocation of the `PAUSE` command. Such an error is signalled by a `4XX` return code.

The meaning of the command arguments is the same as in the `STOP` command.

6.2.3 Parameter setting

The `SET` command sets various control parameters of Speech Daemon. The parameter is always denoted by the first command argument.

All the settings take effect to the client connection (only) and until the parameter setting is changed by another invocation of the appropriate `SET` command or until the connection is closed.

`SET CLIENT_NAME user:client:component`

Set client's name. Client name consists of the user name, client (application) identification, and the identification of the component of the client (application). Each of the parts of the client name may contain only alphanumeric characters.

For example, for a client called `lynx` that creates Speech Daemon connection for its command processing, the name could be set in the following way:

```
SET CLIENT_NAME joe:lynx:cmd_processing
```

The client name is used in the server configuration settings, client listings and message history handling. All its three parts can be arbitrary, but it's important to define and follow rules for each application supporting Speech Daemon, so that a Speech Daemon user can configure all the aspects of the speech output easily.

Usually, this command should be sent as the very first command when a new connection to Speech Daemon is established.

`SET LANGUAGE` *`language`*

Set recommended language for this client to *language*. *language* is the name of the language according to RFC 1766.

For example, to set the preferred language to Czech, you send the following command:

```
SET LANGUAGE cs
```

`SET PRIORITY` *`n`*

Set message priority to *n*. *n* must be one of the values `1`, `2`, and `3`.

`SET PUNCTUATION { all | some | none }`

Set punctuation mode to the given value. `all` means read all punctuation characters, `none` read no punctuation characters, `some` means read only punctuation characters given in the server configuration or defined by the client's last `SET IMPORTANT_PUNCTUATION` command.

`SET IMPORTANT_PUNCTUATION` *`chars`*

Set punctuation characters read when `SET PUNCTUATION some` is set to those in *chars*. *chars* is a sequence of the required characters, without any spaces. *char* may not contain control characters and may not begin with double quotes.

`SET PUNCTUATION_TABLE` *`table`*

Use punctuation table *table* for spelling punctuation characters. *table* must be one of the punctuation table names returned to the command `LIST PUNCTUATION_TABLES` command, see Section 6.2.4 [Information retrieval commands], page 18.

`SET SPELLING_TABLE` *`table`*

Set spelling table to *table*. *table* must be one of the spelling table names returned to the command `LIST SPELLING_TABLES` command, see Section 6.2.4 [Information retrieval commands], page 18.

There is a standard set of spelling tables defined in Appendix C [Standard spelling tables], page 36.

`SET TEXT_TABLE` *`table`*

Set text table to *table*. *table* must be one of the text table names returned to the command `LIST TEXT_TABLES` command, see Section 6.2.4 [Information retrieval commands], page 18.

`SET SOUND_TABLE` *`table`*

Set sound table to *table*. *table* must be one of the text table names returned to the command `LIST SOUND_TABLES` command, see Section 6.2.4 [Information retrieval commands], page 18.

There is a standard set of sound tables defined in Appendix D [Standard sound tables], page 37.

`SET VOICE` *`name`*

Set the voice identified by *name*. *name* must be one of the voice identifiers returned to the command `LIST VOICES`, see Section 6.2.4 [Information retrieval commands], page 18.

There is a standard set of voice identifiers defined in Appendix E [Standard voices], page 38.

`SET RATE` *`n`*

Set the rate of speech. *n* is an integer value within the range from -100 to 100, with 0 corresponding to the default rate of the current speech synthesis output module, lower values meaning slower speech and higher values meaning faster speech.

`SET PITCH` *`n`*

Set the pitch of speech. *n* is an integer value within the range from -100 to 100, with 0 corresponding to the default pitch of the current speech synthesis output module, lower values meaning lower pitch and higher values meaning higher pitch.

`SET HISTORY { on | off }`

Enable (`on`) or disable (`off`) storing of received messages into history.

This command is intended for use by message history browsers and usually should not be used by other kinds of clients.

6.2.4 Retrieving information

The `LIST` command serves for retrieving information that can be presented to the user for selection of the values to the `SET` command. The information listed is selected according to the first argument of the `LIST` command.

`LIST SPELLING_TABLES`

List the names of all the spelling tables available on the server. Each table name is listed on a separate line. Each name may contain only alphanumeric characters and underscores.

Example Speech Daemon response:

```
200-sptable2
200-sptable1
200-sptable44
200-special-table
200 OK Tables listed.
```

The standard spelling tables are always listed, see Appendix C [Standard spelling tables], page 36.

`LIST PUNCTUATION_TABLES`

Similar to `LIST SPELLING_TABLES`, but lists the names of the available punctuation spelling tables.

`LIST TEXT_TABLES`

Similar to `LIST SPELLING_TABLES`, but lists the names of the available text mapping tables.

`LIST SOUND_TABLES`

Similar to `LIST SPELLING_TABLES`, but lists the names of the available sound mapping tables.

The standard sound tables are always listed, see Appendix D [Standard sound tables], page 37.

`LIST VOICES`

Similar to `LIST SPELLING_TABLES`, but lists the available voice names.

The standard voices are always listed, see Appendix E [Standard voices], page 38.

6.2.5 History handling

History is handled by the `HISTORY` command. It can take many forms, described below, that allow browsing, retrieving and repeating stored messages. In each invocation of the `HISTORY` command there is no difference between processing spoken or not spoken messages, all the received messages are processed.

There can be *history cursor* pointing on some message in the history. You can move it across history messages and retrieve the message the cursor is pointing to, using the `HISTORY CURSOR` set of command arguments described below.

`HISTORY GET CLIENT_LIST`

List known client names, their identifiers and status. Each connection is listed on a separate line in the following format:

```
id name status
```

where *id* is a client id that can be used in other history handling requests or in the speech output control commands (see Section 6.2.2 [Speech output control commands], page 16), *name* is the client name as set through the `SET CLIENT_NAME` command, and *status* is `1` for connected clients and `0` for disconnected clients. *id*s are unique within a single run of Speech Daemon.

Sample reply of Speech Daemon:

```
240-0 joe:speechd_client:main 0
240-1 joe:speechd_client:status 0
240-2 unknown:unknown:unknown 1
240 OK CLIENTS LIST SENT
```

`HISTORY GET CLIENT_ID`

Return id of the client itself.

The id is listed on a separate line in the following format:

```
id
```

Example:

```
200-123
200 OK CLIENT ID SENT
```

`HISTORY GET CLIENT_MESSAGES { id | all | self } start number`

List identifiers of messages sent by the client identified by *id*. If the special identifier `all` is used, identifiers of messages sent by all clients are listed; if the special identifier `self` is used, identifiers of messages sent by this client are listed.

number of messages is listed, starting from the message numbered *start*. Both *number* and *start* must be positive integers. The first message is numbered 1, the second 2, etc. If the given range exceeds the range of available messages, no error is signalled and the given range is restricted to the available range of messages.

Messages are sorted by the criterion used in the last client's invocation of the `HISTORY SORT` command. If no `HISTORY SET` has been invoked yet, the messages are sorted from the oldest to the newest, according to their time of arrival to Speech Daemon.

Each message id is listed, together with other information, on a separate line, in the following format:

```
id client-id client-name "time" priority "intro"
```

client-id is a numeric identifier of the client which sent the message, *client-name* is its name as set by the `SET CLIENT_NAME` command, see Section 6.2.3 [Parameter setting commands], page 16. *time* is the time of arrival of the message, in the fixed length `YYYY-MM-DD HH:MM:SS` format. *priority* is the priority of the message, one of the values accepted by the `SET PRIORITY` command, see Section 6.2.3 [Parameter setting commands], page 16.

intro is the introductory part of the message of a certain maximum length, see the `HISTORY SET SHORT_MESSAGE_LENGTH` command. *intro* does not contain any double quotes nor the line feed character.

All the message identifiers in the history, regardless of clients that issued them, are unique within a single run of Speech Daemon and remain unchanged.

`HISTORY GET LAST`

List the id of the last message sent by the client.

The id is listed on a separate line of the following format:

```
id
```

If the client haven't sent any message yet, return an error code.

`HISTORY GET MESSAGE id`

Return the text of the history message identified by *id*. If *id* doesn't refer any message, return an error code instead. The text is sent as a multi-line message, with no escaping or special transformation.

An example SSIP response to the command:

```
200-Hello, world!
200-How are you?
200 OK MESSAGE SENT
```

`HISTORY CURSOR GET`

Get the id of the message the history cursor is pointing on.

The id is listed on a separate line. Sample Speech Daemon reply to this command:

```
243-42
243 OK CURSOR POSITION RETURNED
```

`HISTORY CURSOR SET {` *id* `| all | self } { first | last | pos` *n* `}`

Set the history cursor to the given position. The meaning of the first argument after `SET` is the same as in the `HISTORY GET CLIENT_MESSAGES` command. The argument `first` asks to set the cursor on the first position and the argument `last` asks to set the cursor on the last position of the history of the given client. If the argument `pos` is used, the position is set to *n*, where *n* is a positive integer. It is an error if *id* doesn't identify any client or if *n* doesn't point to any existing position in the history.

As for the order and numbering of the messages in the history, the same rules apply as in `HISTORY GET CLIENT_MESSAGES`, see above.

`HISTORY CURSOR { forward | backward }`

Move the cursor one position `forward`, resp. `backward`, within the messages of the client specified in the last `HISTORY CURSOR SET` command. If there is no next, resp. previous, message, don't move the cursor and return an error code.

`HISTORY SAY` *id*

Speak the message from history identified by *id*. If *id* doesn't refer any message, return an error code instead.

The message is spoken as it would be sent by its originating command (`SPEAK` or `SOUND_ICON`), but the *current* settings (priority, etc.) apply.

`HISTORY SORT { asc | desc } { time | user | client_name | priority | message_type }`

Sort the messages in history according to the given criteria. If the second command argument is `asc`, sort in the ascending order, if it is `desc`, sort in the descending order. The third command argument specifies the message property to order by:

`time`	Time of arrival of the message.
`user`	User name.
`client_name`	Client name, excluding user name.
`priority`	Priority.
`message_type`	Type of the message (text, sound icon, character, key), in the order specified in the Speech Daemon configuration or by the `HISTORY SET MESSAGE_TYPE_ORDERING` command.

The sorting is stable — order of all the messages that are equal in the given ordering remains the same.

The sorting is specific to the given client connection, other connections are unaffected by invocation of this command.

`HISTORY SET SHORT_MESSAGE_LENGTH` *`length`*

Set the maximum length of short versions of history messages to *length* characters. *length* must be a non-negative integer.

Short (truncated) versions of history messages are used e.g. in the answer to the `HISTORY GET CLIENT_MESSAGES` format.

`HISTORY SET MESSAGE_TYPE_ORDERING "`*`ordering`*`"`

Set the ordering of the message types, from the minimum to the maximum. *ordering* is a sequence of the following symbols, separated by spaces: `text`, `sound_icon`, `char`, `key`. The symbols are case insensitive and each of them must be present in *ordering* exactly once.

The specified ordering can be used by the `HISTORY SORT` command.

`HISTORY SEARCH {` *`id`* `| all | self } "`*`condition`*`"`

Return the list of history messages satisfying *condition*. The command allows searching messages by given words. The output format is the same as of the `HISTORY GET CLIENT_MESSAGES` command.

The meaning of the first argument after `SEARCH` is the same as in the `HISTORY GET CLIENT_MESSAGES` command.

condition is constructed according to the following grammar rules:

`condition` `::` *`word`*

Matches messages containing *word*.

`condition` `:: ( !` *`condition`* `)`

Negation of the given condition.

`condition` `:: (` *`condition`* `[ &` *`condition`* `... ] )`

Logical AND — all the conditions must be satisfied.

`condition` `:: (` *`condition`* `[ |` *`condition`* `... ] )`

Logical OR — at least one of the conditions must be satisfied.

Spaces within the condition are insignificant and ignored.

The following rules apply to *word*s:

- *word* is a sequence of adjacent alphanumeric characters.
- If *word* contains any upper-case letter, the search for the word is case sensitive, otherwise it's case insensitive.
- *word* must match whole word, not only its substring.
- *word* can contain the wild card characters `?`, substituting any single alphanumeric character, and `*`, substituting any number (incl. zero) of alphanumeric characters.

Returned messages are sorted by the following rules:

1. The primary sorting is defined by the number of the satisfied subconditions on the top level of the given condition, from the highest (best matching messages first) to the lowest. This takes any effect only if the given condition is the OR rule.
2. The criterion used in the last client's invocation of the `HISTORY SORT` command. If no `HISTORY SORT` has been invoked yet, the messages are sorted from the oldest to the newest, according to their time of arrival to Speech Daemon.

6.2.6 Other commands

`QUIT`	Close the connection.
`HELP`	Print a short list of all SSIP commands, as a multi-line message.

6.3 Return codes

Each line of the SSIP output starts with a three-digit numeric code of the form *NXX* where *N* determines the result group and *xx* denotes the finer classification of the result.

SSIP defines the following result groups:

1xx	Informative response — general information about the protocol, help messages.
2xx	Operation was completely successful.
3xx	Server error, problem on the server side.
4xx	Client error, invalid arguments or parameters received.
5xx	Client error, invalid command syntax, unparseable input.

Result groups *1xx* and *2xx* correspond to successful actions, other groups to unsuccessful actions. Only the groups defined here may be returned from the Speech Daemon.

Currently, only the meaning of the first digit of the result code is defined, the last two digits are insignificant and can be of any value. Clients shouldn't rely on the unspecified digits in any way. If you are going to write your own SSIP implementation, please consult the authors of Speech Daemon to define more precise set of return codes.

6.4 Example of an SSIP relation

The following example illustrates a sample relation with SSIP. The client connects to the Speech Daemon, sets all the common parameters, sends two text messages, displays the list of clients, instructs Speech Daemon to repeat the second message, and closes the connection. Lines starting with a numeric code are response lines of the server, other lines are the lines sent by the client.

```
SET CLIENT_NAME joe:vi:default
208 OK CLIENT NAME SET
SET PRIORITY 2
```

```
202 OK PRIORITY SET
SPEAK
230 OK RECEIVING DATA
Hello, I'm a Speech Daemon communication example!
How are you?
.
225 OK MESSAGE QUEUED
SPEAK
230 OK RECEIVING DATA
Still there?
.
225 OK MESSAGE QUEUED
HISTORY GET CLIENT_LIST
240-1 jim:Emacs:default 0
240-2 jim:Emacs:default 0
240-3 unknown:unknown:unknown 0
240-4 jim:Emacs:default 1
240-5 joe:vi:default 1
240 OK CLIENTS LIST SENT
HISTORY GET LAST
242-39 joe:vi:default
242 OK LAST MSG SAID
QUIT
231 HAPPY HACKING
```

7 Priorities

The possibility to distinguish between several message priority levels seems to be essential. Each message sent by client to speech server should have a priority level assigned.

Speech Daemon provides the system of three priority levels. Every message will either contain explicit level information, or the default value will be considered. There is a separate message queue for each of the levels. The behavior is as follows:

7.1 Level 1

These messages will be said immediately as they come to server. They are never interrupted. These messages should be as short as possible, because they block the output of all other messages. When several concurrent messages are received by server, they are queued and said in the order, they came. When a new message of level 1 comes during lower level message is spoken, lower level message is canceled and removed from the queue (this message is already stored in the history)

7.2 Level 2

Second level messages are said in the moment, when there is no message of level 1 queued. Several messages of level 2 are said in the order, they are received (queued, but in their own queue). This is the default level for messages without explicit level information.

7.3 Level 3

Third level messages are only said, when there are no messages of any higher level queued. If there are level 3 messages being said or waiting in queues, they are interrupted by the last incoming level 3 message and this one is said, in other words, level 3 is interrupting itself.

7.4 How to use them wisely

Example uses for level **one** are:

error messages
very important messages
...

Example uses for level **two** are:

regular program messages
menus
text the user is working on
...

Example uses for level **three** are:

less important status information
letters when typing input
...

8 Multiple output modules

Speech Daemon supports concurrent use of multiple output modules. In the case these output modules provide good synchronization, you can combine them in reading messages. For example if module1 can speak English and Czech while module2 speaks only German, the idea is that if there is something message in German, module2 is used, while module1 is used for the other languages. These rules for selection of output modules can be influenced through the configuration file ‘`speechd.conf`’.

If you want to compile and use a new output module, you should place it in ‘`src/modules`’ in your source directory of Speech Daemon and add it to ‘`src/modules/Makefile.am`’. You can compile and install it by typing: `make; su root; make install`. The last step you have to do is to let Speech Daemon know you want to use this new module by adding a line to ‘`speechd.conf`’ in your configuration directory

```
AddModule module_name
```

and possibly also changing the line

```
DefaultModule new_module
```

to make it default.

See Section 4.3 [Output modules], page 9.

9 Message history

9.1 Access rights

To protect privacy of users, Speech Daemon restricts history access to a certain subset of all the received messages. The following rules apply:

- All the messages issued by a client connection are accessible to that client connection.
- All the messages sent by a given user are accessible to that user.
- All the messages sent by the user `speechd` are accessible to all users on the system running the Speech Daemon instance present in the group `speechd`.
- No other messages are accessible.

Two users are considered the same, if and only if their connections originate on the same host, their user names are the same, and their identity can be checked, as described bellow. Speech daemon does not provide any explicit authentication mechanism. To check the identity of users, Speech Daemon uses the Identification Protocol mechanism defined by RFC 1413 to get the user's identity. If user's identity cannot be checked, the user is considered different of all other connected or previously connected users.

Speech Daemon allows to specify user mapping in its configuration, allowing to change certain users to different users, see Chapter 11 [Configuration], page 31.

10 Speech parameters

10.1 Language selection

Various synthesizers provide different sets of possible languages, they are allowed to speak. We must be able to receive a request for setting particular language (using ISO language code) and reply, if the language is supported.

10.2 Speed

Sped of the speech is supported by all synthesizers, but the values and their ranges differ. Each output module is responsible to set the speed to the value, best responding to current setting. This may be a little bit difficult, because there is no exact scale. We could take some longer English paragraph and take it as a base for our new scale. If this paragraph is said in e.g. ten seconds, this means speed = 100, if it is said in twenty seconds, speed = 200. This way, we can coordinate different scales quite precisely (the paragraph should be long enough).

10.3 Punctuation mode

Punctuation mode describes the way, in which the synthesizer works with non-alphanumeric characters. Most synthesizers support several punctuation modes. We will support a reasonable superset of those modes, which may be implemented in device driver, when not supported by hardware.

10.4 Prosody

Prosody setting allows us, to distinguish punctuation characters in spoken text, as we are familiar in normal speech. This means the way, we pronounce the text with interrogation mark, coma, dot etc.

10.5 Pitch

Pitch is the voice frequency. We face the similar problems here, as with Speed setting.

10.6 Voice type

Most synthesizers provide several voice types, such as male, female, child etc. The set is again different for each of the devices. Speech Daemon should try to find the nearest possible (if the request is child female and it's not available, we will try to use adult female rather then adult male).

10.7 Spelling

Spelling mode is provided by nearly all devices and is also easy to emulate in output module.

10.8 Capital letters recognition

That is again a widely supported feature. However it is desirable to support this internally, using the sound icons feature, but this requires a good possibility of synchronization, which is not possible with all devices.

11 Configuration

Speech Daemon can be configured on several levels. There is a configuration file where permanent settings are stored, but user and applications can also change the majority of parameters on-fly by calling Speech Daemon functions. The third level of configuration can't be changed and it's given by the capabilities of each output device (each output module for each output device reports it's capabilities when it's loaded into Speech Daemon).

We use DotConf for the permanent text file based configuration. See '`speechd.conf`'.

Other parts of this manual deal with the runtime configuration.

Appendix A Key names

This appendix defines all the recognized symbolic key names. The names are case sensitive.

A.1 Auxiliary keys

`control`

`hyper`

`meta`

`shift`

`super`

A.2 Control character keys

`backspace`
`break`

`delete`

`down`

`end`

`enter`

`escape`

`f1`

`f2`

`f3`

`f4`

`f5`

`f6`

`f7`

`f8`

`f9`

`f10`

`f11`

`f12`

`f13`

`f14`

`f15`

`f16`

`f17`

`f18`

`f19`

`f20`

`f21`

`f22`

`f23`

`f24`

`home`

`insert`

`kp-*`

`kp-+`

`kp--`

`kp-.`

`kp-/`

`kp-0`

`kp-1`

`kp-2`

`kp-3`

`kp-4`

`kp-5`

`kp-6`

`kp-7`

`kp-8`

`kp-9`

`kp-enter`

`left`

`menu`

`next`

`num-lock`

`pause`

`print`

`prior`

`return`

`right`

`scroll-lock`
`space`

`tab`

`up`

`window`

A.3 Special key names

`space`

`underscore`
`double-quote`

Appendix B Standard sound icons

There are none currently.

Appendix C Standard spelling tables

The following spelling table names are always present in the output of the `LIST SPELLING tables` command (see Section 6.2.4 [Information retrieval commands], page 18):

```
spelling-short
spelling-long
```

Appendix D Standard sound tables

There are none currently.

Appendix E Standard voices

The following voice names are always present in the output of the `LIST VOICES` command (see Section 6.2.4 [Information retrieval commands], page 18):

`MALE1`

`MALE2`

`MALE3`

`FEMALE1`

`FEMALE2`

`FEMALE3`

`CHILD_MALE`
`CHILD_FEMALE`

The actual presence of any of these voices is not guaranteed. But the command `SET VOICE` (see Section 6.2.3 [Parameter setting commands], page 16) must accept any of them. If the given voice is not available, it is mapped to another voice by the output module.

Appendix F GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program," below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification.") Each licensee is addressed as "you."

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you

indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version," you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 20yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program ‘Gnomovision’
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix G GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

G.0.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept index

Table of Contents