The GNU OpenMP Implementation

Published by the Free Software Foundation
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA

Short Contents

Table of Contents

Introduction

This manual documents the usage of libgomp, the GNU implementation of the OpenMP Application Programming Interface (API) for multi-platform shared-memory parallel programming in C/C++ and Fortran.

1 Enabling OpenMP

To activate the OpenMP extensions for C/C++ and Fortran, the compile-time flag `-fopenmp` must be specified. This enables the OpenMP directive `#pragma omp` in C/C++ and `!$omp` directives in free form, `c$omp`, `*$omp` and `!$omp` directives in fixed form, `!$` conditional compilation sentinels in free form and `c$`, `*$` and `!$` sentinels in fixed form, for Fortran. The flag also arranges for automatic linking of the OpenMP runtime library (Chapter 2 [Runtime Library Routines], page 5).

A complete description of all OpenMP directives accepted may be found in the OpenMP Application Program Interface manual, version 2.5.

2 Runtime Library Routines

The runtime routines described here are defined by section 3 of the OpenMP specifications in version 2.5.

Control threads, processors and the parallel environment.

Initialize, set, test, unset and destroy simple and nested locks.

Portable, thread-based, wall clock timer.

2.1 `omp_get_dynamic` – Dynamic teams setting

Description:

This function returns `true` if enabled, `false` otherwise. Here, `true` and `false` represent their language-specific counterparts.

The dynamic team setting may be initialized at startup by the `OMP_DYNAMIC` environment variable or at runtime using `omp_set_dynamic`. If undefined, dynamic adjustment is disabled by default.

C/C++:

Prototype: `int omp_get_dynamic();`

Fortran:

Interface: `logical function omp_get_dynamic()`

See also: Section 2.8 [omp_set_dynamic], page 7, Section 3.1 [OMP_DYNAMIC], page 13

Reference: OpenMP specifications v2.5, section 3.2.8.

2.2 `omp_get_max_threads` – Maximum number of threads

Description:

Return the maximum number of threads used for parallel regions that do not use the clause `num_threads`.

C/C++:

Prototype: `int omp_get_max_threads();`

Fortran:

Interface: `integer function omp_get_max_threads()`

See also: Section 2.10 [omp_set_num_threads], page 8, Section 2.8 [omp_set_dynamic], page 7

Reference: OpenMP specifications v2.5, section 3.2.3.

2.3 `omp_get_nested` – Nested parallel regions

Description:
This function returns `true` if nested parallel regions are enabled, `false` otherwise. Here, `true` and `false` represent their language-specific counterparts.

Nested parallel regions may be initialized at startup by the `OMP_NESTED` environment variable or at runtime using `omp_set_nested`. If undefined, nested parallel regions are disabled by default.

C/C++:
Prototype: `int omp_get_nested();`

Fortran:
Interface: `integer function omp_get_nested()`

See also: Section 2.9 [omp_set_nested], page 8, Section 3.2 [OMP_NESTED], page 13

Reference: OpenMP specifications v2.5, section 3.2.10.

2.4 `omp_get_num_procs` – Number of processors online

Description:
Returns the number of processors online.

C/C++:
Prototype: `int omp_get_num_procs();`

Fortran:
Interface: `integer function omp_get_num_procs()`

Reference: OpenMP specifications v2.5, section 3.2.5.

2.5 `omp_get_num_threads` – Size of the active team

Description:
The number of threads in the current team. In a sequential section of the program `omp_get_num_threads` returns 1.

The default team size may be initialized at startup by the `OMP_NUM_THREADS` environment variable. At runtime, the size of the current team may be set either by the `NUM_THREADS` clause or by `omp_set_num_threads`. If none of the above were used to define a specific value and `OMP_DYNAMIC` is disabled, one thread per CPU online is used.

C/C++:
Prototype: `int omp_get_num_threads();`

Fortran:
Interface: `integer function omp_get_num_threads()`

See also: Section 2.2 [omp_get_max_threads], page 5, Section 2.10 [omp_set_num_threads], page 8, Section 3.3 [OMP_NUM_THREADS], page 13

Reference: OpenMP specifications v2.5, section 3.2.2.

2.6 omp_get_thread_num – Current thread ID

Description:

Unique thread identification number. In a sequential parts of the program, `omp_get_thread_num` always returns 0. In parallel regions the return value varies from 0 to `omp_get_max_threads`-1 inclusive. The return value of the master thread of a team is always 0.

C/C++:

Prototype: `int omp_get_thread_num();`

Fortran:

Interface: `integer function omp_get_thread_num()`

See also: Section 2.2 [omp_get_max_threads], page 5

Reference: OpenMP specifications v2.5, section 3.2.4.

2.7 omp_in_parallel – Whether a parallel region is active

Description:

This function returns `true` if currently running in parallel, `false` otherwise. Here, `true` and `false` represent their language-specific counterparts.

C/C++:

Prototype: `int omp_in_parallel();`

Fortran:

Interface: `logical function omp_in_parallel()`

Reference: OpenMP specifications v2.5, section 3.2.6.

2.8 omp_set_dynamic – Enable/disable dynamic teams

Description:

Enable or disable the dynamic adjustment of the number of threads within a team. The function takes the language-specific equivalent of `true` and `false`, where `true` enables dynamic adjustment of team sizes and `false` disables it.

C/C++:

Prototype: `void omp_set_dynamic(int);`

Fortran:

Interface:
```
subroutine omp_set_dynamic(set)
integer, intent(in) :: set
```

See also: Section 3.1 [OMP_DYNAMIC], page 13, Section 2.1 [omp_get_dynamic], page 5

Reference: OpenMP specifications v2.5, section 3.2.7.

2.9 omp_set_nested – Enable/disable nested parallel regions

Description:

Enable or disable nested parallel regions, i.e., whether team members are allowed to create new teams. The function takes the language-specific equivalent of `true` and `false`, where `true` enables dynamic adjustment of team sizes and `false` disables it.

C/C++:

Prototype: `void omp_set_dynamic(int);`

Fortran:

Interface:
```
subroutine omp_set_dynamic(set)
integer, intent(in) :: set
```

See also: Section 3.2 [OMP_NESTED], page 13, Section 2.3 [omp_get_nested], page 6

Reference: OpenMP specifications v2.5, section 3.2.9.

2.10 omp_set_num_threads – Set upper team size limit

Description:

Specifies the number of threads used by default in subsequent parallel sections, if those do not specify a `num_threads` clause. The argument of `omp_set_num_threads` shall be a positive integer.

C/C++:

Prototype: `void omp_set_num_threads(int);`

Fortran:

Interface:
```
subroutine omp_set_num_threads(set)
integer, intent(in) :: set
```

See also: Section 3.3 [OMP_NUM_THREADS], page 13, Section 2.5 [omp_get_num_threads], page 6, Section 2.2 [omp_get_max_threads], page 5

Reference: OpenMP specifications v2.5, section 3.2.1.

2.11 omp_init_lock – Initialize simple lock

Description:

Initialize a simple lock. After initialization, the lock is in an unlocked state.

C/C++:

Prototype: `void omp_init_lock(omp_lock_t *lock);`

Fortran:

Interface:
```
subroutine omp_init_lock(lock)
integer(omp_lock_kind), intent(out) :: lock
```

See also: Section 2.15 [omp_destroy_lock], page 10

Reference: OpenMP specifications v2.5, section 3.3.1.

2.12 `omp_set_lock` – Wait for and set simple lock

Description:

Before setting a simple lock, the lock variable must be initialized by `omp_init_lock`. The calling thread is blocked until the lock is available. If the lock is already held by the current thread, a deadlock occurs.

C/C++:

Prototype: `void omp_set_lock(omp_lock_t *lock);`

Fortran:

Interface:
```
subroutine omp_set_lock(lock)
integer(omp_lock_kind), intent(out) :: lock
```

See also: Section 2.11 [omp_init_lock], page 8, Section 2.13 [omp_test_lock], page 9, Section 2.14 [omp_unset_lock], page 9

Reference: OpenMP specifications v2.5, section 3.3.3.

2.13 `omp_test_lock` – Test and set simple lock if available

Description:

Before setting a simple lock, the lock variable must be initialized by `omp_init_lock`. Contrary to `omp_set_lock`, `omp_test_lock` does not block if the lock is not available. This function returns `true` upon success,`false` otherwise. Here, `true` and `false` represent their language-specific counterparts.

C/C++:

Prototype: `int omp_test_lock(omp_lock_t *lock);`

Fortran:

Interface:
```
subroutine omp_test_lock(lock)
logical(omp_logical_kind) :: omp_test_lock
integer(omp_lock_kind), intent(out) :: lock
```

See also: Section 2.11 [omp_init_lock], page 8, Section 2.12 [omp_set_lock], page 9, Section 2.12 [omp_set_lock], page 9

Reference: OpenMP specifications v2.5, section 3.3.5.

2.14 `omp_unset_lock` – Unset simple lock

Description:

A simple lock about to be unset must have been locked by `omp_set_lock` or `omp_test_lock` before. In addition, the lock must be held by the thread calling `omp_unset_lock`. Then, the lock becomes unlocked. If one ore more threads attempted to set the lock before, one of them is chosen to, again, set the lock for itself.

C/C++:

Prototype: `void omp_unset_lock(omp_lock_t *lock);`

Fortran:

Interface: `subroutine omp_unset_lock(lock)`
`integer(omp_lock_kind), intent(out) :: lock`

See also: Section 2.12 [omp_set_lock], page 9, Section 2.13 [omp_test_lock], page 9

Reference: OpenMP specifications v2.5, section 3.3.4.

2.15 `omp_destroy_lock` – Destroy simple lock

Description:
Destroy a simple lock. In order to be destroyed, a simple lock must be in the unlocked state.

C/C++:

Prototype: `void omp_destroy_lock(omp_lock_t *);`

Fortran:

Interface: `subroutine omp_destroy_lock(lock)`
`integer(omp_lock_kind), intent(inout) :: lock`

See also: Section 2.11 [omp_init_lock], page 8

Reference: OpenMP specifications v2.5, section 3.3.2.

2.16 `omp_init_nest_lock` – Initialize nested lock

Description:
Initialize a nested lock. After initialization, the lock is in an unlocked state and the nesting count is set to zero.

C/C++:

Prototype: `void omp_init_nest_lock(omp_nest_lock_t *lock);`

Fortran:

Interface: `subroutine omp_init_nest_lock(lock)`
`integer(omp_nest_lock_kind), intent(out) :: lock`

See also: Section 2.20 [omp_destroy_nest_lock], page 12

Reference: OpenMP specifications v2.5, section 3.3.1.

2.17 `omp_set_nest_lock` – Wait for and set simple lock

Description:
Before setting a nested lock, the lock variable must be initialized by `omp_init_nest_lock`. The calling thread is blocked until the lock is available. If the lock is already held by the current thread, the nesting count for the lock in incremented.

C/C++:

Prototype: `void omp_set_nest_lock(omp_nest_lock_t *lock);`

Fortran:

Interface:
```
subroutine omp_set_nest_lock(lock)
integer(omp_nest_lock_kind), intent(out) :: lock
```

See also: Section 2.16 [omp_init_nest_lock], page 10, Section 2.19 [omp_unset_nest_lock], page 11

Reference: OpenMP specifications v2.5, section 3.3.3.

2.18 `omp_test_nest_lock` – Test and set nested lock if available

Description:

Before setting a nested lock, the lock variable must be initialized by `omp_init_nest_lock`. Contrary to `omp_set_nest_lock`, `omp_test_nest_lock` does not block if the lock is not available. If the lock is already held by the current thread, the new nesting count is returned. Otherwise, the return value equals zero.

C/C++:

Prototype:
```
int omp_test_nest_lock(omp_nest_lock_t *lock);
```

Fortran:

Interface:
```
integer function omp_test_nest_lock(lock)
integer(omp_integer_kind) :: omp_test_nest_lock
integer(omp_nest_lock_kind), intent(inout) :: lock
```

See also: Section 2.11 [omp_init_lock], page 8, Section 2.12 [omp_set_lock], page 9, Section 2.12 [omp_set_lock], page 9

Reference: OpenMP specifications v2.5, section 3.3.5.

2.19 `omp_unset_nest_lock` – Unset nested lock

Description:

A nested lock about to be unset must have been locked by `omp_set_nested_lock` or `omp_test_nested_lock` before. In addition, the lock must be held by the thread calling `omp_unset_nested_lock`. If the nesting count drops to zero, the lock becomes unlocked. If one ore more threads attempted to set the lock before, one of them is chosen to, again, set the lock for itself.

C/C++:

Prototype:
```
void omp_unset_nest_lock(omp_nest_lock_t *lock);
```

Fortran:

Interface:
```
subroutine omp_unset_nest_lock(lock)
integer(omp_nest_lock_kind), intent(out) :: lock
```

See also: Section 2.17 [omp_set_nest_lock], page 10

Reference: OpenMP specifications v2.5, section 3.3.4.

2.20 `omp_destroy_nest_lock` – Destroy nested lock

Description:
Destroy a nested lock. In order to be destroyed, a nested lock must be in the unlocked state and its nesting count must equal zero.

C/C++:
Prototype: `void omp_destroy_nest_lock(omp_nest_lock_t *);`

Fortran:
Interface: `subroutine omp_destroy_nest_lock(lock)`
`integer(omp_nest_lock_kind), intent(inout) :: lock`

See also: Section 2.11 [omp_init_lock], page 8

Reference: OpenMP specifications v2.5, section 3.3.2.

2.21 `omp_get_wtick` – Get timer precision

Description:
Gets the timer precision, i.e., the number of seconds between two successive clock ticks.

C/C++:
Prototype: `double omp_get_wtick();`

Fortran:
Interface: `double precision function omp_get_wtick()`

See also: Section 2.22 [omp_get_wtime], page 12

Reference: OpenMP specifications v2.5, section 3.4.2.

2.22 `omp_get_wtime` – Elapsed wall clock time

Description:
Elapsed wall clock time in seconds. The time is measured per thread, no guarantee can bee made that two distinct threads measure the same time. Time is measured from some "time in the past". On POSIX compliant systems the seconds since the Epoch (00:00:00 UTC, January 1, 1970) are returned.

C/C++:
Prototype: `double omp_get_wtime();`

Fortran:
Interface: `double precision function omp_get_wtime()`

See also: Section 2.21 [omp_get_wtick], page 12

Reference: OpenMP specifications v2.5, section 3.4.1.

3 Environment Variables

The variables `OMP_DYNAMIC`, `OMP_NESTED`, `OMP_NUM_THREADS` and `OMP_SCHEDULE` are defined by section 4 of the OpenMP specifications in version 2.5, while `GOMP_CPU_AFFINITY` and `GOMP_STACKSIZE` are GNU extensions.

3.1 `OMP_DYNAMIC` – Dynamic adjustment of threads

Description:
Enable or disable the dynamic adjustment of the number of threads within a team. The value of this environment variable shall be `TRUE` or `FALSE`. If undefined, dynamic adjustment is disabled by default.

See also: Section 2.8 [omp_set_dynamic], page 7

Reference: OpenMP specifications v2.5, section 4.3

3.2 `OMP_NESTED` – Nested parallel regions

Description:
Enable or disable nested parallel regions, i.e., whether team members are allowed to create new teams. The value of this environment variable shall be `TRUE` or `FALSE`. If undefined, nested parallel regions are disabled by default.

See also: Section 2.9 [omp_set_nested], page 8

Reference: OpenMP specifications v2.5, section 4.4

3.3 `OMP_NUM_THREADS` – Specifies the number of threads to use

Description:
Specifies the default number of threads to use in parallel regions. The value of this variable shall be positive integer. If undefined one thread per CPU online is used.

See also: Section 2.10 [omp_set_num_threads], page 8

Reference: OpenMP specifications v2.5, section 4.2

3.4 `OMP_SCHEDULE` – How threads are scheduled

Description:
Allows to specify `schedule type` and `chunk size`. The value of the variable shall have the form: `type[,chunk]` where `type` is one of `static`, `dynamic` or `guided`. The optional `chunk size` shall be a positive integer. If undefined, dynamic scheduling and a chunk size of 1 is used.

Reference: OpenMP specifications v2.5, sections 2.5.1 and 4.1

3.5 GOMP_CPU_AFFINITY – Bind threads to specific CPUs

Description:

Binds threads to specific CPUs. The variable should contain a space- or comma-separated list of CPUs. This list may contain different kind of entries: either single CPU numbers in any order, a range of CPUs (M-N) or a range with some stride (M-N:S). CPU numbers are zero based. For example, `GOMP_CPU_AFFINITY="0 3 1-2 4-15:2"` will bind the initial thread to CPU 0, the second to CPU 3, the third to CPU 1, the fourth to CPU 2, the fifth to CPU 4, the sixth through tenth to CPUs 6, 8, 10, 12, and 14 respectively and then start assigning back from the beginning of the list. `GOMP_CPU_AFFINITY=0` binds all threads to CPU 0.

There is no GNU OpenMP library routine to determine whether a CPU affinity specification is in effect. As a workaround, language-specific library functions, e.g., `getenv` in C or `GET_ENVIRONMENT_VARIABLE` in Fortran, may be used to query the setting of the `GOMP_CPU_AFFINITY` environment variable. A defined CPU affinity on startup cannot be changed or disabled during the runtime of the application.

If this environment variable is omitted, the host system will handle the assignment of threads to CPUs.

3.6 GOMP_STACKSIZE – Set default thread stack size

Description:

Set the default thread stack size in kilobytes. This is in opposition to `pthread_attr_setstacksize` which gets the number of bytes as an argument. If the stacksize can not be set due to system constraints, an error is reported and the initial stacksize is left unchanged. If undefined, the stack size is system dependent.

Reference: GCC Patches Mailinglist, GCC Patches Mailinglist

4 The libgomp ABI

The following sections present notes on the external ABI as presented by libgomp. Only maintainers should need them.

4.1 Implementing MASTER construct

```
if (omp_get_thread_num () == 0)
  block
```

Alternately, we generate two copies of the parallel subfunction and only include this in the version run by the master thread. Surely that's not worthwhile though...

4.2 Implementing CRITICAL construct

Without a specified name,

```
void GOMP_critical_start (void);
void GOMP_critical_end (void);
```

so that we don't get COPY relocations from libgomp to the main application.

With a specified name, use omp_set_lock and omp_unset_lock with name being transformed into a variable declared like

```
omp_lock_t gomp_critical_user_<name> __attribute__((common))
```

Ideally the ABI would specify that all zero is a valid unlocked state, and so we wouldn't actually need to initialize this at startup.

4.3 Implementing ATOMIC construct

The target should implement the `__sync` builtins.

Failing that we could add

```
void GOMP_atomic_enter (void)
void GOMP_atomic_exit (void)
```

which reuses the regular lock code, but with yet another lock object private to the library.

4.4 Implementing FLUSH construct

Expands to the `__sync_synchronize` builtin.

4.5 Implementing BARRIER construct

```
void GOMP_barrier (void)
```

4.6 Implementing THREADPRIVATE construct

In _most_ cases we can map this directly to `__thread`. Except that OMP allows constructors for C++ objects. We can either refuse to support this (how often is it used?) or we can implement something akin to .ctors.

Even more ideally, this ctor feature is handled by extensions to the main pthreads library. Failing that, we can have a set of entry points to register ctor functions to be called.

4.7 Implementing PRIVATE clause

In association with a PARALLEL, or within the lexical extent of a PARALLEL block, the variable becomes a local variable in the parallel subfunction.

In association with FOR or SECTIONS blocks, create a new automatic variable within the current function. This preserves the semantic of new variable creation.

4.8 Implementing FIRSTPRIVATE LASTPRIVATE COPYIN and COPYPRIVATE clauses

Seems simple enough for PARALLEL blocks. Create a private struct for communicating between parent and subfunction. In the parent, copy in values for scalar and "small" structs; copy in addresses for others TREE_ADDRESSABLE types. In the subfunction, copy the value into the local variable.

Not clear at all what to do with bare FOR or SECTION blocks. The only thing I can figure is that we do something like

```
#pragma omp for firstprivate(x) lastprivate(y)
for (int i = 0; i < n; ++i)
  body;
```

which becomes

```
{
  int x = x, y;

  // for stuff

  if (i == n)
    y = y;
}
```

where the "x=x" and "y=y" assignments actually have different uids for the two variables, i.e. not something you could write directly in C. Presumably this only makes sense if the "outer" x and y are global variables.

COPYPRIVATE would work the same way, except the structure broadcast would have to happen via SINGLE machinery instead.

4.9 Implementing REDUCTION clause

The private struct mentioned in the previous section should have a pointer to an array of the type of the variable, indexed by the thread's *team_id*. The thread stores its final value into the array, and after the barrier the master thread iterates over the array to collect the values.

4.10 Implementing PARALLEL construct

```
#pragma omp parallel
{
  body;
}
```

becomes

```
void subfunction (void *data)
{
```

```
  use data;
  body;
}

setup data;
GOMP_parallel_start (subfunction, &data, num_threads);
subfunction (&data);
GOMP_parallel_end ();
```

```
void GOMP_parallel_start (void (*fn)(void *), void *data, unsigned num_threads)
```

The *FN* argument is the subfunction to be run in parallel.

The *DATA* argument is a pointer to a structure used to communicate data in and out of the subfunction, as discussed above with respect to FIRSTPRIVATE et al.

The *NUM_THREADS* argument is 1 if an IF clause is present and false, or the value of the NUM_THREADS clause, if present, or 0.

The function needs to create the appropriate number of threads and/or launch them from the dock. It needs to create the team structure and assign team ids.

```
void GOMP_parallel_end (void)
```

Tears down the team and returns us to the previous `omp_in_parallel()` state.

4.11 Implementing FOR construct

```
#pragma omp parallel for
for (i = lb; i <= ub; i++)
  body;
```

becomes

```
void subfunction (void *data)
{
  long _s0, _e0;
  while (GOMP_loop_static_next (&_s0, &_e0))
  {
    long _e1 = _e0, i;
    for (i = _s0; i < _e1; i++)
      body;
  }
  GOMP_loop_end_nowait ();
}

GOMP_parallel_loop_static (subfunction, NULL, 0, lb, ub+1, 1, 0);
subfunction (NULL);
GOMP_parallel_end ();
```

```
#pragma omp for schedule(runtime)
for (i = 0; i < n; i++)
  body;
```

becomes

```
{
  long i, _s0, _e0;
  if (GOMP_loop_runtime_start (0, n, 1, &_s0, &_e0))
    do {
      long _e1 = _e0;
      for (i = _s0, i < _e0; i++)
        body;
    } while (GOMP_loop_runtime_next (&_s0, _&e0));
```

```
  GOMP_loop_end ();
}
```

Note that while it looks like there is trickyness to propagating a non-constant STEP, there isn't really. We're explicitly allowed to evaluate it as many times as we want, and any variables involved should automatically be handled as PRIVATE or SHARED like any other variables. So the expression should remain evaluable in the subfunction. We can also pull it into a local variable if we like, but since its supposed to remain unchanged, we can also not if we like.

If we have SCHEDULE(STATIC), and no ORDERED, then we ought to be able to get away with no work-sharing context at all, since we can simply perform the arithmetic directly in each thread to divide up the iterations. Which would mean that we wouldn't need to call any of these routines.

There are separate routines for handling loops with an ORDERED clause. Bookkeeping for that is non-trivial...

4.12 Implementing ORDERED construct

```
void GOMP_ordered_start (void)
void GOMP_ordered_end (void)
```

4.13 Implementing SECTIONS construct

A block as

```
#pragma omp sections
{
  #pragma omp section
  stmt1;
  #pragma omp section
  stmt2;
  #pragma omp section
  stmt3;
}
```

becomes

```
for (i = GOMP_sections_start (3); i != 0; i = GOMP_sections_next ())
  switch (i)
    {
    case 1:
      stmt1;
      break;
    case 2:
      stmt2;
      break;
    case 3:
      stmt3;
      break;
    }
GOMP_barrier ();
```

4.14 Implementing SINGLE construct

A block like

```
#pragma omp single
{
  body;
}
```

becomes

```
if (GOMP_single_start ())
  body;
GOMP_barrier ();
```

while

```
#pragma omp single copyprivate(x)
  body;
```

becomes

```
datap = GOMP_single_copy_start ();
if (datap == NULL)
  {
    body;
    data.x = x;
    GOMP_single_copy_end (&data);
  }
else
  x = datap->x;
GOMP_barrier ();
```

5 Reporting Bugs

Bugs in the GNU OpenMP implementation should be reported via bugzilla. In all cases, please add "openmp" to the keywords field in the bug report.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

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

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

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

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

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

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

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

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

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

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

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

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

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

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

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

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

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

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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

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

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year  name of author

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

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

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
```

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

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

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

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

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

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

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

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

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

Funding Free Software

If you want to have more free software a few years from now, it makes sense for you to help encourage people to contribute funds for its development. The most effective approach known is to encourage commercial redistributors to donate.

Users of free software systems can boost the pace of development by encouraging for-a-fee distributors to donate part of their selling price to free software developers—the Free Software Foundation, and others.

The way to convince distributors to do this is to demand it and expect it from them. So when you compare distributors, judge them partly by how much they give to free software development. Show distributors they must compete to be the one who gives the most.

To make this approach work, you must insist on numbers that you can compare, such as, "We will donate ten dollars to the Frobnitz project for each disk sold." Don't be satisfied with a vague promise, such as "A portion of the profits are donated," since it doesn't give a basis for comparison.

Even a precise fraction "of the profits from this disk" is not very meaningful, since creative accounting and unrelated business decisions can greatly alter what fraction of the sales price counts as profit. If the price you pay is $50, ten percent of the profit is probably less than a dollar; it might be a few cents, or nothing at all.

Some redistributors do development work themselves. This is useful too; but to keep everyone honest, you need to inquire how much they do, and what kind. Some kinds of development make much more long-term difference than others. For example, maintaining a separate version of a program contributes very little; maintaining the standard version of a program for the whole community contributes much. Easy new ports contribute little, since someone else would surely do them; difficult ports such as adding a new CPU to the GNU Compiler Collection contribute more; major new features or packages contribute the most.

By establishing the idea that supporting further development is "the proper thing to do" when distributing free software for a fee, we can assure a steady flow of resources into making more free software.

Index