The Libgsasl Reference Manual

for version 0.0.1, 12 October 2002

Simon Josefsson (bug-libgsasl@josefsson.org)

This is *The Libgsasl Reference Manual*, last updated 12 October 2002, for Version 0.0.1 of the LIBGSASL library.

Table of Contents

1 Introduction

'Libgsasl' is a library that provides a Simple Authentication and Security Layer (SASL) interface for applications. Included are also a set of well known SASL mechanisms.

1.1 Getting Started

This manual documents the 'Libgsasl' library programming interface. All functions and data types provided by the library are explained.

The reader is assumed to possess basic familiarity with SASL.

This manual can be used in several ways. If read from the beginning to the end, it gives a good introduction into the library and how it can be used in an application. Forward references are included where necessary. Later on, the manual can be used as a reference manual to get just the information needed about any particular interface of the library. Experienced programmers might want to start looking at the examples at the end of the manual, and then only read up those parts of the interface which are unclear.

1.2 Features

'Libgsasl' might have a couple of advantages over other libraries doing a similar job.

It's Free Software
: Anybody can use, modify, and redistribute it under the terms of the GNU Lesser General Public License (see Appendix B [Library Copying], page 48).

It's thread-safe
: No global variables are used and multiple library handles and session handles may be used in parallell.

It's internationalized
: It handles non-ASCII username and passwords and user visible strings used in the library (error messages) can be translated into the users' language.

It's portable
: It should work on all Unix like operating systems, including Windows.

Note that the library do not implement any policy to decide whether a certain user is "authenticated" or not. Rather, it uses callbacks back into the application to answer these questions.

1.3 SASL Overview

This section describes SASL from a protocol point of view[1].

The Simple Authentication and Security Layer (SASL) is a method for adding authentication support to connection-based protocols. A protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating a security layer for subsequent protocol interactions.

[1] The text is a lightly adapted version of the introduction section from RFC 2222 by John G. Myers, copyright 1997 by The Internet Society.

The command has a required argument identifying a SASL mechanism. SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores.

If a server supports the requested mechanism, it initiates an authentication protocol exchange. This consists of a series of server challenges and client responses that are specific to the requested mechanism. The challenges and responses are defined by the mechanisms as binary tokens of arbitrary length. The protocol's profile then specifies how these binary tokens are then encoded for transfer over the connection.

After receiving the authentication command or any client response, a server may issue a challenge, indicate failure, or indicate completion. The protocol's profile specifies how the server indicates which of the above it is doing.

After receiving a challenge, a client may issue a response or abort the exchange. The protocol's profile specifies how the client indicates which of the above it is doing.

During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity (frequently known as a userid) from the client to server, and negotiates the use of a mechanism-specific security layer. If the use of a security layer is agreed upon, then the mechanism must also define or negotiate the maximum cipher-text buffer size that each side is able to receive.

The transmitted authorization identity may be different than the identity in the client's authentication credentials. This permits agents such as proxy servers to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying. With any mechanism, transmitting an authorization identity of the empty string directs the server to derive an authorization identity from the client's authentication credentials.

If use of a security layer is negotiated, it is applied to all subsequent data sent over the connection. The security layer takes effect immediately following the last response of the authentication exchange for data sent by the client and the completion indication for data sent by the server. Once the security layer is in effect, the protocol stream is processed by the security layer into buffers of cipher-text. Each buffer is transferred over the connection as a stream of octets prepended with a four octet field in network byte order that represents the length of the following buffer. The length of the cipher-text buffer must be no larger than the maximum size that was defined or negotiated by the other side.

2 Preparation

To use 'Libgsasl', you have to perform some changes to your sources and the build system. The necessary changes are small and explained in the following sections. At the end of this chapter, it is described how the library is initialized, and how the requirements of the library are verified.

A faster way to find out how to adapt your application for use with 'Libgsasl' may be to look at the examples at the end of this manual (see Chapter 10 [Examples], page 39).

2.1 Header

All interfaces (data types and functions) of the library are defined in the header file 'gsasl.h'. You must include this in all programs using the library, either directly or through some other header file, like this:

```
#include <gsasl.h>
```

The name space of 'Libgsasl' is `gsasl_*` for function names, `Gsasl*` for data types and `GSASL_*` for other symbols. In addition the same name prefixes with one prepended underscore are reserved for internal use and should never be used by an application.

2.2 Initialization

'Libgsasl' must be initialized before it can be used. The library is initialized by calling `gsasl_init()` (see Chapter 5 [Global Functions], page 23). The resources allocated by the initialization process can be released if the application no longer has a need to call 'Libgsasl' functions, this is done by calling `gsasl_done()`.

In order to take advantage of the internationalisation features in 'Libgsasl', such as translated error messages, the application must set the current locale using `setlocale()` before initializing 'Libgsasl'.

2.3 Version Check

It is often desirable to check that the version of 'Libgsasl' used is indeed one which fits all requirements. Even with binary compatibility new features may have been introduced but due to problem with the dynamic linker an old version is actually used. So you may want to check that the version is okay right after program startup.

`const char * gsasl_check_version (const char * req_version)` Function

req_version: version string to compare with, or NULL

Check that the the version of the library is at minimum the one given as a string in *req_version* and return the actual version string of the library; return NULL if the condition is not met. If `NULL` is passed to this function no check is done and only the version string is returned. It is a pretty good idea to run this function as soon as possible, because it may also intializes some subsystems. In a multithreaded environment if should be called before any more threads are created.

2.4 Building the source

If you want to compile a source file including the 'gsasl.h' header file, you must make sure that the compiler can find it in the directory hierarchy. This is accomplished by adding the path to the directory in which the header file is located to the compilers include file search path (via the '`-I`' option).

However, the path to the include file is determined at the time the source is configured. To solve this problem, 'Libgsasl' ships with a small helper program `libgsasl-config` that knows the path to the include file and other configuration options. The options that need to be added to the compiler invocation at compile time are output by the '`--cflags`' option to `libgsasl-config`. The following example shows how it can be used at the command line:

```
gcc -c foo.c `libgsasl-config --cflags`
```

Adding the output of '`libgsasl-config --cflags`' to the compilers command line will ensure that the compiler can find the 'Libgsasl' header file.

A similar problem occurs when linking the program with the library. Again, the compiler has to find the library files. For this to work, the path to the library files has to be added to the library search path (via the '`-L`' option). For this, the option '`--libs`' to `libgsasl-config` can be used. For convenience, this option also outputs all other options that are required to link the program with the 'Libgsasl' libararies (in particular, the '`-lgsasl`' option). The example shows how to link '`foo.o`' with the 'Libgsasl' library to a program `foo`.

```
gcc -o foo foo.o `libgsasl-config --libs`
```

Of course you can also combine both examples to a single command by specifying both options to `libgsasl-config`:

```
gcc -o foo foo.c `libgsasl-config --cflags --libs`
```

3 Using the Library

After initialization of the library, the core part of the library is run within a loop until it has finished. The library is handed input from the other protocol entity and results in output which is to be sent to the other entity, or an error code. The library does not send data to the server itself, but only return it in buffers. The main interface to the library uses binary data, but since many common protocols uses Base 64 encoded data, a wrapper around the main function is also provided.

The following pseudo code illustrates how the library is used in a simple client. All the functions used are explained later on in this manual.

```
main()
{
  Gsasl_ctx          *ctx;
  Gsasl_session_ctx  *cctx;
  char *input, output[BUFFERSIZE];
  size_t output_len;
  int rc;

  rc = gsasl_init (&ctx);
  if (rc != GSASL_OK)
    die(gsasl_strerror(rc));

  /* XXX Set callbacks here */

  /* Read supported SASL mechanism from server */
  input = read_from_client();

  /* Select a good mechanism */
  mech = gsasl_client_suggest_mechanism (ctx, input);
  if (mech == NULL)
    die("Cannot find any commonly agreed SASL mechanism...");

  /* Start to use it */
  res = gsasl_client_start (ctx, mech, &cctx);
  if (res != GSASL_OK)
    die(gsasl_strerror (rc));

  input = NULL;
  do
    {
      /* Do one SASL step and unless we're done, send the output to
         server and read new data from server */

      rc = gsasl_client_step_base64 (cctx, input, output, BUFFERSIZE);
      if (rc != GSASL_NEEDS_MORE)
        break;

      write_to_client(output);
```

```
      input = read_from_client();
    }
  while (rc == GSASL_NEEDS_MORE);

  if (rc != GSASL_OK)
    die("Authentication failed... %s\n", gsasl_strerror(rc);

  /* Client is now authenticated -- proceed with actual protocol... */

  gsasl_client_finish (cctx);
  gsasl_done (ctx);
}
```

Notice the XXX comment that said you should specify the callbacks to use there. 'Libgsasl' depend on callbacks to implement user interaction (in the client) and user validation (in the server). If you don't specify any callbacks, very few mechanisms will be supported (like EXTERNAL that don't need any additional information, see Section 4.1 [EXTERNAL], page 8). Since we are building a simple client, we define callbacks which are used by several SASL mechanisms to get username and password. We start by defining the function for querying the username, following the prototype for `Gsasl_client_callback_authentication_id` for the LOGIN mechanism (see Section 4.4 [LOGIN], page 11) .

```
int
callback_username (Gsasl_session_ctx *ctx,
                   char *out,
                   size_t *outlen)
{
  char username[BUFFERSIZE];

  if (out == NULL)
    *outlen = BUFFERSIZE;
  else
    {
      fprintf(stdout, "Enter username: ");
      fgets(username, BUFFERSIZE, stdin);
      *outlen = strlen(BUFFERSIZE);
    }

  return GSASL_OK;
}
```

As you can see, this is a simplistic function that reads a username from the user. The callback for entering the password is similar and follows the `Gsasl_client_callback_password` prototype:

```
int
callback_password (Gsasl_session_ctx *ctx,
                   char *out,
                   size_t *outlen)
{
  char password[BUFFERSIZE];
```

```
  if (out == NULL)
    *outlen = BUFFERSIZE;
  else
    {
      fprintf(stdout, "Enter password: ");
      fgets(password, BUFFERSIZE, stdin);
      *outlen = strlen(BUFFERSIZE);
    }

  return GSASL_OK;
}
```

In reality, the program should probably inhibit echo of the password to the terminal, but that is left as an exercise for the reader.

Now having implemented the callbacks, we are ready to replace the XXX comment with real code that set the callbacks (see Chapter 6 [Callback Functions], page 25). The following does it.

```
gsasl_client_callback_authentication_id_set(ctx, callback_username);
gsasl_client_callback_authorization_id_set(ctx, callback_username);
gsasl_client_callback_password_set(ctx, callback_password);
```

Notice that we use the same callback for the authentication identity and the authorization identity. In reality, this may be too simplistic, but will do for an example.

The simple client is now complete, and will be able to support SASL mechanisms such as PLAIN and CRAM-MD5.

Implementing a server is very similar to the client, the only difference is that you use gsasl_server_*() functions instead of gsasl_client_*() and instead of implementing Gsasl_client_* callbacks implement some Gsasl_server_* callbacks. See each mechanism (see Chapter 4 [Mechanisms], page 8) for details on which callbacks are required and their prototype.

A note for server authors is in place, on the optional initial client output (discussed in section 5.1 of RFC 2222). In a server looking similar to the code above, the first call to `gsasl_server_step_base64` would have a *input* set to NULL. The mechanisms interprete this as your protocol do not support initial client output. If the protocol in which you implement SASL supports initial client output, the first call to `gsasl_server_step_base64` should include a real buffer with the initial client data.

One note for client authors is in place. The code above aborts processing if 'Libgsasl' did not come out of the loop with a GSASL_OK exit code. It is a mistake to not require this, and instead only look at what the server is sending you. Even if the server said you are authenticated, it does not always mean that the SASL mechanism is satisfied. This is specifically true for SASL client mechanisms which perform server authentication. Thus, if you only trust what the server replied instead of requireing a GSASL_OK result, you may open up for fake servers. Don't shortcut the loop with a positive server response.

4 Mechanisms

Different SASL mechanisms have different requirements on the application using it. Some simpler mechanisms, such as LOGIN and PLAIN, are straight forward to hook into existing authentication systems (such as '`/etc/passwd`' via PAM). The client callback for these mechanisms is easy to implement, the user is simply queried for the username and password. The server callbacks pass on the username and password into the policy deciding authentication system (e.g. PAM).

Other mechanism like CRAM-MD5, DIGEST-MD5, and SRP uses hashed passwords. The client callback are the same as for PLAIN and LOGIN. However, the server do not receive the plaintext password via the network but rather a hash of it. Existing policy deciding systems like PAM cannot handle this, so the server callback for these mechanisms are more complicated.

Further mechanisms like GSSAPI (Kerberos 5) assume a specific authentication system. In theory this means that 'Libgsasl' would not need to interact with the application, but rather call this specific authentication system directly. However, some callbacks are supported anyway, to modify the behaviour of how the specific authentication system is used.

Special mechanisms like EXTERNAL and ANONYMOUS are entirely dependent on callbacks.

4.1 The EXTERNAL mechanism

The EXTERNAL mechanism is used to authenticate a user to SASL when SASL is used in an environment which has already authenticated the user. It is often used within TLS or IPSEC protected channels.

This mechanism is only enabled in the server if you implement the callback below and set them in the library (see Chapter 6 [Callback Functions], page 25). It is always enabled in the client as there are no client callbacks.

`int` (***Gsasl_server_callback_external**) (`Gsasl_session_ctx *` *ctx*) — Prototype

ctx: libgsasl handle.

Type of callback function the application implements. It should return `GSASL_OK` if user is authenticated by out of band means, otherwise `GSASL_AUTHENTICATION_ERROR`.

4.2 The ANONYMOUS mechanism

The ANONYMOUS mechanism is used to "authenticate" clients to anonymous services; or rather just indicate that the client wishes to use the service anonymously. The client sends a token, usually her email address.

This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_anonymous**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with client token.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with some input from the user and set the output array length, and return `GSASL_OK`, or fail with an error code.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_anonymous**) (`Gsasl_session_ctx *` *ctx*, `const char *` *token*) Prototype

ctx: libgsasl handle.

ctx: output array with client token.

ctx: on input the maximum size of the output array, on output contains the actual size of the output array. If OUT is

Type of callback function the application implements. It should return `GSASL_OK` if user should be permitted anonymous access, otherwise `GSASL_AUTHENTICATION_ERROR`.

4.3 The PLAIN mechanism

The PLAIN mechanism uses username (authentication identity and authorization identity) and password to authenticate users. Two ways of validating the user is provided, either by having the SASL mechanism retrieve the raw password from the application and perform the validation internally, or by calling the application with authentication identity, authorization identity and password and let it decide. If both the validating and the retrieving callbacks are specified by the application, the validating one will be used.

This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authorization_id**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_authentication_id**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authentication identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authentiction identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authentication identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_password**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with password.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with password of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The password must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_validate**) (`Gsasl_session_ctx *` *ctx*, `char *` *authorization_id*, `char *` *authentication_id*, `char *` *password*) Prototype

ctx: libgsasl handle.

authorization_id: input array with authorization identity.

authentication_id: input array with authentication identity.

password: input array with password.

Type of callback function the application implements. It should return GSASL_OK if and only if the validation of the provided credential was succesful. GSASL_AUTHENTICATION_ERROR is a good failure if authentication failed, but any available return code may be used.

`int` (***Gsasl_server_callback_retrieve**) (`Gsasl_session_ctx *` *ctx*, `char *` *authentication_id*, `char *` *authorization_id*, `char *` *realm*, `char *` *key*, `size_t *` *keylen*) Prototype

ctx: libgsasl handle.

authentication_id: input array with authentication identity.

authorization_id: input array with authorization identity, or NULL.

realm: input array with realm of user, or NULL.

key: output array with key for authentication identity.

keylen: on input the maximum size of the key output array, on output contains the actual size of the key output array.

Type of callback function the application implements. It should retrieve the password for the indicated user and return GSASL_OK, or an error code such as GSASL_AUTHENTICATION_ERROR. The key must be encoded in UTF-8, but need not be normalized in any way.

If KEY is NULL, the function should only populate the KEYLEN output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

4.4 The LOGIN mechanism

The LOGIN mechanism uses username (authorization identity only) and password to authenticate users. Two ways of validating the user is provided, either by having the SASL mechanism retrieve the raw password from the application and perform the validation internally, or by calling the application with authorization identity and password and let it decide. If both the validating and the retrieving callbacks are specified by the application, the validating one will be used.

This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authorization_id**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_password***) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with password.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with password of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The password must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_validate***) (`Gsasl_session_ctx *` *ctx*, `char *` *authorization_id*, `char *` *authentication_id*, `char *` *password*) Prototype

ctx: libgsasl handle.

authorization_id: input array with authorization identity.

authentication_id: input array with authentication identity.

password: input array with password.

Type of callback function the application implements. It should return GSASL_OK if and only if the validation of the provided credential was succesful. GSASL_AUTHENTICATION_ERROR is a good failure if authentication failed, but any available return code may be used.

`int` (***Gsasl_server_callback_retrieve***) (`Gsasl_session_ctx *` *ctx*, `char *` *authentication_id*, `char *` *authorization_id*, `char *` *realm*, `char *` *key*, `size_t *` *keylen*) Prototype

ctx: libgsasl handle.

authentication_id: input array with authentication identity.

authorization_id: input array with authorization identity, or NULL.

realm: input array with realm of user, or NULL.

key: output array with key for authentication identity.

keylen: on input the maximum size of the key output array, on output contains the actual size of the key output array.

Type of callback function the application implements. It should retrieve the password for the indicated user and return GSASL_OK, or an error code such as GSASL_AUTHENTICATION_ERROR. The key must be encoded in UTF-8, but need not be normalized in any way.

If KEY is NULL, the function should only populate the KEYLEN output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

4.5 The CRAM-MD5 mechanism

The CRAM-MD5 mechanism uses username (authorization identity only) and password to authenticate users. Only a hashed password is transfered, which means that you cannot use normal policy deciding authentication systems such as PAM which do not support extraction of passwords. Two ways of validating the user is provided, either by having the SASL mechanism retrieve the raw password from the application and perform the validation internally, or by calling the application with the CRAM-MD5 challenge and response and let it decide. If both the validating and the retrieving callbacks are specified by the application, the validating one will be used.

While not documented in the original CRAM-MD5 specification, this implementation normalizes the username and the authorization identity using the Unicode 3.2 NFKC form according to the proposed update of CRAM-MD5.

This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authorization_id**) (`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_password**) (`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with password.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with password of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The password must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_retrieve***) (`Gsasl_session_ctx *` *ctx*, `char *` *authentication_id*, `char *` *authorization_id*, `char *` *realm*, `char *` *key*, `size_t *` *keylen*) Prototype

ctx: libgsasl handle.

authentication_id: input array with authentication identity.

authorization_id: input array with authorization identity, or NULL.

realm: input array with realm of user, or NULL.

key: output array with key for authentication identity.

keylen: on input the maximum size of the key output array, on output contains the actual size of the key output array.

Type of callback function the application implements. It should retrieve the password for the indicated user and return GSASL_OK, or an error code such as GSASL_AUTHENTICATION_ERROR. The key must be encoded in UTF-8, but need not be normalized in any way.

If KEY is NULL, the function should only populate the KEYLEN output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_cram_md5***) (`Gsasl_session_ctx *` *ctx*, `char *` *username*, `char *` *challenge*, `char *` *response*) Prototype

ctx: libgsasl handle.

username: input array with username.

challenge: input array with CRAM-MD5 challenge.

response: input array with CRAM-MD5 response.

Type of callback function the application implements. It should return GSASL_OK if and only if the validation of the provided credential was succesful. GSASL_AUTHENTICATION_ERROR is a good failure if authentication failed, but any available return code may be used.

4.6 The DIGEST-MD5 mechanism

The DIGEST-MD5 mechanism is based on the same cryptographic operation as CRAM-MD5 but supports more features, such as an authorization identity (proxy authentication) and cryptographic protection of data. Like CRAM-MD5, only a hashed password is transfered, which means that you cannot use e.g. PAM as a backend since it does not support extraction of passwords. Two ways of validating the user is provided, either by having the SASL mechanism retrieve the raw password from the application and perform the validation internally, or by having the SASL mechanism retrieve a hashed version of the secret. The advantage of using the latter method is that you do not need to store plain text user passwords on the server, but rather a one-way hash of the username, realm and password. Still, this one-way hash of the secret should be handled the same way as a clear text password. The advantage is that if someone steals the one-way hash she cannot immediately read users' password. If both the callbacks are specified by the application, the one which retrieve the secret hash will be used.

While not documented in the original DIGEST-MD5 specification, this implementation normalizes the username and the authentication identity using the Unicode 3.2 NFKC form according to the proposed update of DIGEST-MD5.

This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authentication_id**) (`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authentication identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authentiction identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authentication identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_authorization_id**) (`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_password**) (`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with password.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with password of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The password must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_service**) (`Gsasl_session_ctx *` *ctx*, `char *` *service*, `size_t *` *servicelen*, `char *` *hostname*, `size_t *` *hostnamelen*, `char *` *servicename*, `size_t *` *servicenamelen*) Prototype

ctx: libgsasl handle.

service: output array with name of service.

servicelen: on input the maximum size of the service output array, on output contains the actual size of the service output array.

hostname: output array with hostname of server.

hostnamelen: on input the maximum size of the hostname output array, on output contains the actual size of the hostname output array.

servicename: output array with generic name of server in case of replication (DIGEST-MD5 only).

servicenamelen: on input the maximum size of the servicename output array, on output contains the actual size of the servicename output array.

Type of callback function the application implements. It should retrieve the service (which should be a registered GSSAPI host based service name, such as "imap") on the server, hostname of server (usually canoncial DNS hostname) and optionally generic service name of server in case of replication (e.g. "mail.example.org" when the hostname is "mx42.example.org", see the RFC 2831 for more information). It should return GSASL_OK, or an error such as GSASL_AUTHENTICATION_ERROR if it fails.

If SERVICE, HOSTNAME or SERVICENAME is NULL, the function should only populate SERVICELEN, HOSTNAMELEN or SERVICENAMELEN with the output length of the respective field, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size. Furthermore, SERVICENAMELEN may also be NULL, indicating that the mechanism is not interested in this field.

`int` (***Gsasl_server_callback_retrieve**) (`Gsasl_session_ctx *` *ctx*, `char *` *authentication_id*, `char *` *authorization_id*, `char *` *realm*, `char *` *key*, `size_t *` *keylen*) Prototype

ctx: libgsasl handle.

authentication_id: input array with authentication identity.

authorization_id: input array with authorization identity, or NULL.

realm: input array with realm of user, or NULL.

key: output array with key for authentication identity.

keylen: on input the maximum size of the key output array, on output contains the actual size of the key output array.

Type of callback function the application implements. It should retrieve the password for the indicated user and return GSASL_OK, or an error code such as

GSASL_AUTHENTICATION_ERROR. The key must be encoded in UTF-8, but need not be normalized in any way.

If KEY is NULL, the function should only populate the KEYLEN output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_digest_md5**) (`Gsasl_session_ctx *` *ctx*, `char *` *username*, `char *` *realm*, `char *` *secrethash*) Prototype

ctx: libgsasl handle.

username: input array with authentication identity of user.

realm: input array with realm of user.

secrethash: output array that should contain hash of username, realm and password as described for the DIGEST-MD5 mechanism.

Type of callback function the application implements. It should retrieve the secret hash for the given user in given realm and return GSASL_OK, or an error such as GSASL_AUTHENTICATION_ERROR if it fails. The secrethash buffer is guaranteed to have size for the fixed length MD5 hash.

4.7 The NTLM mechanism

The NTLM mechanism uses username (authorization identity only) and password to authenticate users. Only the client side is implemented. This mechanism is only enabled in the client if you implement the callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

Note: Libntlm uses assert() in some places, it may thus crash your client if it is given bad input.

`int` (***Gsasl_client_callback_authorization_id**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_password**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with password.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with password of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The password must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

4.8 The SECURID mechanism

The SECURID mechanism uses authentication and authorization identity and a passcode from a hardware token to authenticate users. This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authentication_id**) Prototype

(`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*)

ctx: libgsasl handle.

out: output array with authentication identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authentiction identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authentication identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_authorization_id**) Prototype

(`Gsasl_session_ctx *` *ctx*`, char *` *out*`, size_t *` *outlen*)

ctx: libgsasl handle.

out: output array with authorization identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authorization identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authorization identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_passcode**) (`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Prototype

ctx: libgsasl handle.

out: output array with passcode.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with passcode of user and set the output array length, and return `GSASL_OK`, or fail with an error code.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_server_callback_validate**) (`Gsasl_session_ctx *` *ctx*, `char *` *authentication_id*, `char *` *authorization_id*, `char *` *passcode*, `char *` *pin*, `char *` *suggestpin*, `size_t *` *suggestpinlen*) Prototype

ctx: libgsasl handle.

authorization_id: input array with authorization identity.

authentication_id: input array with authentication identity.

passcode: input array with passcode.

pin: input array with new pin (this may be NULL).

suggestpin: output array with new suggested PIN.

suggestpinlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should return GSASL_OK if and only if the validation of the provided credential was succesful. GSASL_AUTHENTICATION_ERROR is a good failure if authentication failed, but any available return code may be used.

Two SECURID specific error codes also exists. The function can return GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE to request that the client generate a new passcode. It can also return GSASL_SECURID_SERVER_NEED_NEW_PIN to request that the client generate a new PIN. If the server wishes to suggest a new PIN it can populate the SUGGESTPIN field.

If SUGGESTPIN is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

4.9 The GSSAPI mechanism

The GSSAPI mechanism uses a framework similar to SASL for authenticating the user. While GSSAPI can be implemented using many techniques, libgsasl currently links with MIT's GSSAPI Kerberos 5 library and is limited to Kerberos 5 only. The GSSAPI client mechanism assumes the user acquired credentials (kerberos tickets) before it is invoked

(it will fail if this has not been done). The client need (via callbacks) the name of the service and the name of the user. The server needs the name of the service and a function that authorizes a user. This mechanism is only enabled in the client and server if you implement the respectively callbacks below and set them in the library (see Chapter 6 [Callback Functions], page 25).

`int` (***Gsasl_client_callback_authentication_id**) Prototype
(`Gsasl_session_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*)

ctx: libgsasl handle.

out: output array with authentication identity.

outlen: on input the maximum size of the output array, on output contains the actual size of the output array.

Type of callback function the application implements. It should populate the output array with authentiction identity of user and set the output array length, and return `GSASL_OK`, or fail with an error code. The authentication identity must be encoded in UTF-8, but need not be normalized in any way.

If OUT is NULL, the function should only populate the output length field with the length, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int` (***Gsasl_client_callback_service**) (`Gsasl_session_ctx *` *ctx*, Prototype
`char *` *service*, `size_t *` *servicelen*, `char *` *hostname*, `size_t *` *hostnamelen*, `char *` *servicename*, `size_t *` *servicenamelen*)

ctx: libgsasl handle.

service: output array with name of service.

servicelen: on input the maximum size of the service output array, on output contains the actual size of the service output array.

hostname: output array with hostname of server.

hostnamelen: on input the maximum size of the hostname output array, on output contains the actual size of the hostname output array.

servicename: output array with generic name of server in case of replication (DIGEST-MD5 only).

servicenamelen: on input the maximum size of the servicename output array, on output contains the actual size of the servicename output array.

Type of callback function the application implements. It should retrieve the service (which should be a registered GSSAPI host based service name, such as “imap”) on the server, hostname of server (usually canoncial DNS hostname) and optionally generic service name of server in case of replication (e.g. “mail.example.org” when the hostname is “mx42.example.org”, see the RFC 2831 for more information). It should return GSASL_OK, or an error such as GSASL_AUTHENTICATION_ERROR if it fails.

If SERVICE, HOSTNAME or SERVICENAME is NULL, the function should only populate SERVICELEN, HOSTNAMELEN or SERVICENAMELEN with the output length of the respective field, and return GSASL_OK. This usage may be used by the

caller to allocate the proper buffer size. Furthermore, SERVICENAMELEN may also be NULL, indicating that the mechanism is not interested in this field.

`int (*`**Gsasl_server_callback_service**`) (Gsasl_session_ctx * ctx, char * service, size_t * servicelen, char * hostname, size_t * hostnamelen)` Prototype

ctx: libgsasl handle.

service: output array with name of service.

servicelen: on input the maximum size of the service output array, on output contains the actual size of the service output array.

hostname: output array with hostname of server.

hostnamelen: on input the maximum size of the hostname output array, on output contains the actual size of the hostname output array.

Type of callback function the application implements. It should retrieve the service (which should be a registered GSSAPI host based service name, such as "imap") the server provides and hostname of server (usually canoncial DNS hostname). It should return GSASL_OK, or an error such as GSASL_AUTHENTICATION_ERROR if it fails.

If SERVICE or HOSTNAME is NULL, the function should only populate SERVICELEN or HOSTNAMELEN with the output length of the respective field, and return GSASL_OK. This usage may be used by the caller to allocate the proper buffer size.

`int (*`**Gsasl_server_callback_gssapi**`) (Gsasl_session_ctx * ctx, char * clientname, char * authentication_id)` Prototype

ctx: libgsasl handle.

clientname: input array with GSSAPI client name.

authentication_id: input array with authentication identity.

Type of callback function the application implements. It should return GSASL_OK if and only if the GSSAPI user is authorized to log on as the given authentication_id. GSASL_AUTHENTICATION_ERROR is a good failure if authentication failed, but any available return code may be used. This callback is usually implemented in the application as a call to krb5_kuserok(), such as:

```
int
callback_gssapi (Gsasl_session_ctx *ctx,
 char *clientname,
 char *authentication_id)
{
  int rc = GSASL_AUTHENTICATION_ERROR;

  krb5_principal p;
  krb5_context kcontext;

  krb5_init_context (&kcontext);

  if (krb5_parse_name (kcontext, clientname, &p) != 0)
```

```
    return -1;
  if (krb5_kuserok (kcontext, p, authentication_id))
    rc = GSASL_OK;
  krb5_free_principal (kcontext, p);

  return rc;
}
```

5 Global Functions

`int` **gsasl_init** (`Gsasl_ctx **` *ctx*) Function

ctx: pointer to libgsasl handle.

This functions initializes libgsasl. The handle pointed to by ctx is valid for use with other libgsasl functions iff this function is successful.

GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

`void` **gsasl_done** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

This function destroys a libgsasl handle. The handle must not be used with other libgsasl functions after this call.

`int` **gsasl_client_listmech** (`Gsasl_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Function

ctx: libgsasl handle.

out: output character array.

outlen: input maximum size of output character array, on output contains actual length of output array.

Write SASL names, separated by space, of mechanisms supported by the libgsasl client to the output array. To find out how large the output array must be, call this function with out=NULL.

Returns GSASL_OK if successful, or error code.

`int` **gsasl_server_listmech** (`Gsasl_ctx *` *ctx*, `char *` *out*, `size_t *` *outlen*) Function

ctx: libgsasl handle.

out: output character array.

outlen: input maximum size of output character array, on output contains actual length of output array.

Write SASL names, separated by space, of mechanisms supported by the libgsasl server to the output array. To find out how large the output array must be, call this function with out=NULL.

Returns GSASL_OK if successful, or error code.

`int` **gsasl_client_support_p** (`Gsasl_ctx *` *ctx*, `const char *` *name*) Function

ctx: libgsasl handle.

name: name of SASL mechanism.

Returns 1 if the libgsasl client supports the named mechanism, otherwise 0.

`int` **gsasl_server_support_p** (`Gsasl_ctx *` *ctx*, `const char *` *name*) Function

ctx: libgsasl handle.

name: name of SASL mechanism.

Returns 1 if the libgsasl server supports the named mechanism, otherwise 0.

`const char *` **gsasl_client_suggest_mechanism** (`Gsasl_ctx *` *ctx*, `const char *` *mechlist*) Function

ctx: libgsasl handle.

mechlist: input character array with SASL mechanism names, separated by invalid characters (e.g. SPC).

Returns name of "best" SASL mechanism supported by the libgsasl client which is present in the input string.

`const char *` **gsasl_server_suggest_mechanism** (`Gsasl_ctx *` *ctx*, `const char *` *mechlist*) Function

ctx: libgsasl handle.

mechlist: input character array with SASL mechanism names, separated by invalid characters (e.g. SPC).

Returns name of "best" SASL mechanism supported by the libgsasl server which is present in the input string.

6 Callback Functions

`Gsasl_ctx *` **gsasl_client_ctx_get** (`Gsasl_session_ctx *` *cctx*) Function

cctx: libgsasl client handle

Returns the libgsasl handle given a libgsasl client handle.

`Gsasl_ctx *` **gsasl_server_ctx_get** (`Gsasl_session_ctx *` *sctx*) Function

Returns the libgsasl handle given a libgsasl server handle.

`void` **gsasl_application_data_set** (`Gsasl_ctx *` *ctx*, `void *` *application_data*) Function

ctx: libgsasl handle.

application_data: opaque pointer to application specific data.

Store application specific data in the libgsasl handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

`void *` **gsasl_application_data_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Retrieve application specific data from libgsasl handle. The application data is set using gsasl_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Returns the application specific data, or NULL.

`void` **gsasl_client_application_data_set** (`Gsasl_session_ctx *` *cctx*, `void *` *application_data*) Function

application_data: opaque pointer to application specific data.

Store application specific data in the libgsasl client handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_client_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

`void *` **gsasl_client_application_data_get** (`Gsasl_session_ctx *` *cctx*) Function

Retrieve application specific data from libgsasl client handle. The application data is set using gsasl_client_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Returns the application specific data, or NULL.

`void` **gsasl_server_application_data_set** (`Gsasl_session_ctx *` *sctx*, `void *` *application_data*) Function

application_data: opaque pointer to application specific data.

Store application specific data in the libgsasl server handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_server_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

`void *` **gsasl_server_application_data_get** (`Gsasl_session_ctx *` *sctx*) Function

Retrieve application specific data from libgsasl server handle. The application data is set using gsasl_server_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Returns the application specific data, or NULL.

`void` **gsasl_client_callback_authentication_id_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_authentication_id` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the authentication identity. The function can be later retrieved using gsasl_client_callback_authentication_id_get().

`Gsasl_client_callback_authentication_id` **gsasl_client_callback_authentication_id_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_authentication_id_set().

`void` **gsasl_client_callback_authorization_id_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_authorization_id` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the authorization identity. The function can be later retrieved using gsasl_client_callback_authorization_id_get().

`Gsasl_client_callback_authorization_id` **gsasl_client_callback_authorization_id_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_authorization_id_set().

`void` **gsasl_client_callback_password_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_password` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the password. The function can be later retrieved using gsasl_client_callback_password_get().

`Gsasl_client_callback_password` Function
gsasl_client_callback_password_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_password_set().

`void` **gsasl_client_callback_passcode_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_client_callback_passcode` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the passcode. The function can be later retrieved using gsasl_client_callback_passcode_get().

`Gsasl_client_callback_passcode` Function
gsasl_client_callback_passcode_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_passcode_set().

`void` **gsasl_client_callback_pin_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_client_callback_pin` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to chose a new pin, possibly suggested by the server, for the SECURID mechanism. This is not normally invoked, but only when the server requests it. The function can be later retrieved using gsasl_client_callback_pin_get().

`Gsasl_client_callback_pin` **gsasl_client_callback_pin_get** Function
(`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_pin_set().

`void` **gsasl_client_callback_service_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_client_callback_service` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the name of the service. The service buffer should be a registered GSSAPI host-based service name, hostname the name of the server. Servicename is used by DIGEST-MD5 and should be the name of generic server in case of a replicated service. The function can be later retrieved using gsasl_client_callback_service_get().

`Gsasl_client_callback_service` Function
gsasl_client_callback_service_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_service_set().

`void` **gsasl_client_callback_anonymous_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_anonymous` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to set the anonymous token, which usually is the users email address. The function can be later retrieved using gsasl_client_callback_anonymous_get().

`Gsasl_client_callback_anonymous` **gsasl_client_callback_anonymous_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_anonymous_set().

`void` **gsasl_client_callback_qop_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_qop` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to determine the qop to use after looking at what the server offered. The function can be later retrieved using gsasl_client_callback_qop_get().

`Gsasl_client_callback_qop` **gsasl_client_callback_qop_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_qop_set().

`void` **gsasl_client_callback_maxbuf_set** (`Gsasl_ctx *` *ctx*, `Gsasl_client_callback_maxbuf` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the client to inform the server of the largest buffer the client is able to receive when using the DIGEST-MD5 "auth-int" or "auth-conf" Quality of Protection (qop). If this directive is missing, the default value 65536 will be assumed. The function can be later retrieved using gsasl_client_callback_maxbuf_get().

`Gsasl_client_callback_maxbuf` **gsasl_client_callback_maxbuf_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_client_callback_maxbuf_set().

`void` **gsasl_server_callback_validate_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_validate` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for deciding if user is authenticated using authentication identity, authorization identity and password. The function can be later retrieved using gsasl_server_callback_validate_get().

`Gsasl_server_callback_validate` Function

gsasl_server_callback_validate_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_validate_set().

`void` **gsasl_server_callback_retrieve_set** (`Gsasl_ctx *` *ctx*, Function

`Gsasl_server_callback_retrieve` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for deciding if user is authenticated using authentication identity, authorization identity and password. The function can be later retrieved using gsasl_server_callback_retrieve_get().

`Gsasl_server_callback_retrieve` Function

gsasl_server_callback_retrieve_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_retrieve_set().

`void` **gsasl_server_callback_cram_md5_set** (`Gsasl_ctx *` *ctx*, Function

`Gsasl_server_callback_cram_md5` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for deciding if user is authenticated using CRAM-MD5 challenge and response. The function can be later retrieved using gsasl_server_callback_cram_md5_get().

`Gsasl_server_callback_cram_md5` Function

gsasl_server_callback_cram_md5_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_cram_md5_set().

`void` **gsasl_server_callback_digest_md5_set** (`Gsasl_ctx *` *ctx*, Function

`Gsasl_server_callback_digest_md5` *cb*)

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for retrieving the secret hash of the username, realm and password for use in the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_digest_md5_get().

`Gsasl_server_callback_digest_md5` Function
gsasl_server_callback_digest_md5_get (`Gsasl_ctx *` *ctx*)
ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_digest_md5_set().

`void` **gsasl_server_callback_external_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_server_callback_external` *cb*)
ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for deciding if user is authenticated out of band. The function can be later retrieved using gsasl_server_callback_external_get().

`Gsasl_server_callback_external` Function
gsasl_server_callback_external_get (`Gsasl_ctx *` *ctx*)
ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_external_set().

`void` **gsasl_server_callback_anonymous_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_server_callback_anonymous` *cb*)
ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for deciding if user is permitted anonymous access. The function can be later retrieved using gsasl_server_callback_anonymous_get().

`Gsasl_server_callback_anonymous` Function
gsasl_server_callback_anonymous_get (`Gsasl_ctx *` *ctx*)
ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_anonymous_set().

`void` **gsasl_server_callback_realm_set** (`Gsasl_ctx *` *ctx*, Function
`Gsasl_server_callback_realm` *cb*)
ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server to know which realm it serves. The realm is used by the user to determine which username and password to use. The function can be later retrieved using gsasl_server_callback_realm_get().

`Gsasl_server_callback_realm` **gsasl_server_callback_realm_get** Function
(`Gsasl_ctx *` *ctx*)
ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_realm_set().

`void` **gsasl_server_callback_qop_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_qop` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server to know which quality of protection it accepts. The quality of protection eventually used is selected by the client though. It is currently used by the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_qop_get().

`Gsasl_server_callback_qop` **gsasl_server_callback_qop_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_qop_set().

`void` **gsasl_server_callback_maxbuf_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_maxbuf` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server to inform the client of the largest buffer the server is able to receive when using the DIGEST-MD5 "auth-int" or "auth-conf" Quality of Protection (qop). If this directive is missing, the default value 65536 will be assumed. The function can be later retrieved using gsasl_server_callback_maxbuf_get().

`Gsasl_server_callback_maxbuf` **gsasl_server_callback_maxbuf_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_maxbuf_set().

`void` **gsasl_server_callback_cipher_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_cipher` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server to inform the client of the cipher suites supported. The DES and 3DES ciphers must be supported for interoperability. It is currently used by the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_cipher_get().

`Gsasl_server_callback_cipher` **gsasl_server_callback_cipher_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_cipher_set().

`void` **gsasl_server_callback_securid_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_securid` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for validating a user via the SECURID mechanism. The function should return GSASL_OK if user authenticated successfully, GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE if it wants another passcode, GSASL_SECURID_SERVER_NEED_NEW_PIN if it wants a PIN change, or an error. When (and only when) GSASL_SECURID_SERVER_NEED_NEW_PIN is returned, suggestpin can be populated with a PIN code the server suggests, and suggestpinlen set to the length of the PIN. The function can be later retrieved using gsasl_server_callback_securid_get().

`Gsasl_server_callback_securid` **gsasl_server_callback_securid_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_securid_set().

`void` **gsasl_server_callback_gssapi_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_gssapi` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server for checking if a GSSAPI user is authorized for username (by, e.g., calling krb5_userok()). The function should return GSASL_OK if the user should be permitted access, or an error code such as GSASL_AUTHENTICATION_ERROR on failure. The function can be later retrieved using gsasl_server_callback_gssapi_get().

`Gsasl_server_callback_gssapi` **gsasl_server_callback_gssapi_get** (`Gsasl_ctx *` *ctx*) Function

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_gssapi_set().

`void` **gsasl_server_callback_service_set** (`Gsasl_ctx *` *ctx*, `Gsasl_server_callback_service` *cb*) Function

ctx: libgsasl handle.

cb: callback function

Specify the callback function to use in the server to set the name of the service. The service buffer should be a registered GSSAPI host-based service name, hostname the name of the server. The function can be later retrieved using gsasl_server_callback_service_get().

`Gsasl_server_callback_service` Function

gsasl_server_callback_service_get (`Gsasl_ctx *` *ctx*)

ctx: libgsasl handle.

Return the callback earlier set by calling gsasl_server_callback_service_set().

7 Session Functions

`int` **gsasl_client_start** (`Gsasl_ctx *` *ctx*, `const char *` *mech*, `Gsasl_session_ctx **` *xctx*) Function

ctx: libgsasl handle.

mech: name of SASL mechanism.

xctx: pointer to client handle.

This functions initiates a client SASL authentication. This function must be called before any other gsasl_client_*() function is called.

Returns GSASL_OK if successful, or error code.

`int` **gsasl_server_start** (`Gsasl_ctx *` *ctx*, `const char *` *mech*, `Gsasl_session_ctx **` *xctx*) Function

ctx: libgsasl handle.

mech: name of SASL mechanism.

xctx: pointer to server handle.

This functions initiates a server SASL authentication. This function must be called before any other gsasl_server_*() function is called.

Returns GSASL_OK if successful, or error code.

`int` **gsasl_client_step** (`Gsasl_session_ctx *` *xctx*, `const char *` *input*, `size_t` *input_len*, `char *` *output*, `size_t *` *output_len*) Function

xctx: libgsasl client handle.

input: input byte array.

input_len: size of input byte array.

output: output byte array.

output_len: size of output byte array.

Perform one step of SASL authentication in client. This reads data from server (specified with input and input_len), processes it (potentially invoking callbacks to the application), and writes data to server (into variables output and output_len).

The contents of the output buffer is unspecified if this functions returns anything other than GSASL_NEEDS_MORE.

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

`int` **gsasl_server_step** (`Gsasl_session_ctx *` *xctx*, `const char *` *input*, `size_t` *input_len*, `char *` *output*, `size_t *` *output_len*) Function

xctx: libgsasl server handle.

input: input byte array.

input_len: size of input byte array.

output: output byte array.

output_len: size of output byte array.

Perform one step of SASL authentication in server. This reads data from client (specified with input and input_len), processes it (potentially invoking callbacks to the application), and writes data to client (into variables output and output_len).

The contents of the output buffer is unspecified if this functions returns anything other than GSASL_NEEDS_MORE.

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

`int` **gsasl_client_step_base64** (`Gsasl_session_ctx *` *xctx*, `const char *` *b64input*, `char *` *b64output*, `size_t` *b64output_len*) Function

xctx: libgsasl client handle.

b64input: input base64 encoded byte array.

b64output: output base64 encoded byte array.

b64output_len: size of output base64 encoded byte array.

This is a simple wrapper around gsasl_client_step() that base64 decodes the input and base64 encodes the output.

See gsasl_client_step().

`int` **gsasl_server_step_base64** (`Gsasl_session_ctx *` *xctx*, `const char *` *b64input*, `char *` *b64output*, `size_t` *b64output_len*) Function

xctx: libgsasl server handle.

b64input: input base64 encoded byte array.

b64output: output base64 encoded byte array.

b64output_len: size of output base64 encoded byte array.

This is a simple wrapper around gsasl_server_step() that base64 decodes the input and base64 encodes the output.

See gsasl_server_step().

`void` **gsasl_client_finish** (`Gsasl_session_ctx *` *xctx*) Function

xctx: libgsasl client handle.

Destroy a libgsasl client handle. The handle must not be used with other libgsasl functions after this call.

`void` **gsasl_server_finish** (`Gsasl_session_ctx *` *xctx*) Function

xctx: libgsasl server handle.

Destroy a libgsasl server handle. The handle must not be used with other libgsasl functions after this call.

`int` **gsasl_encode** (`Gsasl_session_ctx *` *xctx*, `const char *` *input*, `size_t` *input_len*, `char *` *output*, `size_t *` *output_len*) Function

xctx: libgsasl session handle.

input: input byte array.

input_len: size of input byte array.

output: output byte array.

output_len: size of output byte array.

Encode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

Returns GSASL_OK if encoding was successful, otherwise an error code.

`int` **gsasl_decode** (`Gsasl_session_ctx *` *xctx*, `const char *` *input*, `size_t` *input_len*, `char *` *output*, `size_t *` *output_len*) Function

xctx: libgsasl session handle.

input: input byte array.

input_len: size of input byte array.

output: output byte array.

output_len: size of output byte array.

Decode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

Returns GSASL_OK if encoding was successful, otherwise an error code.

8 Utilities

`int` **gsasl_base64_encode** (`unsigned char const *` *src*, `size_t` *srclength*, `char *` *target*, `size_t` *targsize*) Function

src: input byte array

srclength: size of input byte array

target: output byte array

targsize: size of output byte array

Encode data as base64. Converts characters, three at a time, starting at src into four base64 characters in the target area until the entire input buffer is encoded.

Returns the number of data bytes stored at the target, or -1 on error.

`int` **gsasl_base64_decode** (`char const *` *src*, `unsigned char *` *target*, `size_t` *targsize*) Function

src: input byte array

target: output byte array

targsize: size of output byte array

Decode Base64 data. Skips all whitespace anywhere. Converts characters, four at a time, starting at (or after) src from Base64 numbers into three 8 bit bytes in the target area.

Returns the number of data bytes stored at the target, or -1 on error.

`void` **gsasl_hexdump** (`FILE *` *fh*, `const char *` *buffer*, `size_t` *len*) Function

fh: file handle

buffer: input byte array

len: size of input byte array

Print a byte array to given file handle, mostly for debugging purposes.

`int` **gsasl_md5pwd_get_password** (`const char *` *filename*, `const char *` *username*, `char *` *key*, `size_t *` *keylen*) Function

filename: filename of file containing passwords.

username: username string.

key: output character array.

keylen: input maximum size of output character array, on output contains actual length of output array.

Retrieve password for user from specified file. To find out how large the output array must be, call this function with out=NULL.

The file should be on the UoW "MD5 Based Authentication" format, which means it is in text format with comments denoted by # first on the line, with user entries looking as username\tpassword. This function removes \r and \n at the end of lines before processing.

Return GSASL_OK if output buffer contains the password, GSASL_AUTHENTICATION_ERROR if the user could not be found, or other error code.

9 Error Handling

Most functions in 'Libgsasl' are returning an error if they fail. For this reason, the application should always catch the error condition and take appropriate measures, for example by releasing the resources and passing the error up to the caller, or by displaying a descriptive message to the user and cancelling the operation.

Some error values do not indicate a system error or an error in the operation, but the result of an operation that failed properly.

9.1 Error values

Errors are returned as an `int`. Except for the OK case an application should always use the constants instead of their numeric value. Applications are encouraged to use the constants even for OK as it improves readability. Possible values are:

`GSASL_OK` This value indicates success. The value of this error is guaranteed to always be `0`.

9.2 Error strings

`const char *` **gsasl_strerror** (`int` *err*) Function

err: libgsasl error code

Returns a pointer to a statically allocated string containing a description of the error with the error value *err*. This string can be used to output a diagnostic message to the user.

10 Examples

This chapter contains example code which illustrate how 'Libgsasl' can be used when writing your own application.

10.1 Example 1

This is the minimal program which uses 'Libgsasl' (including internationalization features) without doing anything.

```
#include <locale.h>
#include <stdio.h>
#include <gsasl.h>

/* Build using the following command:
 * gcc -o foo foo.c `libgsasl-config --cflags --libs`
 */

int
main (int argc, char *argv[])
{
  Gsasl_ctx *ctx;
  int res;

  setlocale (LC_ALL, "");

  if (gsasl_check_version(GSASL_VERSION) == NULL)
    {
      fprintf(stderr, "Libgsasl is %s expected %s\n",
      gsasl_check_version(NULL), GSASL_VERSION);
      return 1;
    }

  res = gsasl_init (&ctx);
  if (res != GSASL_OK)
    {
      fprintf(stderr, "Cannot initialize libgsasl: %s\n",
      gsasl_strerror(res));
      return 1;
    }

  /* Do things here ... */

  gsasl_done(ctx);

  return 0;
}
```

11 Acknowledgements

Simon Josefsson created the library autumn 2002 when he really should have been studying mathematics.

The makefiles, manuals, etc borrowed much from Libgcrypt written by Werner Koch.

Cryptographic functions for some SASL mechanisms uses Libgcrypt written by Werner Koch. The NTLM mechanism uses Libntlm written by Grant Edwards and uses code from Samba written by Andrew Tridgell. The GSSAPI mechanism uses Kerberos written by MIT.

Appendix A Copying This Manual

A.1 GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other written 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 that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

   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. (For example, 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.

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 "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human modification. Opaque formats include PostScript, PDF, 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 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.

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 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 publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. 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 less than five).

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", and 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. In any section entitled "Acknowledgments" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments 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 as "Endorsements" or to conflict in title with any Invariant Section.

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.

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 "Acknowledgments", 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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

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.

A.1.1 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 1.1
> or any later version published by the Free Software Foundation;
> with the Invariant Sections being *list their titles*, with the
> Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.
> A copy of the license is included in the section entitled ‘‘GNU
> Free Documentation License’’.

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being *list*"; likewise for Back-Cover Texts.

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.

Appendix B GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

B.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, 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 and use pieces of it in new free programs; and that you are informed that you can do these things.

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

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program

by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does *Less* to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

B.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, 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 library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete 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 distribute a copy of this License along with the Library.

   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 Library or any portion of it, thus forming a work based on the Library, 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. The modified work must itself be a software library.
   b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
   c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
   d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, 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 Library, 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 Library.

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

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you 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.

   If distribution of 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 satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
   b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
   c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
   d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
   e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

   For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be 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.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library 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.

9. 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 Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library 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 with this License.

11. 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 Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library 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 Library.

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.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library 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.
13. The Free Software Foundation may publish revised and/or new versions of the Lesser 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 Library 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 Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.
14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, 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

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. 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 LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

B.3 How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. 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 library's name and an idea of what it does.*
> Copyright (C) *year name of author*
>
> This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
>
> This library 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 Lesser General Public License for more details.
>
> You should have received a copy of the GNU Lesser General Public License along with this library; 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.

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

> Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
>
> *signature of Ty Coon*, 1 April 1990
> Ty Coon, President of Vice

That's all there is to it!

Appendix C GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

C.1 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.

C.2 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

C.3 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 a brief idea of what it does.*
> Copyright (C) *yyyy 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) 19*yy 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.

Concept Index

Function and Data Index

(

G

Short Contents

Table of Contents