# DDE Client

## A Dynamic Link Library for Dynamic Data Exchange

Dipl.-Ing. Robert Matovinovic
Harare, Zimbabwe
robert.matovinovic@web.de

31.10.2006

# Contents

# Abstract

This is the documentation for DDEClient.dll a dynamic link library providing DDE connectivity in C.

# Introduction

DDEClient.dll is a dynamic link library, which exports C procedures to set up a DDE (dynamic data exchange) client. A DDE client is a program, which controls, submit or retrieve data from another program called server. The server provides a DDE service. DDE is an old method to communicate between programs under Windows OS. Still there are applications out there which can only communicate with others via DDE. Such an application was the reason to build DDEClient.

Although the DLL can be used with any program, which can import C procedures, it was written to build an interface with PLT scheme v3.50 and its foreign function interface.

Funny enough, one should think there must be already hell of a lot DDE stuff around to do that without C/C++ programming. Nothing that I found worked out of the box for my case. And nothing compiled nearly error free under Microsoft Visual C++ 7 (MSVC++ 7), the one I happened to have access to. So this became my first project in C/C++ and Windows programming. I should have started with something easier, I know now.

The source was developed in Microsoft VS.NET with VC++ 7. The code uses some features of MSVC++ 7. As far as I know the only Microsoft specific thing I use is atlstr.h, which uses Microsoft specific C++, for the string handling of error messages. But the DLL exports pure C procedures. All in all that means it does not automatically compile with another compiler without any changes. I did not bother over such details. C, C++, Windows and DDE gave me enough troubles.

Currently the software is in beta state and need intense testing that I cannot do on my own. So I appreciate any user and bug reports. Send them including questions, suggestions and critics to robert.matovinovic@web.de.

# Features

- DDEClient.dll enables DDE connections between a client which uses the dll and up to 20 services/server applications.
- Request, execute and poke transactions can be performed synchronously and asynchronously. Advise transactions cannot be performed. They are not implemented since I don't have any idea how they work and no examples to test.
- Error reporting, if there are errors during conversation. Errors can be displayed by system message boxes or the message strings can be displayed wherever the programmer wants to.
- Consistent naming and return values of exported functions to easily get used to them.

The software is provided as is no warranty is given

# Requirements

DDEClient.dll was developed with Microsoft Visual Studio .NET 2003 and tested under Windows XP SP2.

# Files

With this file you should get the following files:

| | |
|---|---|
| DDEClient.dll | Compiled library ready to use |
| DDEClient.lib | Lib file for static linking of library |
| DDEClient.h | Header file for dll use in other C/C++ programs |
| DDEClient.cpp | Source file |
| DCErrors.h | Header file for DDEClient.cpp for texts of error messages |
| DDEClientDLL.pdf | This file |
| DDEClientCons.exe | Example console application |
| DDEClientCons.cpp | Source file for example |

# Installation

Copy the file DDEClient.dll into any folder you desire. Unless you copy it in the same folder as the program that it calls or in Windows\system32 or Windows\system or whatever you have specified in the PATH environment variable you have to call the library with the full path.

# Provided procedures and variables

## *Conventions*

All exported functions and Variables begin with “DC”, which stands for DdeClient like DCInit.

All exported functions except void DCFinalize() and char* DCLastError are boolean to handle errors easily.

## *Concept*

The concept of DDEClient.dll is driven by setting up a DDE conversation with error handling in an easy and consistent manner. Therefore all functions except two are Boolean. Their return values indicate if a function succeeded or not. Subsequently it is possible to handle an error by an if-statement. This approach is supported by an error display function, which will retrieve the error message.

Since the functions do not return any data of the DDE conversation, there is an exported data structure, which let access the data if any are returned.

## *Procedures*

### bool DCInit()

Initializes DDEML.dll which contains all the DDE procedures used by the other procedures. Has to be called before any other of the following procedures.

### bool DCConnect(WORD* pConvNo, char szService[], char szTopic[])

| | |
|---|---|
| pConvNo | [out] Pointer to a variable which takes the conversation number. The conversation number is the index which always has to be given the following procedures to distinguish multiple conversations. |
| szService | [in] Name of the DDE service application, i.e. "winword", usually the service name is the name of the programs exe-file. |
| szTopic | [in] String of the topic the service is connected for. There are different themes over which you can communicate with the service. Mostly a topic is a file name which is currently open in the service. Most services support also the "system" topic. |

Establishes DDE connection to a service of a server application, i.e. Excel.

### bool DCTransaction(WORD wC, char szType[],char szItem[], char szData[], char szFormat[], int nTimeout, char szAccess[])

| | |
|---|---|
| wC | [in] conversation number, given by DCConnect |
| szType | [in] transaction type, can be "execute", "poke", "request" not case sensitive |
| szItem | [in] Somehow the command you want to send to the service. I.e. to see what topics (see above) are available you can set it to "Topics", after connecting the service with topic system, but not all programs support this topic. |
| szData | [in] the data you send by a poke command |
| szFormat | [in] format of DDE data. It determines in which format the data between the applications is sent. See "Supported Clipboard Formats" below. |
| nTimeout | [in] any number, which indicates the waiting time in milliseconds until the transaction waits to be finished before returning to the program. If the number is greater 0 and the service doesn't return in time an error occurs. 0 indicates an asynchronous transaction, which means the program is not waiting for the answer from the service but resumes execution. See also DCAsynchTransactionCompleted |
| szAccess | [in] flag for format of the DDE data, can be "byte" or "string", for convenience to access the data without further conversion. It is different from szFormat in so far that szFormat handles the data format in between the communicating applications; the service has to know in which format the client wants the data depending on the command given. How the user formats it, is something different. |

It is a general procedure for execute, poke and request transactions, the communication with the service.

### bool DCRequestString(WORD wC, char szItem[], int nTimeout)

Abbreviation of a DCTransaction request, which always stores DDE data as string; meaning of arguments see above.

### bool DCRequest(WORD wC, char szItem[], char szFormat[], int nTimeout)

Abbreviation of a DCTransaction request, which always stores DDE data as bytes; meaning of arguments see above.

### bool DCAsynchTransactionCompleted(WORD wC, DWORD dwTransID, bool bWait)

| | |
|---|---|
| wC | [in] conversation number, given by DCConnect |
| dwTransID | [in] transaction number, returned by the transaction, see DCDA[wC]->dwTransID |
| bWait | [in] flag to indicate whether the procedure should wait until the transaction is completed before returning (true) or do the test for finishing only once and return (false) |

Important procedure for asynchronous transactions. Since the program does not wait for asynchronous transactions to be finished, otherwise they were synchronous, at one stage the program itself must check if the transaction is completed. This is especially necessary for request transactions because otherwise the DDE data cannot be accessed. Every transaction gets it unique ID. With this ID and the conversation number can be checked if the transaction has completed. If so the procedure returns true otherwise false.

Note: This procedure can only be used safely if no other transaction has been executed between the call for the transaction and its call. This is because all messages sent by the service, regardless which transaction and which conversation are concerned, will be dispatched by the procedure, and only the one indicated by the procedure parameters will be checked for. So others may be finished as well but cannot be indicated by the procedure, because the messages which cause their finishing are already processed. In such a case, DCDA[i]->dwTransID = = DCDA[i]->dwCbTransID can be checked and if true the transaction is finished.

### bool DCAbandonTransaction(WORD wC, DWORD dwTransID)

| | |
|---|---|
| wC | [in] conversation number, given by DCConnect |
| dwTransID | [in] transaction number, returned by the transaction, see DCDA[wC]->dwTransID |

Procedure to abandon an asynchronous transaction if it takes to long or for whatever reason

### bool DCFreeDdeMem(WORD wC)

| | |
|---|---|
| wC | [in] conversation number, given by DCConnect |

Frees so called handles of the DDE connection and frees memory allocated to access DDE data.

Important! Procedure has to be called after every request transaction otherwise memory consumption increases with next request transaction and cannot be freed later on.

**bool DCDisconnect(WORD wConvNo)**

| | |
|---|---|
| wConvNo | [in] conversation number, given by DCConnect |

Closes down the DDE connection to a service of a server, frees memory

**bool DCUninit()**

Uninitialize the DDEML.dll

**void DCFinalize()**

Frees memory used by DDEClient.dll, which was not freed previously, what is most important for error handling, but is not able to free memory whose pointers were already changed, which happens if you forget to call DCFreeDdeMem after a request transaction.

**char* DCLastError()**

Returns either an error string if the variable bDCErrorExport  is true or displays the error message in a message box.

Note: Giving back the error message in a message box is not advisable except for starting to make oneself familiar with the library, because the display of the message box does not automatically stop the calling program, but the executions in the DLL. That means if a message box appears other calls to the dll will fail until the message box is clicked away and therefore cause further errors. So it is advisable to handle the error strings in the calling program by setting DCErrorExport to TRUE, which is the default.

**char* DCVersion()**

returns the version of the dll.

## *Variables*

### DCDA[wC]

DDE data structure for each conversation accessing data received by DDE transaction. Since the procedures do not return any data, except DCConnect which gives back the conversation number, this data structure holds all the variables to access the data.

| | |
|---|---|
| wC | [in] conversation number, given by DCConnect |
| pData | BYTE*, pointer to begin of DDE data returned by the call of DdeAccessData during the transaction or by the callback function |
| dwLen | DWORD, length of DDE data returned by DdeAccessData |
| pszData | char*, string pointer to DDE data, if the data should be accessed as string |
| szAccType[6] | char, string for access type of data, this is a variable for the convenience to indicate if the DDE data should be returned directly as string or bytes. Can be set to “string” or “byte” |
| dwTransID | DWORD, ID of transaction, used to determine completion of an asynchronous transaction by comparing it with dwCbTransID. If both are equal the transaction has been completed |
| dwCbTransID | DWORD, ID of transaction, returned by callback function with asynch. transaction by comparing it with dwTransID. If both are equal the transaction has been completed. |

### bDCErrorExport

Boolean, controls the output of DCLastError(). Set to TRUE DCLastError() gives back the error message as a string, what means it exports it to the calling program. Set to FALSE DCLastError() displays the last error message in a message box, not exporting it.

# Supported Clipboard Formats

The formats can be used by DDEClient functions in the szFormat variable as strings. They determine the structure of the DDE data which is returned by the service. How to retrieve the data out of this formats in your DDE conversation you have to look up the Windows SDK. As long as you use text based formats things should work fine. For other formats you have to set szAccess in any case to “byte”.

| | |
|---|---|
| CF_TEXT | Null-terminated, plain ANSI text in a global memory block. |
| CF_BITMAP | A bitmap compatible with Windows 2.x. |
| CF_METAFILEPICT | A Windows metafile with some additional information about how the metafile should be displayed. |
| CF_SYLK | An ASCII text format used by some older Microsoft products. |
| CF_DIF Software | Art's data interchange format (DIF). Also an ASCII text format. |
| CF_TIFF | Tag image file format (TIFF) data in a global memory block. |
| CF_OEMTEXT | Similar to CF_TEXT but using the OEM character set. |
| CF_DIB | A global memory block containing a Windows device-independent |

| | |
|---|---|
| | bitmap (DIB) as a BITMAPINFO structure followed by the bitmap bits. |
| CF_PALETTE | A color-palette handle. (Used in conjunction with CF_DIB.) |
| CF_PENDATA | Data is for the pen extensions to Windows. |
| CF_RIFF | Resource interchange file format (RIFF) data as a global memory block. |
| CF_WAVE | A specific case of RIFF in which the contained data is a waveform (sampled sound). |

# Usage

## *Structure of DDE process*

To implement a DDE conversation you need to set up a number of steps which are described below.

As structure of DDE process in this document is regarded the set of commands which is needed for a successful DDE conversation. The procedures used in DDEClient.dll have to be used as follows.

| | |
|---|---|
| DCInit | Initializes DDEML.dll |
| DCConnect | Establishes DDE connection to a service |
| DCTransaction or<br>DCRequestString or<br>DCRequest | One of these procedures actually does the DDE transaction, i.e. open a new file, send/request data |
| DCAsynchTransactionCompleted | Necessary only to retrieve data of asynchronous transactions |
| Do anything with your DDE data | |
| DCFreeDdeMem | Necessary after every request transaction when data handling is finished, if there are more than one between DCConnect and DCDisconnect. Otherwise memory leaks occur |
| DCDisconnect | Closes down the DDE connection to a service of a server, frees memory |
| DCUninit | Uninitializes DDEML.dll |
| DCFinalize | Frees memory used by DDEClient.dll, but not necessarily all, see Memory Issues |

DCInit and DCUninit have only to be called once in a program. DCConnect, DCDisconnect can be called indefinite times as a pair in a program, with the desired transaction(s) in between. DCFreeDdeMem has to be called after each request transaction when all DDE data processing is done and for any reason DCDisconnect is not appropriate to call. This may be if you want to omit to call DCConnect, DCDisconnect every time with a transaction. DCFreeDdeMem is not incorporated in the transaction procedures, to leave the freedom to the programmer, what to do with the DDE data. Otherwise the data had to be copied to a buffer, which has to have a fixed format, which might not suit the data in terms of data type and length of data.

The important procedures for the conversation are DCTransaction, DCRequestString, DCRequest. DCTransaction is the most general. It can be used for request, execute and poke transactions. DCRequestString is a convenient abbreviation of DCTransaction, which always stores DDE data as string. DCRequest abbreviates also DCTransaction but stores DDE data in bytes, thus leaving it to the programmer how to use the data.

### Synchronous/Asynchronous Transactions

There are these to modes of transactions. Synchronous transactions wait for the service to answer for a certain time, after that they signal an error. During the transaction as long as it is not finished or the timeout is not reached it enters a modal loop which blocks all other actions of the program.

Asynchronous transactions are appropriate, if you don't want to limit your transaction by a timeout, when it has to be finished. Another advantage is the possibility to execute other calculations in the calling program while the transaction is processed by the service, because the transaction doesn't enter a modal loop until it is finished as in synchronous mode. But you have to call DCAsynchTransactionCompleted in order to retrieve data after an asynchronous request transaction. This seems a strange behavior but the callback function which processes the data of an asynchronous transaction is only invoked, if the message from the service is catch by PeekMessage in DCAsynchTransactionCompleted.

## *Data Handling*

Handling the data of the DDE conversation the DCDA structure is used. There is an array of currently 20 such structures, which means it is possible to have 20 connections to different services a one time. DCDA allows accessing the data returned by a transaction. The only transaction which gives back data now is the request transaction.

Besides DCTransaction there are two abbreviated types of request transactions: DCRequestString and DCRequest. The first writes data as string. The second writes data as bytes. Both write the data on the heap which is accessed through pointers of the DCDA structure. DCDA[i]->pszData for the string variant DCDA[i]->pData and DCDA[i]->dwLen, as pointer and length information, for the byte variant. This approach does not limit the size of data, but data is written to the heap which means in fact they are present double in memory. Therefore it is of big importance to free memory as soon as the data is not used any more.

If an error occurs during a transaction which allocates memory (only request transactions) the memory is freed by the error branch of the transaction automatically. However memory allocated for the conversation is not freed then, since it is up to the programmer how he handles the error. To quit the program completely use DCFinalize to free all allocated memory.

## *Memory Issues*

The dll can provide access to DDE data only by writing it to the heap to dynamically adjust the space which is needed. Therefore memory has to be freed explicitly after usage for any request transaction. This is partly done automatically by the dll, partly the user is responsible for that, because it is not known in advance, when the user will not need the data anymore. There are two possible schemes to follow to assure memory freeing (DCInit and DCUninit are omitted in that pseudo code):

```
1.  DCConnect
        DCTransaction (or DCRequest, or DCRequestString)
        Do anything with your data
    DCDisconnect
2.  DCConnect
        DCTransaction (or DCRequest, or DCRequestString)
        Do anything with your data
        DCFreeDdeMem
        DCTransaction (or DCRequest, or DCRequestString)
        Do anything with your data
        DCFreeDdeMem
        .
        .
        .
    DCDisconnect
```

The first scheme is to prefer if there are only a few DDE commands to send. The second is to prefer if you have to communicate extensively, since DCConnect may take some time to connect to the service over and over again.

Note: If you forget to free memory in the second scheme between transactions this memory becomes inaccessible while still allocated! Thus causing a memory leak which can consume all your memory, if for example you make this mistake in a loop.

Keep in mind, the space once assigned for the program on the heap will never be reduced, by freeing memory. It is kept as memory space the program can use. Freeing memory means therefore that the heap already assigned to the program can be used for other assignments. Should the already assigned heap be too small, heap is added. The heap will grow with the number of conversations run parallel. To this is added the size of the biggest single DDE data which is returned from a service. If you wish to reduce the heap assigned to the program completely even despite you free memory every time you should, close all conversations and call DCFinalize().

## *Error Handling*

All DDE procedures return FALSE, if they do not succeed. In that case they save a message in a CString before they return. This CString is accessed by the procedure DCLastError. To get the error message one has to check for the success of the procedure and if not call DCLastError().

C Example code:

```
if(!DCTransaction(wConvNo,"request",szItem,szData,szFormat,1000, "string"))
      ErrorOccured();
else
{
      // output data to console after synchronous transaction complete
      cprintf("\r\n string:\r\n%s\r\n",DCDA[wConvNo]->pszData);
      // free memory after data usage
      DCFreeDdeMem(wConvNo);
}
```

```
void ErrorOccured()
{
      if (bDCErrorExport)
            // print error message to Console
            cprintf("\r\nError: %s\r\n",DCLastError());
      else
            // display error in message box
            DCLastError();
}
```

The example above how simple error handling can be implemented. The procedure ErrorOccured() could handle also anything you want to be handled.

Note: The check if a sent command was processes properly by the service is not consistent. Whereas a wrong request command does not return data, you can't take that check for execute or poke commands. Thus make sure that all the commands you send are error free!

Note: Error messages are not conversation specific. There is only one queue of error messages for all conversations.

## *Multiple DDE Connections*

You can connect to up to 20 service applications at one time. DCConnect takes a pointer to a variable in which a conversation number is stored which specifies a certain conversation. Subsequently this conversation number has to be used for every action concerning this conversation and its data.

With asynchronous transactions it is possible to parallelize the connections thus making parallel computation possible.

The conversation number for different conversations starts from 0! The next conversation number will always be determined by the first unused number. That means if you open for example 3 conversations which will have the numbers 0, 1 and 2, and you close conversation 1 and opens a new one, this new one will get the conversation number 1, since it is the first free number in the row then.

## *Example*

The file DDEClientCons.cpp gives commented examples of the usage of DDEClient.dll. DDEClientCons.exe is the compiled console application of DDEClientCons.cpp using Microsoft Word as service application. Open Microsoft Word, make sure you have write access to your `C:\` drive and there is no file named `test.doc`.

# Inside DDEClient.dll

Here are some comments on development issues of the dll.

## *Comments on Procedures*

### Callback Function

The callback function handles transactions of the DDE session. It reacts to certain types of transactions. A client application can not invoke every transaction so only some of them are handled in the callback function.

`XTYP_XACT_COMPLETE` is only invoked, when a transaction is asynchronous. The invocation is operation system dependent. DCAsynchTransactionCompleted considers this.

`XTYP_CONNECT` is not needed; it is invoked on the service side to state that a connection is accepted.

`XTYP_DISCONNECT` is invoked, if the partner application in our case the service is terminating the connection.

`XTYP_ERROR` is invoked currently on only one DDEML error by Microsofts implementation that is out of memory. And this is only because during DDE initialization the flag MF_ERRORS is set.

## *Comments on Data structures*

The DLL defines two data structures for keeping data related to the DDE conversation. They are named DDEVARS and DCDATAACCESS.

DDEVARS keeps all variables needed by the conversation for the genuine DDE commands. Their names are the same as in the reference for the `DdeClientTransaction` command of the Windows SDK. DDEVARS is only needed inside the DLL.

DCDATAACCESS keeps the application specific variables to access the DDE data returned by `DdeClientTransaction` with request transactions and to check whether an asynchronous transaction has been completed. A pointer to the data structure is exported, so that it can be accessed from programs calling DLL procedures.

Memory for both structures is allocated explicitly, which is especially important with asynchronous transactions, since variables on file scope will be deallocated after the callback function terminates. Therefore memory has to be freed explicitly by calling DCDdeDisconnect. Also the structure definition is in the header file for in the calling program otherwise the pointer to the structure cannot be imported.

Since there are different conversations possible at the same time the data structures of every single conversation have to be separated. This is done by an array of pointers, which addresses the data of each conversation by a distinct array index.

## Thanks

I owe a lot concerning this project to the following people, companies, which I never met but through the presence of their work on the internet:

- **www.angelfire.com/biz/rhaminisys/ddeinfo.html** for their info on DDE and their example archive xlddec.zip
- **Jürgen Wolf and Galileo Computing** for his very instructive book on C programming “C von A bis Z”, at www.galileo-press.de/openbook/c_von_a_bis_z/
- **Steven Randy Davis** for his book “C++ for Dummies”
- **www.functionx.com**, who ever are behind it, for its perfect tutorial on creating dlls at www.functionx.com/visualc/libraries/win32dll.htm, I wouldn’t have started without it.
- **Faweb, Takebishi** for their DDE example at http://www.faweb.net/us/ioserver/sample_vc.html
- **Graeme S. Roy** for his library mpatrol to check this dll for memory leaks at www.cbmamiga.demon.co.uk/mpatrol/,
- **Robert Schmitt** for his article and binaries to get mpatrol up and running so quickly at www.codeguru.com/cpp/w-p/win32/tutorials/article.php/c12231/.
- **Microsofts knowledgebase** for some articles like KB279721, but the DDE documentation is mostly quiet unreadable in my opinion.

## License

This software and all related documentation and files are licensed under GNU Lesser General Public License. A copy of it is printed here:

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[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.]

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 packages--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.

## GNU LESSER GENERAL PUBLIC LICENSE
## 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

## 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 a brief 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., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301
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!