LCDproc Developer’s Guide

The Hitchhiker’s Guide to LCDproc 0.5

Guillaume Filion

gfk@logidac.com

LCDproc Developer's Guide: The Hitchhiker's Guide to LCDproc 0.5

by Guillaume Filion

This document is a guide to LCDproc written for developers. It covers LCDproc 0.5. Users should read the user guide.

Table of Contents

Chapter 1. Introduction

1.1. About this Document

This document is meant to be a reference for LCDproc developers. It tries to indicate you where to find the relevant information about LCDproc's inner workings.

> **Note:** Please note that this document is still "under construction". If you run into any trouble feel free to write to the LCDproc mailing list. See http://lcdproc.org/mail.php3 for details on how to subscribe to the list.
>
> Therefore you might want to have a look at http://lcdproc.sf.net/docs/, to get the latest version of this document, unless you want to generate it yourself from the docbook files in the CVS).

This document was written for LCDproc 0.5.

In several other places e-mails and other documents have been included in this document. The authors of those are listed below every such document.

Chapter 2. Shared files

2.1. Introduction

Here we provide functions that should be used by all parts of the program.

2.2. report.h : Debugging and reporting

To enable the debug() function on all of the software, just type: ./configure --enable-debug and recompile with ’make’.

To enable the debug() function only in specific files: 1) Configure without enabling debug (that is without --enable-debug) 2) Edit the source file that you want to debug and put the following line at the top, before the #include "report.h" line: #define DEBUG 3) Then recompile with ’make’ This way, the global DEBUG macro is off but is locally enabled in certains parts of the software.

The reporting levels have the following meaning.

Reporting Levels

0 RPT_CRIT

Critical conditions: the program stops right after this. Only use this if the program is actually exited from the current function.

1 RPT_ERR

Error conditions: serious problem, program continues. Use this just before you return -1 from a function.

2 RPT_WARNING

Warning conditions: Something that the user should fix, but the program can continue without a real problem. Ex: Protocol errors from a client.

3 RPT_NOTICE

Major event in the program: (un)loading of driver, client (dis)connect.

4 RPT_INFO

Minor event in the program: the activation of a setting, details of a loaded driver, a key reservation, a keypress, a screen switch.

5 RPT_DEBUG

Insignificant event: What function has been called, what subpart of a function is being executed, what was received and sent over the socket, etc.

Levels 4 (maybe) and 5 (certainly) should be reported using the debug function. The code that this function generates will not be in the executable when compiled without debugging. This way memory and CPU cycles are saved.

report.h file defines 3 functions for debugging and reporting:

2.2.1. Sets reporting level and message destination

```
int set_reporting(char * application_name, int new_level, int new_dest);
```

Returns the content of the byte.

2.2.2. Report the message to the selected destination if important enough

```
void report(const int level, const char *format, ...);
```

Returns nothing (void).

The format parameter is the same as the one used by printf.

2.2.3. Send debugging information if important enough

Consider the debug function to be exactly the same as the report function. The only difference is that it is only compiled in if DEBUG is defined.

2.3. LL.h : Linked Lists (Doubly-Linked Lists)

2.3.1. Creating a list

To create a list, do the following:

```
LinkedList *list;
list = LL_new();
if(!list) handle_an_error();
```

The list can hold any type of data. You will need to typecast your datatype to a "void *", though. So, to add something to the list, the following would be a good way to start:

```
typedef struct my_data {
  char string[16];
  int number;
} my_data;

my_data *thingie;

for(something to something else) {
  thingie = malloc(sizeof(my_data));
  LL_AddNode(list, (void *)thingie);  // typecast it to a "void *"
}
```

For errors, the general convention is that "0" means success, and a negative number means failure. Check LL.c to be sure, though.

2.3.2. Changing data

To change the data, try this:

```
thingie = (my_data *)LL_Get(list);  // typecast it back to "my_data"
thingie->number = another_number;
```

You don’t need to "Put" the data back, but it doesn’t hurt anything.

```
LL_Put(list, (void *)thingie);
```

However, if you want to point the node’s data somewhere else, you’ll need to get the current data first, keep track of it, then set the data to a new location:

```
my_data * old_thingie, new_thingie;

old_thingie = (my_data *)LL_Get(list);
```

```
LL_Put(list, (void *)new_thingie);

// Now, do something with old_thingie.  (maybe, free it?)
```

Or, you could just delete the node entirely and then add a new one:

```
my_data * thingie;

thingie = (my_data *)LL_DeleteNode(list);
free(thingie);

thingie->number = 666;

LL_InsertNode(list, (void *)thingie);
```

2.3.3. Iterations throught the list

To iterate on each list item, try this:

```
LL_Rewind(list);
do {
  my_data = (my_data *)LL_Get(list);
  /* ... do something to it ... */
} while(LL_Next(list) == 0);
```

2.3.4. Using the list as a stack or a queue

You can also treat the list like a stack, or a queue. Just use the following functions:

```
LL_Push()      // Regular stack stuff: add, remove, peek, rotate
LL_Pop()
LL_Top()
LL_Roll()

LL_Shift()     // Other end of the stack (like in perl)
LL_Unshift()
LL_Look()
LL_UnRoll()

LL_Enqueue()   // Standard queue operations
LL_Dequeue()
```

There are also other goodies, like sorting and searching.

2.3.5. Future

Array-like operations will come later, to allow numerical indexing:

```
LL_nGet(list, 3);
LL_nSwap(list, 6, 13);
LL_nPut(list, -4, data);   // Puts item at 4th place from the end..
```

More ideas for later:

```
LL_MoveNode(list, amount);  // Slides a node to another spot in the list
-- LL_MoveNode(list, -1); // moves a node back one toward the head
```

That’s about it, for now... Be sure to free the list when you’re done!

See LL.c for more detailed descriptions of these functions.

Chapter 3. The LCDproc client language

3.1. Introduction

The LCDproc clients, for example lcdproc, connect over the network to LCDd. In their communication they use a protocol, often refered to as the "widget language". In this chapter the widget language will be discussed.

3.2. Opening a session

The essence of talking to LCDd is quite simple. First you will need to connect to the LCDproc port (usually 13666) on the correct IP address (by default localhost). Once you have established the connection you should say "hello", to let LCDd know you are a good guy. It will respond by telling some LCDproc data, like version and screen width and height. Now your session is open and you can start sending 'real' commands.

LCDd can send a number of strings itself. As a response to your commands, it will usually send a "success" string, or a string starting with "huh" in case of any error. See further below for other strings sent by LCDd.

You can test all these commands by opening a TCP/IP connection manually, like with:

```
telnet localhost 13666
```

This way, you can check how the various commands work. It's in this case best to have no other clients. If you do have other clients, you will receive "listen" and "ignore" messages that will disturb your typing.

3.3. The various command

The commands and their parameters are listed below, along with the responses you can expect. If you need a space or a special char in a string, you should quote the string with double quotes. If you need to use a double quote, escape it with a backslash.

The LCDproc commands

hello

Opens the session with the LCDd server program. This command is required before other commands can be issued. The response will be a string in the format:

```
connect <name> <value> <name> <value> ...
```

Every name will be followed by a value. The client should read all parameters it needs and store their values. The following parameters are in use:

hello response parameters

LCDproc

Indicates the version number of LCDd.

protocol

Indicates the widget language version number. This number is only changed when the language of a newer version has become incompatible with the previous version.

wid

Tells the client the width of the attached display device.

hgt

Tells the client the height of the attached display device.

cellwid

How many pixels is a character wide (space between character cells not included)

cellhgt

How many pixels is a character high (space between character cells not included)

lcd

This word is NOT followed by a value ! Hey do we really need this word in the response string ?

client_set <attributes>

Sets attributes for the current client. The current client is the one from the connection that you send this command on, in other words: yourself.

client_set attributes

-name <name>

Sets the client’s name as visible to a user.

-heartbeat on|off|open

Sets the client’s heartbeat setting. This setting overrides the screen’s setting, so you can enable the heartbeat for all your screens at once. If "open", which is the default, the screen’s setting will be used.

-backlight on|off|toggle|open|blink|flash

Sets the client's backlight setting. This setting overrides the screen's setting, so you can enable the backlight for all your screens at once. If "open", which is the default, the screen's setting will be used. See screen_set attribute for details on the backlight modes.

screen_add <new_screen_id>

Adds a screen to be displayed. The screen will be identified by <new_screen_id>. Later you will need this id to add widgets to this screen.

screen_del <screen_id>

Removes the given screen.

screen_set <screen_id> <attributes>

Sets attributes for the given screen. The following attributes exist:

screen_set attributes

-name <name>

Sets the screen's name as visible to a user.

-wid <int>
-hgt <int>

Sets the size of the screen in characters. If unset, the full display size is assumed.

-priority <pri-class>

priority: The following priority classes exist: hidden (screen will never be visible), background (only visible when no normal info screens exist), info (normal info screen, default priority), foreground (an active client), alert (screen has an important message for you), input (the client is doing interactive input). LCDd will only show screens with the highest priority at that moment. So when there are 3 info screens and 1 foreground screen, only the foreground screen will be visible. Only background, info and foreground screens will rotate. Higher classes do not rotate because their function is not suitable for rotation.

-heartbeat on|off|open
-backlight on|off|toggle|open|blink|flash

If "open" (which is default), the state will be determined by the client's setting. "blink" is a moderately striking backlight variation, "flash" is VERY strinking.

-duration <seconds*8>

A screen will be visible for this amount of time every rotation. The value is in eights of a second.

-timeout <seconds*8>

After the screen has been visible for a total of this amount of time, it will be deleted. The value is in eights of a second. Currently the client will not be informed of the deletion (TODO?).

-cursor on|off|under|block

If on, a cursor will be visible. Depending on your hardware, this will be a hardware or software cursor. The specified cursor shape (block or under) might not be available in which case an other cursor shape will be used instead.

-cursor_x <int>
-cursor_y <int>

Coordinates are always 1-based. So top-left is (1,1).

widget_add <screen_id> <new_widget_id> <widgettype> [-in <frame_id>]

Adds a widget to the given screen. The <widgetid> sets the new identifier for this widget. The following widget types exist:

widget types

string

A simple text.

title

A title bar for above the screen.

hbar

A horizontal bar.

vbar

A vertical bar.

icon

A predefined or client-defined icon.

scroller

A variation of the string type that scrolls the text horizontally or vertically.

frame

A frame with that can contain widgets itself. In fact a frame displays an other screen in it.

num

A big number. They have a size of 3x4 characters. The special number 10 is a colon, that you can use for a clock. This character is 1x4.

widget_del <screen_id> <widget_id>

Deletes the given widget from the screen.

widget_set <screen_id> <widget_id> <widget specific parameters>

Sets parameters for a widget. Because not all widgets are created equal, the various widget types require different attributes.

widget_set required parameters per widget type

string

<x> <y> <text>

title

<text>

hbar
vbar

<x> <y> <length>

icon

<x> <y> <iconname>

scroller

<left> <top> <right> <bottom> <direction> <speed> <text>

direction can be "h", "m" or "v".

speed is the number of movements per rendering stroke (8 times/second).

frame

<left> <top> <right> <bottom> <width> <height> <direction> <speed>

direction can be "h" or "v".

speed is the number of movements per rendering stroke (8 times/second).

num

<x> <int>

x is the normal character x coordinate on the display.

int is the number to display, 0 to 9. Number 10 is a special number that will place a colon.

client_add_key [-excl|-shared] {<key>}+

Tells the server that the current client wants to make use of the given key(s). If you reserve the key(s) in shared mode, other clients can still reserve these keys too. If you reserve the key(s) in exclusive mode no other client can reserve them again. Key(s) reserved in shared mode will only be returned when a screen of the current client is active. These keys can be used for interaction with a visible screen (default). Key(s) reserved in exclusive mode will be returned regardless of which screen is active. They can be used to trigger a special feature or to make a screen come to foreground. Note that you cannot reserve a key in exclusive mode when an other client has reserved it in shared mode.

client_del_key {<key>}+

Ends the reservation of the given key(s).

menu_add_item <menu_id> <new_item_id> <type>

Adds a new menuitem to a menu. The main menu of a client, will be created automatically as soon as the client adds an item. This main menu has an empty id ("") and the name is identical to the name of the client.

menu item types

action

This item should trigger an action. It consists of simple text.

checkbox

Consists of a text and a status indicator. The status can be on (Y), off (N) or gray (o).

ring

Consists of a text and a status indicator. The status can be one of the strings specified for the item.

slider

Is visible as a text. When selected, a screen comes up that shows a slider. You can set the slider using the cursor keys. When Enter is pressed, the menu returns.

numeric

Allows the user to input an integer value. Is visible as a text. When selected, a screen comes up that shows the current numeric value, that you can edit with the cursor keys and Enter. The number is ended by selecting a ’null’ input digit. After that the menu returns.

alpha

Is visible as a text. When selected, a screen comes up that shows the current string value, that you can edit with the cursor keys and Enter. The string is ended by selecting a ’null’ input character. After that the menu returns.

ip

Allows the user to input an ip number (v4 or v6). When selected, a screen comes up that shows an ip number that can be edited - digit by digit - via left/right (switch digit) and up/down keys (increase/decrease).

menu

This is a submenu. It is visible as a text, with an appended ’>’. When selected, the submenu becomes the active menu.

menu_del_item <menu_id> <item_id>

Removes a menuitem <item_id> from menu <menu_id>. The menu named "" is the client’s main menu.

menu_set_item <menu_id> <item_id> <item_specific_options>

Sets parameters for the menuitem(s). Each item type knows different parameters.

options for the various menu items

for all item types

-text <string>

The visible text of the item.

action

-menu_result none|close|quit (none)

Sets what to do with the menu when this action is selected: none: the menu stays as it is; close: the menu closes and returns to a higher level; quit: quits the menu completely so you can foreground your app.

checkbox

-value <value>

Set the value of the item. Can be off, on or gray.

-allow_gray false|true (false)

Sets if a grayed checkbox is allowed.

ring

-value <int> (0)

Sets the index in the stringlist that is currently selected.

-strings <string> (empty)

> This single string should contain the strings that can be selected. They should be tab-separated (\t).

slider

-value <int> (0)

> Sets its current value.

-mintext <string> ("")
-maxtext <string> ("")

> The texts at the left and right side of the slider.

-minvalue <int> (0)
-maxvalue <int> (100)

> The minimum and maximum values of the slider.

-stepsize <int> (1)

> The stepsize of the slider. If you use 0, you can control the movement completely from your client.

numeric

-value <int> (0)

> Sets its current value.

-minvalue <int> (0)
-maxvalue <int> (100)

> The minimum and maximum values that are allowed. If one of them is negative, the user will be able to enter negative numbers too.
>
> TODO: floats!

alpha

-value <string> ("")

> Sets its current value.

-password_char <string> ("")

If used, instead of the typed characters, this character will be visible.

-minlength <int> (0)
-maxlength <int> (10)

Sets the minimum and maximum allowed lengths.

-allow_caps false|true (true)
-allow_noncaps false|true (false)
-allow_numbers false|true (false)

(Dis)allow these groups of characters.

-allowed_extra <string> ("")

The chars in this string are also allowed.

ip

-value <string> ("192.168.1.245")

Set the value of the item, e.g. "192.168.1.245" (v4) or ":::ffff:ffff:ffff:ffff:ffff" (v6).

-v6 false|true (false)

Changes IP version from default v4.

menu

This is a submenu. It is visible as a text, with an appended '>'. When selected, the submenu becomes the active menu.

menuscreen_goto <menu_id>

Changes current menu to <menu_id>. Depending on the configure option --enable-permissive-menuscreen-goto the client may switch to any (if enabled) or his menus only (if not enabled).

backlight on|off|toggle|blink|flash

Set's the client's backlight state.

output on|off|<int>

Sets the general purpose output on some display modules to this value. Use "on" to set all outputs to high state, and "off" to set all to low state. The meaning of the integer value is dependent on your specific device, usually it is a bitpattern describing the state of each output line.

noop

This command does nothing and is always successful. Can be useful to be sent at regular intervals to make sure your connection is still alive.

3.4. LCDd messages

LCDd can send messages back to the client. These messages can be directly related to the last command, or generated for some other reason. Because messages can be generated at any moment, the client should read from the connection at regular intervals. A very simple client could simply ignore all received messages. Not reading the messages will cause trouble !

success

> This is the reponse to a command in case everything went ok.

huh? <error-description>

> This is the reponse to a command in case something has gone wrong. The description is not meant to be parsed, it’s only meant for the programmer of the client. It might be that your command has only been partially executed, for example if you try to reserve 3 keys, and one fails. Your client might need to undo its actions completely.

listen <screen_id>
ignore <screen_id>

> The given screen is now visible on the display (listen) or it is not visible anymore on the display (ignore).

key <key>

> This message will be sent if there was a keypress that should be delivered to the current client.

menuevent <eventtype> <id> [<value>]

> The user did something with a client supplied menu. The type of action can be:
>
> select (action)
>
> > The item was activated.
>
> update (checkbox, ring, numeric, alpha)
>
> > The item was modified by the user, so LCDd sends an updated value.
>
> plus (slider)
> minus (slider)
>
> > The slider was moved to left (minus) or right (plus), so LCDd sends an updated value.
>
> enter
>
> > This item has been entered, which means it is currently active on the screen. The client could now for example update the value of the item. If it is a menu, it may be needed to update the values of the items in it too, because they may be visible too.

leave

  This item has been left, so it is currenly not the (main) active item anymore.

Multiple messages may be generated by one action of the user.

Chapter 4. Making a LCDproc driver

4.1. Introduction

LCDproc is meant to be modular, it is relatively easy to add new input and output drivers to LCDproc. Actually, there are a few things that you can do to make your life easier, they are listed here.

This chapter will explain you the major steps and few gotchas of adding your own driver to LCDproc. Enjoy!

4.2. Shared files specific for drivers

Driving an LCD display is not easy; you need to address ports, to send bytes in a certain order, to respect timing, and unfortunaly no two operating system let you do this in the same way. But don't dispair! There's hope! Someone in a galaxy far far away, has allready done the dirty job for you! This dirty job has been put in shared files. These shared files are full cross platform and are automagically configured by the configure script. You only need to include them and use their functions to benefit from them.

These files are provided only for drivers, others are provided for all of LCDproc. These files are located in the shared directory, they have a dedicated chapter in this book.

4.2.1. port.h : Parallel port I/O

The file port.h, located in the server/drivers/ directory provide Input/Output and port permissions for the PC compatible parallel port, also known as the LPT port.

Of course, these functions will only work if the computer where LCDproc runs has parallel port! In these situations, the configure script will see this and disable drivers that need a parallel port.

port.h file defines 6 static inline functions for port I/O:

4.2.1.1. Read a byte from port

```
static inline int port_in(unsigned short int port);
```

Returns the content of the byte.

4.2.1.2. Write a char(byte) ’val’ to port

```
static inline void port_out(unsigned short int port, unsigned char val);
```

Returns nothing (void).

4.2.1.3. Get access to a specific port

```
static inline int port_access(unsigned short int port);
```

Returns 0 if successful, -1 if failed.

4.2.1.4. Close access to a specific port

```
static inline int port_deny(unsigned short int port);
```

Returns 0 if successful, -1 if failed.

4.2.1.5. Get access to multiple sequential ports

```
static inline int port_access_full(unsigned short int port, unsigned short
int count);
```

Returns 0 if successful, -1 if failed.

4.2.1.6. Close access to multiple sequential ports

```
static inline int port_deny_full(unsigned short int port, unsigned short int
count);
```

Returns 0 if successful, -1 if failed.

4.2.1.7. Example use

```
#include "port.h"

/* Get access to these 3 ports:
    0x378 (CONTROL),
    0x379 (STATUS) and
    0x37A (DATA)
*/
if ( -1 == port_access_multiple(0x378,3) ) {
/* Access denied, do something */
}

/* Write a ’A’ to the control port */
ort_out(0x378, ’A’);

/* Read from the status port */
char status = port_in(0x379);

/* Close the 3 ports */
port_deny_multiple(0x378,3);
```

Chapter 5. Adding your driver to LCDproc

5.1. Introduction

LCDproc is meant to be modular, it is relatively easy to add new input and output drivers to LCDproc.

This chapter will explain you the major steps and few gotchas of adding your own driver to LCDproc. Enjoy!

5.2. Autoconf, automake, börk börk börk!

How I Learned to Stop Worrying and Love the Configure Script

It was decided pretty early in LCDproc's life to use GNU autoconf and GNU automake. This allows LCDproc to be ported to several platforms with much less effort. It can be quite daunting to understand how autoconf & automake interact with each others and with your code, but don't be discouraged. We have taken great care in making this as simple as possible for programers to add their own driver to LCDproc. Hopefully, you'll only have to modify two files, one for autoconf and one for automake.

The first thing you need to do is to find a name for your driver, it should be as descriptive as possible; most drivers are named after their respective chipset, for example hd44780, mtc_s16209x, sed1330 and stv5730, others are named after the company that makes that particular LCD display, for example CFontz and MtxOrb. Remember that these names are case sensitive. In this chapter, we'll use myDriver (which is an absolute non-descriptive name).

5.2.1. Autoconf and its friend, acinclude.m4

You need to add your driver to function LCD_DRIVERS_SELECT of file acinclude.m4. This can be done in three steps.

5.2.1.1. Step 1

First you need to add your driver name to the list of possible choices in the help screen.

This:

```
AC_ARG_ENABLE(drivers,
   [  --enable-drivers=<list> compile driver for LCDs in <list>.]
```

```
    [                  drivers may be separated with commas.]
    [                  Possible choices are:]
    [                    mtxorb,cfontz,cfontz633,curses,text,lb216,]
    [                    hd44780,joy,irman,lirc,bayrad,glk„mtc_s16209x]
    [                    stv5730,sed1330,sed1520,svga,lcdm001,t6963]
    [                    lcterm,icp_a106]
    [                  \"all\" compiles all drivers],
    drivers="$enableval",
```

becomes:

```
AC_ARG_ENABLE(drivers,
   [  --enable-drivers=<list> compile driver for LCDs in <list>.]
    [                  drivers may be separated with commas.]
    [                  Possible choices are:]
    [                    mtxorb,cfontz,cfontz633,curses,text,lb216,]
    [                    hd44780,joy,irman,lirc,bayrad,glk„mtc_s16209x]
    [                    stv5730,sed1330,sed1520,svga,lcdm001,t6963]
    [                    lcterm,icp_a106,myDriver]
    [                  \"all\" compiles all drivers],
    drivers="$enableval",
```

5.2.1.2. Step 2

Second, you need to add your driver to the list of all drivers.

This:

```
if test "$drivers" = "all"; then
drivers=[mtxorb,cfontz,...biglist...,lcterm,icp_a106]
fi
```

becomes:

```
if test "$drivers" = "all"; then
drivers=[mtxorb,cfontz,...biglist...,lcterm,icp_a106,myDriver]
fi
```

5.2.1.3. Step 3

Then last, you need to add your driver to be big switch-case in this function, see below.

```
myDriver)
DRIVERS="$DRIVERS myDriver${SO}"
actdrivers=["$actdrivers myDriver"]
;;
```

If your driver only works in some platform or requires a particular library or header, you can add your autoconf test here. You can see how other drivers do it, but if you're not sure on how to do this, just send an email to the mailing list and we'll make it for you.

5.2.2. Automake and its friend, Makefile.am

Allready half of the job is done! Not to bad, wasn't it? The rest should be just as easy. In this section, you'll be adding your driver to the file server/drivers/Makefile.am. As you can guess, it's the Makefile for the drivers. This can be done in three (or two) simple steps.

5.2.2.1. Step 1

First, you need to add your driver to the list of drivers in this file, this list is called EXTRA_PROGRAMS.

This

```
EXTRA_PROGRAMS = bayrad CFontz ...biglist... text wirz_sli
```

becomes

```
EXTRA_PROGRAMS = bayrad CFontz ...biglist... text wirz_sli myDriver
```

5.2.2.2. Step 2

This second step is only needed if your driver needs a particular library. If it doesn't, you can skip to step 3.

You basically need to put you driver name followed by _LDADD and egal this to the name of the library that you need. Usually, these library are substituted by a autoconf variable, if you're not comfortable with this, you send an email to the mailing list and we'll set this up for you.

For example, we would put this for our fictional driver

```
myDriver_LDADD = @SOMESTRANGELIB@
```

5.2.2.3. Step 3

Last but not least, you need to specify which source files should be associated with your driver. You put your driver name followed by _SOURCES and egal this to a space separated list of the source and header files. See below for an example.

```
myDriver_SOURCES =  lcd.h myDriver.c myDriver.h report.h
```

5.2.3. Test your setup

You’re almost done! You only need to check out if you didn’t made any mistake. Just run sh autogen.sh to regenerate the configure script and Makefiles, then run ./configure --enable-driver=myDriver and type make. If your driver compiles without error, then congratulations, you’ve just added your driver to LCDproc! Remember to submit a patch to the mailing list so that we can add it to the standard distribution.

If you had an error, just send us an email describing it to the mailing list and we’ll try to help you.

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

A.1. 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.

A.2. 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.

A.3. 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.

A.4. 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.

A.5. 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.

A.6. 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 "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.7. 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.

A.8. 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.

A.9. 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.

A.10. 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.

A.11. 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.12. 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.