GNU GLOBAL Source Code Tag System

Edition 5.4, for GNU GLOBAL version 5.4
29 January 2007

by Tama Communications Corporation

This manual is for GNU GLOBAL (version 5.4, 29 January 2007), a source code tag system that works the same way across diverse environments.

Published by Tama Communications Corporation
Tama-shi, Tokyo, Japan.

1 Overview of the tools

1.1 What is this?

GNU GLOBAL is a source code tag system that works the same way across diverse environments. You can locate specified object in source files and move there easily. It is useful for hacking a large project containing many subdirectories, many `#ifdef` and many `main()` functions.

It is similar to ctags or etags but is different from them at the point of independence of any editor.

1.2 Concept of project.

GNU GLOBAL can treat a source tree containing subdirectories as a project. It is similar to CVS. You need not specify where the tag file is. Instead, global(1) will locate the tag file by itself. If tag file isn't found in the current directory, global(1) search parent directories for tag file.

User's position (current directory) is the first argument for GLOBAL's command.

1.3 Features.

GNU GLOBAL has following features:

- support C, C++, Yacc, Java, PHP4 and assembly.
- work the same way across diverse environments. Currently, support followings:
  - Shell command line
  - Bash shell.
  - Vi editor clone (nvi, elvis, vim)
  - Less viewer
  - Emacs editor (emacs, mule, xemacs)
  - Glimmer editor
  - Web browser
  - Doxygen documentation system
- find the locations of specified object quickly.
- locate not only object definitions but also object references.
- allows duplicate objects.
- locate also path which includes specified pattern.
- search not only in a source tree but also in library paths.
- understand POSIX 1003.2 regular expressions.
- support external search engine (grep and idutils).
- generate hypertext of source code.
- tag files are independent of machine architecture.
- plugin parser is available to treat new language.

- support incremental updating of tag files.
- support customizing with gtags.conf.
- generate completion list for completing input method.

2 Command line GLOBAL

You can use tag facilities from shell command line. It is a big merit of GLOBAL compared with any other tag system.

2.1 Preparation.

First of all, you must execute gtags(1)(see Section 5.2 [gtags], page 37) at the root of source tree. For example, if you want to browse vi's source code:

```
$ cd /usr/src/usr.bin/vi
$ gtags
```

Gtags traverse subdirectories and makes four databases at the root of the source tree.

```
$ ls G*
GPATH   GRTAGS  GSYMS   GTAGS
```

- 'GTAGS' database of object definitions
- 'GRTAGS' database of object references
- 'GSYMS' database of other symbols
- 'GPATH' database of path names

You should prepare for considerable disk space for tag files. For example, FreeBSD 7.0 kernel source code requires:

```
source code(/usr/src/sys)       123MB

GPATH                             1MB
GTAGS                            24MB
GRTAGS                           28MB
GSYMS                            32MB
-------------------------------------
total of tag files               85MB
```

2.2 Basic usage.

Consider the following source tree:

```
ROOT/           <- the root of source tree (GTAGS,GRTAGS,...)
|
|- DIR1/
|   |
|   |- fileA.c   .....  +---------------+
|   |                   |main(){        |
|   |                   |       func1();|
|   |                   |       func2();|
|   |                   |}              |
|   |                   +---------------+
|   |
|   |- fileB.c   .....  +---------------+
|                       |func1(){ ... } |
|                       +---------------+
|- DIR2/
    |
    |- fileC.c   .....  +---------------+
                        |#ifdef X       |
                        |func2(){ i++; }|
                        |#else          |
                        |func2(){ i--; }|
                        |#endif         |
                        |func3(){       |
                        |       func1();|
                        |}              |
                        +---------------+
```

- You can get the relative path of your object from anywhere in the source tree. You need not specify where the tag file is. Global will locate the tag file by itself.

```
$ cd ROOT
$ global func1
DIR1/fileB.c            # func1() is defined in fileB.c
$ cd DIR1
$ global func1
fileB.c                 # relative path from DIR1
$ cd ../DIR2
$ global func1
../DIR1/fileB.c         # relative path from DIR2
```

- The '-r' option locates object references.

```
$ global -r func2
../DIR1/fileA.c           # func2() is referred from fileA.c
```

- You can use POSIX regular expressions.

```
$ cd ROOT
$ global 'func[1-3]'
DIR1/fileB.c              # func1, func2 and func3 are matched
DIR2/fileC.c
```

- The '-x' option shows the details. It is similar to the '-x' option in ctags(1).

```
$ global func2
DIR2/fileC.c
$ global -x func2
func2              2 DIR2/fileC.c       func2(){ i++; }
func2              4 DIR2/fileC.c       func2(){ i--; }
```

- The '-a' option produces the absolute path name.

```
$ global -a func1
/home/user/ROOT/DIR1/fileB.c
```

- The -s command locates any symbols which are not defined in 'GTAGS'.

```
$ global -xs X
X                  1 DIR2/fileC.c #ifdef X
```

- The -g command locates any patterns including symbols. It is similar to grep(1).

```
$ global -xg '#ifdef'
#ifdef             1 DIR2/fileC.c #ifdef X
```

- The -P command enables you to locate path which includes specified string.

```
$ global -P fileB
DIR1/fileB.c
$ global -P '1/'
DIR1/fileA.c
DIR1/fileB.c
$ global -P '\.c$'
DIR1/fileA.c
DIR1/fileB.c
DIR2/fileC.c
```

- The -f command enables you see the list of objects of specified file.

```
$ global -f DIR2/fileC.c
func2              2 DIR2/fileC.c   func2(){ i++; }
func2              4 DIR2/fileC.c   func2(){ i--; }
func3              6 DIR2/fileC.c   func3(){
```

2.3 Applied usage.

You can make multiple tag files. For example, you can execute gtags at ROOT/, version1.0/ and version2.0/.

```
ROOT/                   <- the root of source tree      (GTAGS,...)
|
|- version1.0/          <- the root of version1.0       (GTAGS,...)
|   |
|   |- file.c    .....  +---------------+
|                       |func1(){ i++; }|
|                       +---------------+
|
|- version2.0/          <- the root of version2.0       (GTAGS,...)
    |
    |- file.c    .....  +---------------+
                        |func1(){ i--; }|
                        +---------------+
```

- When you are in the version1.0 directory, global will only locate objects that are in version1.0.

```
$ cd ROOT/version1.0
$ global -x func1
func1              1 file.c          func1(){ i++; }
```

- When you are in the version2.0, global will only locate objects that are in version2.0.

```
$ cd ROOT/version2.0
$ global -x func1
func1              1 file.c          func1(){ i--; }
```

- If you are at ROOT/, or you set the `GTAGSROOT` environment variable to ROOT, then global will locate objects in both directories.

```
$ cd ROOT
$ global -x func1
func1              1 version1.0/file.c  func1(){ i++; }
func1              1 version2.0/file.c  func1(){ i--; }
```

There is another usage of `GTAGSROOT`.

- If your source files are on a read-only device, such as CDROM, then you cannot make databases at the root of the source tree. In such cases, please do the following:

```
$ mkdir /var/dbpath
$ cd /cdrom/src                 # the root of source tree
$ gtags /var/dbpath             # make tag file in /var/dbpath
$ export GTAGSROOT=`pwd`
$ export GTAGSDBPATH=/var/dbpath
$ global func
```

- If you want all references to an object that is not defined in the source tree to be treated as calls to library functions or system calls, then you can specify library directories with the GTAGSLIBPATH environment variable.

  You should execute gtags at each directory of the path. If 'GTAGS' is not found in a directory, global ignores that directory.

```
$ pwd
/develop/src/mh                         # this is the source tree
$ gtags
$ ls G*TAGS
GRTAGS  GTAGS
$ global mhl
uip/mhlsbr.c                            # mhl() is found
$ global strlen                         # strlen() is not found
$ (cd /usr/src/lib; gtags)              # library source
$ (cd /usr/src/sys; gtags)              # kernel source
$ export GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
$ global strlen
../../../usr/src/lib/libc/string/strlen.c  # found in library
$ global access
../../../usr/src/sys/kern/vfs_syscalls.c   # found in kernel
```

  Of course, the user program does not call kernel functions directly, but at least it is useful.

- If you forget a object name, you can use the -c (complete) command.

```
$ global -c kmem                # maybe k..k.. kmem..
kmem_alloc
kmem_alloc_pageable
kmem_alloc_wait
kmem_free
kmem_free_wakeup
kmem_init
kmem_malloc
kmem_suballoc                   # This is what I need!
$ global kmem_suballoc
../vm/vm_kern.c
```

- You can use the -c command with complete command in shell.
  In bash:

```
$ funcs()
> {
>         local cur
>         cur=${COMP_WORDS[COMP_CWORD]}
>         COMPREPLY=(`global -c $cur`)
> }
$ complete -F funcs global
$ global kmem_TABTAB
kmem_alloc           kmem_alloc_wait      kmem_init
kmem_alloc_nofault   kmem_free            kmem_malloc
kmem_alloc_pageable  kmem_free_wakeup     kmem_suballoc
$ global kmem_sTAB
$ global kmem_suballoc
../vm/vm_kern.c
```

  In tcsh:

```
% set funcs=(`global -c`)
% complete global 'n/*/$funcs/'
% global kmem_TAB
kmem_alloc          kmem_free_wakeup
kmem_alloc_pageable kmem_init
kmem_alloc_wait     kmem_malloc
kmem_free           kmem_suballoc
% global kmem_sTAB
% global kmem_suballoc
../vm/vm_kern.c
```

- You can edit all files that include specified object by typing one command, for example:

```
$ vi `global func1`     # edit fileB.c
```

- If you want to browse many files in order, do the following:

```
$ global -xr fork | awk '{printf "view +%s %s\n",$2,$3}'
view +650 ../dev/aic7xxx/aic7xxx_asm.c
view +250 ibcs2/ibcs2_misc.c
view +401 linux/linux_misc.c
view +310 ../kern/init_main.c
view +318 ../kern/init_main.c
view +336 ../kern/init_main.c
view +351 ../kern/init_main.c
```

```
$ !! | sh                # from now on, go to next tag with 'ZZ'.
```

3 Various applications

3.1 Global facility for Bash

Special support for bash is available.

3.1.1 Features.

- Vi-like tag stack is available.
- Emacs-like tag name completion is available.
- Editor or viewer is automatically invoked.
- Tag mark facility is available.
- Yor can manage directory list by cookie facility.

3.1.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3. And you can invoke bash(1) with —rcfile option.

```
$ bash --rcfile /usr/local/share/gtags/globash.rc
```

You will see a prompt like this:

```
[/usr/src/sys]/kern _
```

This prompt means that the current directory is '/usr/src/sys/kern' and the root of the source tree is '/usr/src/sys'. Tag and marker are valid only in a project.

When you get out of the project, globash warns like:

```
[/usr/src/sys] cd ..
You are going to get out of current project.
Tag stack and marker will be removed. Sure? ([y]/n)_
```

If you answer 'y' and *RET* or just *RET* in above example then tag stack and marker will be removed.

If you need help then please type 'ghelp'.

3.1.3 Basic usage.

- Almost global(1)(see Section 5.1 [global], page 33)'s command character is available as a command.

```
[/usr/src/sys] x fork              <- (global -x fork)
>    1  fork             94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] r                   <- (global -xr fork)
>    1  fork             85 alpha/linux/linux_machdep.c
```

```
     2  fork            184 i386/linux/linux_machdep.c
[/usr/src/sys] s lbolt             <- (global -xs lbolt)
>    1  lbolt           1210 i386/isa/wd_cd.c     tsleep((cad
     2  lbolt           1211 i386/isa/wd_cd.c     tsleep((cad
     3  lbolt           709 i386/isa/wfd.c     tsleep ((caddr
...
[/usr/src/sys] g                   <- (global -xg lbolt)
>    1  lbolt           1210 i386/isa/wd_cd.c     tsleep((cad
...
[/usr/src/sys] P init              <- (global -xP init)
>    1  path    1 dev/hea/eni_init.c
     2  path    1 dev/hfa/fore_init.c
     3  path    1 i386/i386/initcpu.c
     4  path    1 kern/init_main.c
     5  path    1 kern/init_sysent.c
     6  path    1 kern/vfs_init.c
     7  path    1 vm/vm_init.c
[/usr/src/sys] _
```

If no tag name is specified then it is assumed the latest tag name.

- You can select a tag by show command.

```
[/usr/src/sys] x main
>    1  main             70 alpha/alpha/gensetdefs.c main(in
     2  main            1500 alpha/alpha/ieee_float.c main(i
     3  main            227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] show 3
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
```

The default editor is vi(1) but you can specify it statically by `EDITOR` environment variable or temporarily by option.

```
[/usr/src/sys] show -e 3
(Preloaded emacs show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] show -l 3
(Load less and show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] show -g 3
(Preloaded mozilla show boot/alpha/boot1/boot1.c at line 227.)
```

- You can use vi-like tag stack. You can return previous tag list by pop or *CTL-T* command.

```
[/usr/src/sys] x main
>    1  main             70 alpha/alpha/gensetdefs.c main(in
```

```
     2  main            1500 alpha/alpha/ieee_float.c main(i
     3  main             227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] show 3
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
[/usr/src/sys] x fork          <- push new tag on tag stack.
>    1  fork              94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] pop             <- pop tag stack.
[/usr/src/sys] show
(Load editor and show boot/alpha/boot1/boot1.c at line 227.)
```

3.1.4 Applied usage.

- You can memory tags using 'mark' command.

```
[/usr/src/sys] x fork
>    1  fork              94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] mark
[/usr/src/sys] x main
>    1  main              70 alpha/alpha/gensetdefs.c main(in
     2  main            1500 alpha/alpha/ieee_float.c main(i
     3  main             227 boot/alpha/boot1/boot1.c main()
....
[/usr/src/sys] mark -l              <- show marker list.
     1  fork              94 kern/kern_fork.c fork(p, uap)
[/usr/src/sys] mark 1               <- select a marker.
(Load editor and show kern/kern_fork.c at line 227.)
[/usr/src/sys] list
>    1  main              70 alpha/alpha/gensetdefs.c main(in
     2  main            1500 alpha/alpha/ieee_float.c main(i
     3  main             227 boot/alpha/boot1/boot1.c main()
....
```

  Marked tags are valid until you get out of current project or quit current bash session.

- You can memory directories using 'cookie' command.

```
[/usr/src/sys] cookie               <- drop cookie.
[/usr/src/sys] cd kern
[/usr/src/sys]/kern cookie          <- drop cookie again.
[/usr/src/sys]/kern cd ../i386
[/usr/src/sys]/i386 cookie -l       <- show cookie list.
     1  /usr/src/sys/kern
     2  /usr/src/sys
[/usr/src/sys]/i386 warp 2          <- warp to selected cookie.
[/usr/src/sys] _
```

  Cookie directories are valid until you delete them.

3.2 Less using GLOBAL.

You can use GLOBAL as a tag system of less instead of ctags.

3.2.1 Features.

- You can use most of GLOBAL's facilities from less-370 or the later.
- Less viewer support duplicated tag.

3.2.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from less, you need to set environment variable `LESSGLOBALTAGS` to "global".

```
$ export LESSGLOBALTAGS=global
```

3.2.3 Basic usage.

- To go to func1, you can say

  ```
  $ less -t func1
  ```

  Please note that if '`tags`' exist then less use it. If you want to use '`GTAGS`' even if '`tags`' exist then please specify tag file explicitly like this.

  ```
  $ less -TGTAGS -t func1
  ```

- To go to the referenced point of func1, please specify '`GRTAGS`'.

  ```
  $ less -TGRTAGS -t func1
  ```

  In the same way, you can use '`GTAGS`', '`GRTAGS`', '`GSYMS`', '`GPATH`' as tag file.
- If a number of objects are located, less goes to the first tag. You can go to next tag by typing *`t`* and back by typing *`T`*.

  `t` go to next tag.

  `T` go to previous tag.
- From less session, you can use *`:t`* command to locate new symbol. But in this case, you cannot change tag file from one specified by `-`*`T`* option.

3.2.4 Applied usage.

- With `-`*`T`*`-` option, less read standard input as tag file. It is very valuable. You can connect global and less with pipe line.

  ```
  $ global -x func | less -T-
  ```

In the same way, you can use following command lines.

```
# pattern match with grep(1).
$ global -xg 'lseek(.*)' | less -T-

# pattern match with idutils(1).
$ global -xI func | less -T-

# all objects definitions in *.c.
$ global -f *.c | less -T-

# all files includes 'init' in its path.
$ global -Px init | less -T-
```

- If your editor doesn't support GLOBAL directly then you can use less as a footstool.

```
# invoke less
$ less -t main
main(int argc, char **argv)
{
int i;
.....
[xxx/main.c (tag 1 of 55)]

# type 'v'(vi) command in less session.
v

# load vi and show the same position.
.....
main((int argc, char **argv)
{
int i;
.....
[xxx/main.c 313 lines, 7783 char]

# type 'ZZ' command in vi session.
ZZ

# exit vi and back to less session.
main(int argc, char **argv)
{
int i;
.....
[xxx/main.c (tag 1 of 55)]
```

3.3 Extended nvi-1.79 using GLOBAL.

You can use GLOBAL as a tag system of Nvi editor instead of ctags.

3.3.1 Features.

- You can use most of GLOBAL's facilities from the editor.
- Recognition of the current token and its type.
- Extended nvi is completely backward-compatible with the original nvi. You can use GLOBAL's facilities only in 'gtags mode'.

3.3.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from nvi, you need to get into *gtagsmode*. There are several ways to do this:

1. Start nvi with '`-G`' option

```
$ nvi -G file.c
```

2. Start nvi and execute `set gtagsmode`.

```
$ nvi file.c
~
~
~
:set gtagsmode
```

3. Write the above set command to the '`.exrc`' or '`.nexrc`' and start nvi

```
$HOME/.exrc
+----------------------------
|set gtagsmode
```

You must start nvi under the source tree described in Section 2.1 [Preparation], page 3.

3.3.3 Basic usage.

- To go to func1, you can say

```
:tag func1
```

  It seemes the same as original nvi, but extended nvi use '`GTAGS`' instead of '`tags`'.
- To go to the referenced point of func1, add the option *`-r`*

```
:tag -r func1
```

  Extended nvi use '`GRTAGS`'.
- If a number of objects are located, extended nvi goes to the first tag. You can go to next tag by typing *`:tagnext`* and back by typing *`:tagprev`*.

```
Suggested .nexrc:
set gtagsmode
map ^N :tagnext^M
map ^P :tagprev^M
```

- *CTL-]* command is available. In gtags mode, if current token is not a function then it is equivalent to *:tag -s current token*. Otherwise, if you are in the first column of a line, it is equivalent to *:tag -r current token* else it is equivalent to *:tag current token*.
- You can use the *-s* option. It locates any symbols which are not defined in 'GTAGS'.

  ```
  :tag -s pat
  ```

  Extended nvi use 'GSYMS'.
- The *-g*, *-f* and *-P* option are also available. It works like command line. Extended nvi use no index file.

  ```
  :tag -g pat
  ```

- Other tag commands are also available:

  CTL-T Return to the most recent tag context.

  :tagpop Go to the specified tag in the tags stack.

  :tagtop Go to the top tag in the tags stack.

  :display tags
  Display the tags stack.

3.3.4 Applied usage.

- In large projects that include many main() function like MH, you can start nvi like this:

  ```
  $ nvi -G -t main
  ```

  You can browse all commands sequentially.
- When you want to check objects the name of which start with "set" or "get", use:

  ```
  $ nvi -G -t '^[sg]et'
  ```

  Of course, the following command is also available:

  ```
  :tag ^[sg]et
  ```

- If your source files are on a read only device like a CD-ROM, please do the following:

```
$ mkdir /var/dbpath              # directory for the tag file
$ cd /cdrom/src                  # the root of the source tree
$ gtags /var/dbpath              # make tag files in /var/dbpath
$ export GTAGSROOT=`pwd`
$ export GTAGSDBPATH=/var/dbpath
$ nvi -G -t main
```

- If you want all references to an object that is not defined in the source tree to be treated as references to library functions or as system calls, do the following:

```
$ cd /usr/src/lib
$ gtags                          # probably as a root
$ cd /usr/src/sys
$ gtags
$ export GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
```

- If you examine vi's source,

```
$ cd /usr/src/usr.bin/vi
$ gtags
$ nvi -G -t main
```

You can start from nvi and browse the whole unix world as if you were using hypertext.

3.4 nvi-1.81.5 using GLOBAL.

You can use GLOBAL as a tag system of Nvi editor instead of ctags.

3.4.1 Features.

- You can use most of GLOBAL's facilities from the editor.
- Recognition of the current token and its type.

3.4.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from nvi, you need write to '`.nexrc`' like this: It assumed that gtags.pl is put on '`$HOME/perl`'.

```
$HOME/.nexrc
+----------------------------
|perl use lib "$ENV{'HOME'}/perl"
|perl require 'gtags.pl'
|map ^P :tagprev^M
|map ^N :tagnext^M
```

```
|map ^] :perl tag^M
|ab gtag perl tag qw(
|ab gta perl tag qw(
|ab gt perl tag qw(
```

You must start nvi under the source tree described in Section 2.1 [Preparation], page 3.

3.4.3 Basic usage.

- To go to func1, you can say

  ```
  :perl tag qw(func1)

  Suggested .nexrc:
  ab gtag perl tag qw(
  ab gta perl tag qw(
  ab gt perl tag qw(
  ```

- To go to the referenced point of func1, add the option *-r*

  ```
  :perl tag qw(-r func1)
  ```

- If a number of objects are located, nvi goes to the first tag. You can go to next tag by typing *:tagnext* and back by typing *:tagprev*.

  ```
  Suggested .nexrc:
  map ^N :tagnext^M
  map ^P :tagprev^M
  ```

- If you don't specify any argument. ':perl tag' command do the followins: If current token is not a function then it is equivalent to *:perl tag qw(-s current token)*. Otherwise, if you are in the first column of a line, it is equivalent to *:perl tag qw(-r current token)* else it is equivalent to *:perl tag qw(current token)*.

  ```
  Suggested .nexrc:
  map ^] :perl tag^M
  ```

  It is similar to *CTL-]* command.

- You can use the *-s* option. It locates any symbols which are not defined in 'GTAGS'.

  ```
  :perl tag qw(-s pat)
  ```

- The *-g*, *-f* and *-P* option are also available. It works like command line.

  ```
  :perl tag qw(-g pat)
  ```

- When you want to check objects the name of which start with "set" or "get", use:

```
:perl tag qw(^[sg]et)
```

- Other tag commands are also available:

  CTL-T Return to the most recent tag context.

  :tagpop Go to the specified tag in the tags stack.

  :tagtop Go to the top tag in the tags stack.

  :display tags
  Display the tags stack.

3.5 Elvis using global

Elvis 2.1 has new `tagprg` and `tagprgonce` variables for running an external tag search program. You can use them with GLOBAL.

3.5.1 Features.

- You can use most of GLOBAL's facilities from the editor.
- No source level patch is needed.
- Mouse events are supported.

3.5.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, start elvis and execute `set tagprg="global -t $1"` like this.

```
$ elvis
~
~
~
~
~
~
:set tagprg="global -t $1"
```

3.5.3 Basic usage.

- To go to func1, you can say

  ```
  :tag func1
  ```

  It seemes the same as original elvis, but elvis execute `global -t func1` internally and read it instead of tags file.
- To go to the referenced point of func1, add '`-r`' option.

```
:tag -r func1
```

Elvis executes command like `global -t -r func1` internally.

- To go to any symbols which are not defined in 'GTAGS', try this.

```
:tag -s lbolt
```

- To go to any strings other than symbols, try this.

```
:tag -g Copyright
```

- When using -r, -s or -g, you had better to use browse command.

```
:browse -r fork
```

It brings a following selection list. You can select tag and go to the point.

```
Browse -r fork (2 matches)
+----------------+----------------+--------------------
| TAG NAME       | SOURCE FILE    | SOURCE LINE
+----------------+----------------+--------------------
|fork            |ux/linux_misc.c | (line 565)
|fork            |ern/init_main.c | (line 191)
+----------------+----------------+--------------------
```

- To get list of objects in a file, use -f command.

```
:browse -f main.c                <- locate definitions in main.c
```

- Other tag commands are also available:

CTL-] go to the definition of current token.

CTL-T return to the most recent tag context.

:tag without argument, go to the next tag.

:pop return to the most recent tag context.

:stack display the tags stack.

:stag creates a new window and moves its cursor to the tag's definition point.

:sbrowse same with 'browse' but show in a new window.

3.5.4 Applied usage.

- You can use POSIX regular expressions.

```
:tag ^put_                    <- locate objects start with 'put_'

:browse -g 'fseek(.*L_SET)' <- locate fseek() using L_SET argument
```

- You can browse object list of many files.

```
:browse -f *.c                <- locate objects in *.c
```

- You can browse project files whose path includs specified pattern.

```
:browse -P /vm/               <- under vm/ directory
:browse -P \.h$               <- all include files
:browse -P init               <- path including 'init'
```

- You can use mouse to select tag.

```
+----------------+----------------+--------------------
| TAG NAME       | SOURCE FILE    | SOURCE LINE
+----------------+----------------+--------------------
|fork            |ux/linux_misc.c | (line 565)
|fork            |ern/init_main.c | (line 191)
+----------------+----------------+--------------------
```

Please select tag name with mouse cursor and double click on the left button and you go to the tag's point. In source screen, also select an object name and double click on the left button and you can go to the point that the object is defined. To come back, double click on the right button.

3.6 Vim using global

In vim 6.2 or later, you can use gtags.vim script.

3.6.1 Features.

- You can use most of GLOBAL's facilities from the editor.
- Intelligent recognition of the current token and its type.
- Special character '%', '#' and input completion are available.

3.6.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, copy 'gtags.vim' to your plugin directory or source it from your vimrc.

```
$ cp /usr/local/share/gtags/gtags.vim $HOME/.vim/plugin
```

3.6.3 Basic usage.

- To go to main, you can say

  ```
  :Gtags main
  ```

  Vim execute `global -t main`, parse the output, list located objects in quickfix window and load the first entry. The quickfix windows is like this:

  ```
  gctags/gctags.c|119| main
  global/global.c|154| main
  gozilla/gozilla.c|156| main
  gtags/gtags.c|199| main
  libglibc/getopt.c|701| main
  libglibc/getopt1.c|93| main
  [Error List]
  ```

  You can go to any entry using quickfix command.

  :cn go to the next entry.

  :cp go to the previous entry.

  :ccN go to the N'th entry.

  :cl list all entries.

  You can see the help of quickfix like this:

  ```
  :h quickfix
  ```

- To go to the referenced point of func1, add '`-r`' option.

  ```
  :Gtags -r func1
  ```

  vim executes command like `global -t -r func1` internally.
- To go to any symbols which are not defined in '`GTAGS`', try this.

  ```
  :Gtags -s lbolt
  ```

- To go to any strings other than symbols, try this.

  ```
  :Gtags -g Copyright
  ```

- To get list of objects in a file, use -f command.

```
:Gtags -f main.c                 <- locate objects in main.c
```

If you are editing 'main.c' itself, you can use '%' instead.

```
:Gtags -f %                      <- locate objects in main.c
```

3.6.4 Applied usage.

- You can use POSIX regular expressions.

```
:Gtags ^put_                  <- locate objects start with 'put_'

:Gtags -g fseek(.*SEEK_SET) <- locate fseek() using SEEK_SET
```

- Input completion is available.

  :Gtags fu*TAB*

  :Gtags func1 <- 'nc1' is appended by vim
- You can browse project files whose path includes specified pattern.

```
:Gtags -P /vm/              <- under vm/ directory
:Gtags -P \.h$              <- all include files
:Gtags -P init              <- path including 'init'
```

- You can use all options of global(1) except for the -c, -p, -u and all long name options. They are sent to global(1) as is. For example,

```
:Gtags -gi paTtern          <- match to both 'PATTERN' and 'pattern'.
```

  About the other options, See Section 5.1 [global], page 33.
- The GtagsCursor command brings you to the definition or reference of the current token in C language. The GtagsCursor is not perfect though is considerably wise. If current token is not a function then it is equivalent to *:Gtags -s current token*. Otherwise, if you are in the first column of a line, it is equivalent to *:Gtags -r current token* else it is equivalent to *:Gtags current token*.

```
:GtagsCursor

Suggested map:
map <C-]> :GtagsCursor<CR>
```

- If you have the hypertext generated by htags(1) then you can display the same place on mozilla browser. Let's load mozilla and try this:

```
:Gozilla

Suggested map:
map <C-g> :Gozilla<CR>
```

- If you want to load vim with all main()s then following command line is useful.

```
$ vim '+Gtags main'
```

3.7 Extended emacs using global

You can use GLOBAL as a tag system of Emacs editor instead of etags.

3.7.1 Features.

- You can use most of GLOBAL's facilities from the editor.
- More intelligent recognition of the current token and its type.
- Tag completion is available for input tag name.
- Mouse events are supported.

3.7.2 Preparation.

First, do the preparation of global. See Section 2.1 [Preparation], page 3.

Second, to use global from emacs, you need to load the '`gtags.el`' and execute gtags-mode function in it.

1. Write the autoload function to the '`$HOME/.emacs`', start emacs and execute the gtags-mode function. If you don't put '`gtags.el`' in standard macro directory, you need to add the directory to `load-path`.

```
$HOME/.emacs
+------------------------------------------------------
|(setq load-path (cons "/home/owner/global" load-path))
|(autoload 'gtags-mode "gtags" "" t)

$ emacs

|
|J_:-----Mule: *scratch*       (Lisp Interaction)--L16--All----
|M-x gtags-mode[RET]
+------------------------------------------------------
```

   If you want to get into gtags-mode on c-mode then you can append followings into the '`$HOME/.emacs`'.

```
  (setq c-mode-hook
      '(lambda ()
          (gtags-mode 1)
```

```
))
```

2. Specify the root directory of the source tree using `gtags-visit-rootdir`. If you have tag files in /usr/src/sys then please do like this:

```
Visit root directory: /usr/src/sys
```

3.7.3 Basic usage.

- To go to func1, invoke `gtags-find-tag` and you can see a prompt in mini-buffer. Then input the tag name.

  ```
  Find tag: func1 <- 'Find tag: ' is a prompt
  ```

- To go to a point that references func1, invoke `gtags-find-rtag`.

  ```
  Find tag (reference): func1
  ```

- Tag name completion is available. You need to execute `gtags-make-complete-list` command before it.

  ```
  Find tag: fuTAB

  Find tag: func1                         <- 'nc1' is appended by emacs
  ```

- If a number of objects are located, emacs goes into *GTAGS SELECT MODE* like this:

  ```
  +-------------------------------------------------------------
  |main            347 i386/isa/ultra14f.c main()
  |main            128 kern/init_main.c   main(framep)
  |main            104 netiso/clnp_debug.c main()
  |main            164 netiso/xebec/main.c main(argc, argv)
  |
  |
  |
  |
  |
  |J_:--%*-Mule: *scratch*       (Gtags Select)--L1--All----
  |[GTAGS SELECT MODE] 4 lines
  +-------------------------------------------------------------
  ```

  You can select a tag line by using any emacs command and pressing *RET*, and you can go to the tag's point. When you want to go to the next or the previous tag, you can return to 'GTAGS SELECT MODE' with `gtags-pop-stack` and reselect.

- `gtags-find-tag-from-here` command is available.

  If current token is a definition, it is equivalent to *`Find tag (reference): current token`RET*, otherwise it is equivalent to *`Find tag: current token`RET*. (This facility is supported only in C language. GLOBAL decides this intelligentlly, but may sometimes misunderstand.)
- To go to any symbols which are not defined in '`GTAGS`', try `gtags-find-symbol`.

  ```
  Find symbol: lbolt <- 'Find symbol:' is a prompt
  ```

- To go to any strings other than symbols, try `gtags-find-with-grep`.

  ```
  Find pattern: Copyright
  ```

3.7.4 Applied usage.

- You can use POSIX regular expressions.

  ```
  Find tag: ^put_                  <- locate tags start with 'put_'
  ```

- If your source files are on a read-only device like a CDROM, please do the following:

  ```
  $ mkdir /var/dbpath              # directory for the tag file
  $ cd /cdrom/src                  # the root of the source tree
  $ gtags /var/dbpath              # make tag files in /var/dbpath
  $ export GTAGSROOT=`pwd`
  $ export GTAGSDBPATH=/var/dbpath
  $ emacs -f gtags-mode
  ```

- If you want all references to an object that is not defined in the source tree to be treated as references to library functions or as system calls, do the following:

  ```
  $ cd /usr/src/lib
  $ gtags                          <- probably as a root
  $ cd /usr/src/sys
  $ gtags
  $ export GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
  $ emacs -f gtags-mode
  ```

- Mouse command is avalable.

  If you use X version emacs, try the following

  Move the mouse cursor to an object name and click the middle button. You will then go to the object's definition, or to its references, depending on the context. In 'GTAGS SELECT MODE', move the mouse cursor to a line and click the center button.

  To return to the previous position, click the right button.

3.8 Hypertext generator

You can use GLOBAL's facilities from WWW browser.

3.8.1 Features.

- Htags makes hypertext from C, C++, Yacc and Java source files.
- Once the hypertext is generated, you need nothing other than a WWW browser.
- You can move the hypertext to anywhere. It is independent of the source code.
- You can use all of your browser's functions, such as search, history, bookmark, save, frames, windows.

3.8.2 Preparation.

At first, you must ensure that you have a lot of disk space for hypertext. For example, FreeBSD 7.0 kernel source code (123MB) requires disk space from 600 to 1200MB.

```
source code(/usr/src/sys)            123MB
GPATH,GTAGS,GRTAGS,GSYMS              86MB

hypertext (with no option)           645MB
hypertext (with -s option)          1168MB
hypertext (with -D option)           383MB
hypertext (with -s and -D option)    616MB
```

Please invoke gtags(1)(see Section 5.2 [gtags], page 37) and htags(1)(see Section 5.3 [htags], page 40) in order like this:

```
(at your source directory)
$ gtags                  # make tag files(GTAGS,GRTAGS,GSYMS)
$ htags                  # make hypertext(HTML/)
```

Then you will find an 'HTML' subdirectory in the current directory. Htags has many options.

```
Suggested option:
$ htags -vsanohIT
```

If http server is available then the -D and -f option are also useful.

3.8.3 Usage.

Please start a web browser like this:

```
$ lynx HTML/index.html
```

You will understand the usage by looking at the examples.

You can move the HTML directory to anywhere. It is independent of the source code.

Using mozilla, you can also utilize hypertext from your command line like this:

```
$ mozilla # load mozilla
$ global -x main
main        10 main.c main(int argc, char *argv[]) {
$ gozilla +10 main.c # usage is similar to vi editor.
(show main.c at 10 on mozilla's screen.)
```

But in this case, you must not move HTML directory from the source directory.

3.9 Doxygen using global

You can use GLOBAL as a source browser of Doxygen.

Doxygen Release 1.4.3 or later includs config option USE_HTAGS. When enabled in combination with SOURCE_BROWSER=YES, htags(1) is used as the source browser instead of doxygen's own.

Here is an example.

```
(in source directory)
$ doxygen -g
$ vi Doxyfile
+---------------------------------
|...
|INPUT                  = .
|RECURSIVE              = YES
|SOURCE_BROWSER         = YES
|USE_HTAGS              = YES
|...

$ doxygen
$ lynx html/index.html
```

4 Other topics

4.1 How to config GLOBAL.

You can customize GLOBAL using configuration file.

```
# cp gtags.conf /etc/gtags.conf         # system wide config file.
# vi /etc/gtags.conf

$ cp gtags.conf $HOME/.globalrc         # personal config file.
$ vi $HOME/.globalrc
```

If '$HOME/.globalrc' exists then GLOBAL use it. Else if '/etc/gtags.conf' exists then GLOBAL use it. Otherwise default value will be used. The format of 'gtags.conf' is resemble to termcap(5). By default, 'default' target is used. About the capabilities, please see each command manual. See Chapter 5 [Reference], page 33.

4.2 Plug-in parser.

You can write new parser and use as a plugin parser.

4.2.1 How to plug in a parser.

Copy 'gtags.conf' to '/etc/gtags.conf' or '$HOME/.globalrc'.

If you would like to use exuberant ctags included by Vim editor,

```
$ cd /vim source directory/src/ctags
$ cp Makefile.unix Makefile
$ make
# cp ctags /usr/local/bin/ctags-exuberant
$ export GTAGSLABEL=ctags-exuberant     # see gtags.conf
$ gtags
$ ls G*
GPATH   GTAGS
```

'GRTAGS' and 'GSYMS' don't exist, simply because these parsers don't support the '-r' option and '-s' option like gtags-parser(1) does.

4.2.2 Requirement of plugin parser.

Plug-in parser must print tag information to standard output in the same style as `ctags -x`, ie.:

```
[1]    [2] [3]          [4]
----------------------------------------------------------------
main    20 ./main.c     main(argc, argv)        /* xxx */
```

```
[1] tag name
[2] line number the tag appeared
[3] path name. It must be equal to argument path name.
[4] line image
```

Plug-in parser must process the files in the order they are given in the argument. In each file, any order is acceptable.

- Good example

  The following `good-prog` does correct operation as a plugin parser.

```
$ good-prog a.c b.c          <= order: a.c -> b.c
            ~~~~~~~
main          25 a.c  main(int argc, char *argv[])
func          45 a.c  func(int a) {
sub2          20 b.c  sub2() {
sub1          10 b.c  sub1() {
                  ^
                  |
                 *** order: a.c -> b.c (Good!)
```

- Bad example

  The following `bad-prog` does wrong operation as a plugin parser.

```
$ bad-prog a.c b.c          <= order: a.c -> b.c
main          25 a.c  main(int argc, char *argv[])
sub2          20 b.c  sub2() {
sub1          10 b.c  sub1() {
func          45 a.c  func(int a) {
                  ^
                  |
                 *** order: b.c -> a.c (BAD!!!)
```

4.3 Incremental updating.

Modifying some source files, you need not remake whole tag files. Instead, you can use incremental updating facility ('-u' option).

```
$ gtags
$ cd kern
$ vi tty.c                                  # modify tty.c
...
:wq
$ global -vu                                # -v means verbose
[Sun Dec  6 16:27:47 JST 1998] Gtags started
 Tag found in '/usr/src/sys'.
 Incremental update.
[Sun Dec  6 16:28:48 JST 1998] Updating 'GTAGS'.
 [1/1] deleting tags of kern/tty.c
 [1/1] adding tags of kern/tty.c
[Sun Dec  6 16:28:59 JST 1998] Updating 'GRTAGS'.
 [1/1] deleting tags of kern/tty.c
 [1/1] adding tags of kern/tty.c
[Sun Dec  6 16:28:14 JST 1998] Updating 'GSYMS'.
 [1/1] deleting tags of kern/tty.c
 [1/1] adding tags of kern/tty.c
 Global databases have been modified.
[Sun Dec  6 16:28:30 JST 1998] Done.
$ global -vu                                # try again
[Sun Dec  6 16:28:48 JST 1998] Gtags started
 Tag found in '/usr/src/sys'.
 Incremental update.
 Global databases are up to date.           # do nothing
[Sun Dec  6 16:28:52 JST 1998] Done.
```

5 Command References

5.1 global - print the locations of specified object.

NAME

global - print the locations of specified object.

SYNOPSIS

global [-aGilnqrstTvx][-e] pattern
global -c[qsv] prefix
global -f[anqrstvx] files
global -g[aGilnoOqtvx][-e] pattern
global -I[ailnqtvx][-e] pattern
global -p[qrv]
global -P[aGilnoOqtvx][-e] pattern
global -u[qv]

DESCRIPTION

Global find the locations of specified object in C, C++, Yacc, Java, PHP and Assembly source files. Global can treat a source tree, that is, a directory that has subdirectories and source files. You can get the relative path of objects from anywhere within the tree. Global can locate not only object definitions but also object references and other symbols. Duplicate entries are allowed.

In advance of using this command, you must execute gtags(1) at the root directory of the source tree to make tag files. Then you can execute at anywhere in the source tree.

COMMANDS

The following commands are available:

pattern
: Print object which match to the pattern. Extended regular expressions which are the same as those accepted by egrep(1) are available.

'`-c`', '`--completion`' [prefix]
: Print candidate definition names which start with specified prefix. Prefix is not specified, print all definition names.

'`-f`', '`--file`' files
: Print all tags in the files. This option implies -x option.

'`-g`', '`--grep`'
: Print all lines which match to the pattern.

'`-I`', '`--idutils`'
: Print all lines which match to the pattern. This function use idutils(1) as a search engine. To use this command, you need to install idutils(1) in your system and you must execute gtags(1) with '`-I`' option.

'-p', '--print-dbpath'
: Print the location of 'GTAGS'.

'-P', '--path' [pattern]
: Print the path which match to the pattern. If no pattern specified, print all.

'-u', '--update'
: Locate tag files and update them incrementally.

'--version'
: Show version number.

'--help'
: Show help.

OPTIONS

The following options are available:

'-a', '--absolute'
: Print absolute path name. By default, print relative path name.

'-e', '--regexp' pattern
: Use pattern as the pattern; useful to protect patterns beginning with -.

'-G', '--basic-regexp'
: Interpret pattern as a basic regular expression. The default is extended regular expression. This option is valid for the '-g' and '-P' command.

'-i', '--ignore-case'
: ignore case distinctions in pattern.

'-l', '--local'
: Print just objects which exist under the current directory.

'-n', '--nofilter'
: Suppress sort filter and path conversion filter.

'-o', '--other'
: Search pattern in not only source files but also other files like 'README'. This option is valid only with '-g' or '-P' command.

'-O', '--only-other'
: Search pattern only in other files like 'README'. This option is valid only with '-g' or '-P' command. This option override the '-o' option.

'-q', '--quiet'
: Quiet mode.

'-r', '--reference', '--rootdir'
: Print the locations of object references. By default, print object definitions. With the '-p' option, print the root directory of source tree.

'--result' format
: format may be 'path', 'ctags', 'ctags-x', 'grep' or 'cscope'. The '--result=ctags' and '--result=ctags-x' are equivalent to the '-t' and '-x' respectively. The '-t' and '-x' are given to priority more than the '--result' option.

'`-s`', '`--symbol`'
: Print the locations of specified symbol other than definitions.

'`-t`', '`--tags`'
: Print with standard ctags format.

'`-T`', '`--through`'
: Go through all the tag files listed in *GTAGSLIBPATH*. By default, stop searching when tag is found. This option is ignored when either '`-s`', '`-r`' or '`-l`' option is specified.

'`-v`', '`--verbose`'
: Verbose mode.

'`-x`', '`--cxref`'
: In addition to the default output, produce the line number and the line contents.

EXAMPLES

```
$ ls -F
Makefile      src/    lib/
$ gtags
$ global main
src/main.c
$ global -x main
main              10 src/main.c  main (argc, argv) {
$ global -x '^[sg]et'
set_num           20 lib/util.c  set_num(values)
get_num           30 lib/util.c  get_num() {
$ global -rx '^[sg]et'
set_num          113 src/op.c            set_num(32);
set_num          225 src/opop.c               if (set_num(0) > 0) {
get_num           90 src/op.c            while (get_num() > 0) {
$ cd lib
$ global -rx '^[sg]et'
set_num          113 ../src/op.c            set_num(32);
set_num          225 ../src/opop.c               if (set_num(0) > 0) {
get_num           90 ../src/op.c            while (get_num() > 0) {
$ global strlen
$ (cd /usr/src/sys; gtags)
$ export GTAGSLIBPATH=/usr/src/sys
$ global strlen
../../../usr/src/sys/libkern/strlen.c
$ (cd /usr/src/lib; gtags)
$ GTAGSLIBPATH=/usr/src/lib:/usr/src/sys
$ global strlen
../../../usr/src/lib/libc/string/strlen.c
```

FILES

'`GTAGS`' Tag file for object definitions.

'GRTAGS' Tag file for object references.

'GSYMS' Tag file for other symbols.

'GPATH' Tag file for path of source files.

'GTAGSROOT'
: If environment variable *GTAGSROOT* is not set and 'GTAGSROOT' exist in the same directory with 'GTAGS' then use the value as *GTAGSROOT*.

'/etc/gtags.conf', '$HOME/.globalrc'
: Configuration file.

ENVIRONMENT

The following environment variables affect the execution of global:

GTAGSROOT
: The directory which is the root of source tree.

GTAGSDBPATH
: The directory on which gtags database exist. This value is ignored when *GTAGSROOT* is not defined.

GTAGSLIBPATH
: If this variable is set, it is used as the path to search for library functions. If the specified function is not found in a source tree, global also search in these paths.

GTAGSLABEL
: If this variable is set, its value is used as the label of configuration file. The default is `default`.

CONFIGURATION

The following configuration variables affect the execution of global:

`icase_path`(boolean)
: Ignore case distinctions in the pattern.

DIAGNOSTICS

Global exits with a non 0 value if an error occurred, 0 otherwise.

SEE ALSO

gtags-parser(1), gtags(1), htags(1), less(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

AUTHOR

Tama Communications Corporation.

HISTORY

The global command appeared in FreeBSD 2.2.2.

5.2 gtags - create tag files for global.

NAME

gtags - create tag files for global.

SYNOPSIS

gtags [-iIqvw][-f file][-n number][dbpath]

DESCRIPTION

Gtags recursively collect the source files under the current directory, pickup symbols and write the cross-reference data into tag files ('GTAGS', 'GRTAGS', 'GSYMS' and 'GPATH'). You should execute this command at the root of the source tree.

C, C++, yacc, java, PHP and Assembly source files are supported. Files whose names end in '.c' or '.h' are assumed to be C source files and are searched for C style routine and macro definitions. Files whose names end in '.c++' '.cc' '.cpp' '.cxx' '.hxx' '.hpp' '.C' '.H' are assumed to be C++ source files. Files whose names end in '.y' are assumed to be YACC source files. Files whose names end in '.java' are assumed to be Java source files. Files whose names end in '.php' '.php3' '.phtml' are assumed to be PHP source files. Files whose names end in '.s' or '.S' are assumed to be Assembler source files. Other files are searched for C style definitions.

OPTIONS

The following options are available:

'--config'[=name]
: Show the value of config variable name. If name is not specified then show whole of config entry.

'-f', '--file' file
: Read from file a list of file names which should be considered as the candidate of source files. By default, all files under the current directory are considered as the candidate. If file is '-', read from standard input. File names must be separated by newline.

'--gtagsconf' file
: Load user's configuration from file.

'--gtagslabel' label
: label is used as the label of configuration file. The default is `default`.

'-i', '--incremental'
: Update tag files incrementally. You had better use global(1) with the -u option.

'-I', '--idutils'
: Make index files for idutils(1).

'`-n`', '`--max-args`' number
: Maximum number of arguments for gtags-parser(1). By default, gtags invokes the parser with arguments as a lot as possible to decrease the frequency of invoking.

'`-q`', '`--quiet`'
: Quiet mode.

'`-v`', '`--verbose`'
: Verbose mode.

'`-w`', '`--warning`'
: Print warning messages.

dbpath
: The directory in which tag files are generated. The default is the current directory. It is useful when your source directory is on a read only device like CDROM.

EXAMPLES

```
$ ls -F
Makefile      src/    lib/
$ gtags -v
$ global -x main
main              10 src/main.c  main (argc, argv) {
```

FILES

'`GTAGS`' Tag file for function definitions.

'`GRTAGS`' Tag file for function references.

'`GSYMS`' Tag file for other symbols.

'`GPATH`' Tag file for path of source files.

'`/etc/gtags.conf`', '`$HOME/.globalrc`'
: Configuration file.

ENVIRONMENT

The following environment variables affect the execution of gtags:

GTAGSCONF
: If this variable is set, its value is used as the configuration file. The default is '`$HOME/.globalrc`'.

GTAGSLABEL
: If this variable is set, its value is used as the label of configuration file. The default is `default`.

GTAGSCACHE
: If this variable is set, its value is used as the size of btree cache. The default is 500000 (bytes).

CONFIGURATION

The following configuration variables affect the execution of gtags. You can see the default value for each variable with the '`--config`' option.

`GTAGS`(string)
: If this variable is set, its value is used as the command line of parser for GTAGS. The default is '`gtags-parser -dt %s`'.

`GRTAGS`(string)
: If this variable is set, its value is used as the command line of parser for GRTAGS. The default is '`gtags-parser -dtr %s`'.

`GSYMS`(string)
: If this variable is set, its value is used as the command line of parser for GSYMS. The default is '`gtags-parser -dts %s`'.

`skip`(comma separated list)
: Gtags skips files which listed in this list. As a special exception, gtags collect values from multiple `skip` variables. If the value ends with '/', it assumed as a directory and gtags skips all files under it. If the value start with '/', it assumed relative path from the root of source directory.

`suffixes`(comma separated list)
: Suffixes of target source file. As a special exception, gtags collect values from multiple `suffixes` variables. This variable is obsoleted. If the langmap variable is defined gtags no longer refers this.

`icase_path`(boolean)
: Ignore case distinctions in the path. Suffixes check are affected by this capability.

`langmap`(comma separated list)
: Language mapping. Each comma-separated map consists of the language name, a colon, and a list of file extensions. Default mapping is 'c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.cpp.cxx.hxx.hpp.C.H,php:.php.php3.phtml'.

DIAGNOSTICS

Gtags exits with a non 0 value if an error occurred, 0 otherwise.

MESSAGE FORMAT

Verbose message has important level. The most important level is 0, the second is 1 and so on. All the message has level numbers leading blanks.

SEE ALSO

gtags-parser(1), global(1), htags(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

BUG

'GTAGS', 'GRTAGS' and 'GSYMS' are very large. In advance of using this command, check the space of your disk.

Assembler support is far from complete. It extracts only ENTRY() and ALTENTRY() from source file. Probably valid only for FreeBSD and Linux kernel source.

There is no concurrency control about tag files.

Symbols in Assembly source files are not extracted for 'GSYMS'.

AUTHOR

Tama Communications Corporation.

HISTORY

The gtags command appeared in FreeBSD 2.2.2.

5.3 htags - generate hypertext from source code.

NAME

htags - generate hypertext from source code.

SYNOPSIS

htags [-acDfFghInosTvwx][-d dbpath][-m name][-S cgidir][-t title][dir]

DESCRIPTION

Htags makes hypertext of C, C++, Yacc, Java, PHP and Assembly source code.

In advance of using this command, you must execute gtags(1) from the root directory of the source tree. Then you can execute htags from the same place. Htags makes an directory named 'HTML' and generates hypertext in it. You can start browsing from 'HTML/index.html'.

Since htags generates static hypertext as long as the '-D' or '-f' option is not specified, you can move it anywhere and browse it with any browser without web server.

You must use same parser for both gtags(1) and htags. If you use the default parser, it is not necessary to consider for it.

OPTIONS

The following options are available:

'-a', '--alphabet'
: Make an alphabetical function index which is suitable for a large project.

'--caution'
: Include caution message to prohibit downloading.

'-c', '--compact'
: Compress html files by gzip(1). You need to set up a web server so that gzip(1) is invoked for each compressed file. See 'HTML/.htaccess' that is generated by htags.

'--cvsweb' url
: Include cvsweb URL. url is used as base of URL. When directory 'CVS' exists in the root directory of source project, content of 'CVS/Repository' is used as relative path from the base.

'--cvsweb-cvsroot' cvsroot
: Specifies cvsroot in cvsweb URL.

'-D', '--dynamic'
: Generate object lists dynamically using CGI program. By default, object lists are generated statically. Though this option decrease both the size and the generation time of the hypertext, you need to set up a web server, and you cannot move the hypertext from the source directroy.

'-d', '--dbpath' dbpath
: Specifies the directory in which 'GTAGS' and 'GRTAGS' exist. The default is the current directory.

'-f', '--form'
: Support search form using CGI program. You need to set up a web server, and you cannot move the hypertext from the source directroy.

'-F', '--frame'
: Use frame for each part of the contents.

'--full-path'
: List file names with full path in file index. By default, list just the last component of a path.

'-g', '--gtags'
: Execute gtags(1) before creating hypertext. The '-v', '-w' and dbpath are passed to gtags.

'-h', '--func-header'[=position]
: Insert function header for each function. By default, htags doesn't generates it. You can specify the position using position argument, which allows one of `before`, `right` and `after`. The default position is `after`.

'-I', '--icon'
: Use icons instead of text for some links.

'--disable-grep'
: Disable grep in search form(-f,–form).

'--gtagsconf' file
: Load user's configuration from file.

'--gtagslabel' label
: label is used for the label of configuration file. The default is `default`.

'`--insert-header`' file
: Insert custom header derived from file after <body> tag.

'`--insert-footer`' file
: Insert custom footer derived from file before </body> tag.

'`-m`', '`--main-func`' name
: Specify the main function name. The default is `main`.

'`-n`', '`--line-number`'[=columns]
: Print line numbers. By default, doesn't print them. The default value of columns is 4.

'`--no-map-file`'
: Doesn't generate '`MAP`' and '`FILEMAP`' file. By default, htags generates them.

'`-o`', '`--other`'
: Pick up not only source files but also other files except for binary files.

'`--statistics`'
: Print statistics information.

'`-s`', '`--symbol`'
: Make anchors not only for functions but also other symbols. '`GSYMS`' tag file needed.

'`-S`', '`--secure-cgi`' cgidir
: Write CGI programs into the cgidir to realize a centralised CGI program. Script alias is '`/cgi-bin`' by default. You can overwrite this value using config variable `script_alias` in '`gtags.conf`'.

'`-t`', '`--title`' title
: The title of this hypertext. The default is the last component of the current directory.

'`-T`', '`--table-flist`'[=fields]
: Generate file list using <table> tag. The fields is used for field number in a line. The default is 5.

'`--table-list`'
: List tags using <table> tag.

'`--tabs number`'
: Tab stop. The default is 8.

'`-v`', '`--verbose`'
: Verbose mode.

'`-w`', '`--warning`'
: Print warning messages.

'`-x`', '`--xhtml`'[=version]
: Generate XHTML hypertext instead of HTML. If the '`--frame`' option is specified then generate XHTML-1.0 Frameset for index.html and generate XHTML-1.0 Transitional for other files, else if version is `1.1` or config variable `xhtml_version` is set to `1.1` then generate XHTML-1.1 else XHTML 1.0 Transitional.

dir The directory in which hypertext is generated. The default is the current directory.

EXAMPLES

```
$ gtags -v
$ htags -sanohITvt 'Welcom to XXX source tour!'
$ firefox HTML/index.html
```

FILES

'GTAGS' Tag file for function definitions.

'GRTAGS' Tag file for function references.

'GSYMS' Tag file for other symbols.

'GPATH' Tag file for path of source files.

'/etc/gtags.conf', '$HOME/.globalrc'
Configuration file.

'HTML/index.html'
Index file for hypertext.

'HTML/MAP'
Mapping file for converting tag into path of hypertext. External system utilize this file.

'HTML/FILEMAP'
Mapping file for converting file name into path of hypertext. External system utilize this file.

'HTML/style.css'
Style sheet file. This file is generated when the '--xhtml' option is specified.

ENVIRONMENT

The following environment variables affect the execution of htags:

TMPDIR If this variable is set, its value is used as the directory to make temporary files. The default is '/tmp'.

GTAGSCONF
If this variable is set, its value is used as the configuration file. The default is '$HOME/.globalrc'.

GTAGSLABEL
If this variable is set, its value is used as the label of configuration file. The default is `default`.

GTAGSCACHE
If this variable is set, its value is used as the size of btree cache. The default is 500000 (bytes).

CONFIGURATION

The following configuration variables affect the execution of htags: If the '`--xhtml`' option is specified then all definitions of HTML tag are ignored. Instead, you can customize the appearance using style sheet file ('`style.css`').

`datadir`(string)
: Shared data directory. The default is '/usr/local/share' but you can change the value using configure script. Htags lookup template files in the 'gtags' directory in this data directory.

`htags_options`(string)
: Default options for htags. This value is inserted into the head of arguments.

`xhtml_version`(1.0|1.1)
: XHTML version. 1.0 and 1.1 are acceptable. The default is 1.0.

`body_begin`(string)
: Begin tag for body. The default is '<body text=#191970 bgcolor=#f5f5dc vlink=gray>'.

`body_end`(string)
: End tag for body. The default is '</body>'.

`table_begin`(string)
: Begin tag for table. The default is '<table>'.

`table_end`(string)
: End tag for table. The default is '</table>'.

`title_begin`(string)
: Begin tag for Title. The default is '<h1>'.

`title_end`(string)
: End tag for Title. The default is '</h1>'.

`comment_begin`(string)
: Begin tag for comments. The default is '<i>'.

`comment_end`(string)
: End tag for comments. The default is '</i>'.

`dynamic`(bool)
: Generate object list dynamically.

`sharp_begin`(string)
: Begin tag for 'define'. The default is ''.

`sharp_end`(string)
: End tag for 'define'. The default is ''.

`brace_begin`(string)
: Begin tag for brace. The default is ''.

`brace_end`(string)
: End tag for brace. The default is ''.

`reserved_begin`(string)
: Begin tag for reserved word. The default is ''.

`reserved_end`(string)
: End tag for reserved word. The default is ''.

`position_begin`(string)
: Begin tag for posiotion mark. The default is ''.

`position_end`(string)
: End tag for posiotion mark. The default is ''.

`colorize_warned_line`(boolean)
: Colorize warned line using `warned_line_begin` and `warned_line_end`. The default is false.

`warned_line_begin`(string)
: Begin tag for line which htags warned. The default is ''.

`warned_line_end`(string)
: End tag for line which htags warned. The default is ''.

`hr`(string)
: Horizontal rules. The default is '<hr>'.

`ncol`(number)
: Columns of line number. The default is 4.

`tabs`(number)
: Tab stop. The default is 8.

`flist_fields`(number)
: Field number of file index. The default is 5.

`full_path`(boolean)
: List file names with full path in file index. By default, list just the last component of a path.

`table_list`(boolean)
: List tags using <table> tag. The default is false.

`table_flist`(boolean)
: Use <table> tag for file index. The default is false.

`normal_suffix`(string)
: Suffix for normal html file. The default is 'html'.

`no_map_file`(boolean)
: Doesn't generate 'MAP' file. The default is false.

`gzipped_suffix`(string)
: Suffix for compressed html file. The default is 'ghtml'.

`script_alias`(string)
: Script alias for safe cgi script ('-S').

`show_position`(boolean)
: Show position per function definition. The default is false.

`definition_header`(no|before|right|after)
: Position of function header. The default is 'no'.

`other_files`(boolean)
: File index includes not only source files but also other files. The default is false.

`disable_grep`(boolean)
: Disable grep in search form(-f,–form). The default is false.

`enable_idutils`(boolean)
: Enable idutils in search form(-f,–form). The default is false.

`include_file_suffixes`(comma separated list)
: Suffixes of include file. The default is 'h,hxx,hpp,H,inc.php'.

`langmap`(comma separated list)
: Language mapping. Each comma-separated map consists of the language name, a colon, and a list of file extensions. Default mapping is 'c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.cpp.cxx.hxx.hpp.C.H,php:.php.php3.phtml'.

`copy_files`(boolean)
: Copy files instead of linking. When the '`-f`' option is used, htags make links of tag files in '`cgi-bin`' directory by default.

DIAGNOSTICS

Htags exits with a non 0 value if an error occurred, 0 otherwise.

MESSAGE FORMAT

Verbose message has important level. The most important level is 0, the second it 1 and so on. All the message has level numbers leading blanks.

SEE ALSO

gtags-parser(1), global(1), gtags(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

BUG

Generated hypertext is VERY LARGE. In advance, check the space of your disk.

PHP supprt is far from complete.

AUTHOR

Tama Communications Corporation.

HISTORY

The htags command appeared in FreeBSD 2.2.2.

5.4 gtags-parser - print cross reference list for gtags.

NAME

gtags-parser - print cross reference list for gtags.

SYNOPSIS

gtags-parser [-bdenrstvw] file ...

DESCRIPTION

Gtags-parser print cross reference list for gtags(1) from the specified C, C++, yacc, java, PHP and Assembly source to standard output. Each line of output contains the object name, the line number which it appears, the file in which it is defined, and a line image separated by white-space. It's same with the output of ctags(1) with '-x' option.

Depending upon the options provided to gtags-parser, objects will consist of function definitions, function references and other symbols.

Files whose names end in '.c' or '.h' are assumed to be C source files and are searched for C style routine and macro definitions. Files whose names end in '.c++' '.cc' '.cpp' '.cxx' '.hxx' '.hpp' '.C' '.H' are assumed to be C++ source files. Files whose names end in '.y' are assumed to be YACC source files. Files whose names end in '.java' are assumed to be Java source files. Files whose names end in '.php' '.php3' '.phtml' are assumed to be PHP source files. Files whose names end in '.s' or '.S' are assumed to be Assembler source files. Other files are searched for C style definitions.

Yacc files each have a special tag. yyparse is the start of the second section of the yacc file.

This command is the default parser of GLOBAL source code tag system.

OPTIONS

The following options are available:

'-b', '--begin-block'
: Force level 1 block to begin when reach a left brace at the first column. (C only)

'-d', '--define'
: Pick up not only function but also macro without argument as a definition.

'-e', '--end-block'
: Force level 1 block to end when reach a right brace at the first column. (C only)

'-n', '--no-tags'
: Suppress output of tags. It is useful to use with '-w' option.

'-r', '--reference'
: Locate function references instead of function definitions. 'GTAGS' is needed at the current directory. (C, C++ and Java source only) By default, locate function definitions.

'-s', '--symbol'
: Collect symbols other than functions. By default, locate function definitions.

'-t', '--typedef'
: Pick up not only function but also typedef name and enum member as a definition.

'-v', '--verbose'
: Verbose mode.

'-w', '--warning'
: Print warning message.

'--langmap'=map
: Language mapping. Each comma-separated map consists of the language name, a colon, and a list of file extensions. Default mapping is 'c:.c.h,yacc:.y,asm:.s.S,java:.java,cpp:.c++.cc.cpp.cxx.hxx.hpp.C.H,php:.php.php3.phtml'.

The '-r' and '-s' options override each other; the last one specified determines the method used.

DIAGNOSTICS

Gtags-parser exits with a non 0 value if an error occurred, 0 otherwise. Duplicate objects are not considered errors.

SEE ALSO

global(1), gtags(1), htags(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

BUG

Gtags-parser relies on the input being well formed, and any syntactical errors will completely confuse it.

Assembler support is far from complete. Probably valid only for FreeBSD and Linux kernel source.

AUTHOR

Tama Communications Corporation.

HISTORY

The gtags-parser(gctags) command appeared in FreeBSD 2.2.2.

5.5 gozilla - force mozilla to display specified source file.

NAME

gozilla - force mozilla to display specified source file.

SYNOPSIS

gozilla [-b browser][-p][+no] file
gozilla [-b browser][-p] -d name

DESCRIPTION

Gozilla force mozilla to display specified source file as a hypertext. Gozilla can be used with other browsers like firefox and epiphany.

In advance of using this command, you must execute gtags(1) and htags(1) at the root directory of the source tree to make tag files. Then you can execute gozilla at anywhere in the source tree.

First form:
You can specify source file and the line number optionally.

Second form:
You can specify definition name directly. Definition name must exist in '`GTAGS`' tag file.

Some browsers require you to load it before executing gozilla. Whether or not gozilla waits for exiting of browser depends on browser.

OPTIONS

The following options are available:

'`+no`' line number. It must be a line on which function definition or function reference is exist. If you execute htags(1) with '`-l`' option, you can specify any line.

'`-b`' browser
browser to use. By default, it is assumed mozilla.

'`-p`' just print generated target URL.

file path of source file or alias name.

'`-d`' name print function.

'`-q`', '`--quiet`'
Quiet mode.

'`-v`', '`--verbose`'
Verbose mode.

'`--version`'
Show version number.

'`--help`' Show help.

FILES

'`HTML/`' hypertext of source tree.

'`GTAGS/`' tags file for function definitions.

'`$HOME/.gozillarc`'
alias file. Please read source code for the detail.

ENVIRONMENT

GTAGSROOT
The directory which is the root of source tree.

GTAGSDBPATH
The directory on which gtags database exist. This value is ignored when *GTAGSROOT* is not defined.

BROWSER
browser to use. By default, it is assumed mozilla.

EXAMPLES

```
$ gtags
$ htags
$ global -x main
main              82 ctags.c          main(argc, argv)
$ mozilla &
$ gozilla +82 ctags.c

$ firefox &
$ gozilla -b firefox +82 ctags.c

$ setenv BROWSER 'epiphany --new-tab'
$ epiphany &
$ gozilla +82 ctags.c
```

DIAGNOSTICS

Gozilla exits with a non 0 value if an error occurred, 0 otherwise.

SEE ALSO

global(1), gtags(1), htags(1), firefox(1), epiphany(1), mozilla(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

NOTES

Gozilla means 'Global for mozilla'.

BUGS

Gozilla can treat not only source file but also normal file, directory, HTML file and even URL, because it is omnivorous.

AUTHORS

Tama Communications Corporation.

HISTORY

The gozilla command appeared in FreeBSD 2.2.2 but did not installed by default.

5.6 gtags-cscope - pseudo cscope which implements the line-oriented interface

NAME

gtags-cscope - pseudo cscope which implements the line-oriented interface

SYNOPSIS

gtags-cscope [-Cqv]

DESCRIPTION

Gtags-cscope is a pseudo cscope which implements the line-oriented interface. You can use this command for various clients instead of true cscope.

Since gtags-cscope is intended to make GLOBAL available through cscope interface, the output is not necessarily the same as cscope.

OPTIONS

The following options are available:

'-C', '--ignore-case'
: Ignore letter case when searching.

'-q', '--quiet'
: Quiet mode.

'-v', '--verbose'
: Verbose mode.

EXAMPLES

```
$ gtags-cscope
>> help
0<arg>: Find this C symbol
1<arg>: Find this definition
2<arg>: Find functions called by this function
(Not implemented yet.)
3<arg>: Find functions calling this function
4<arg>: Find this text string
6<arg>: Find this egrep pattern
7<arg>: Find this file
8<arg>: Find files #including this file
c: Toggle ignore/use letter case
r: Rebuild the database
q: Quit the session
h: Show help
>> 1main
cscope: 9 lines
global/global.c main 158 main(int argc, char **argv)
gozilla/gozilla.c main 155 main(int argc, char **argv)
```

```
gtags-parser/gctags.c main 158 main(int argc, char **argv)
gtags-cscope/gtags-cscope.c main 115 main(int argc, char **argv)
gtags/gtags.c main 150 main(int argc, char **argv)
htags-refkit/htags_path2url.c main 281 main(int argc, char **argv)
htags/htags.c main 1400 main(int argc, char **argv)
libglibc/getopt.c main 704 main (argc, argv)
libglibc/getopt1.c main 93 main (argc, argv)
>> q
$ _
```

DIAGNOSTICS

Gtags-cscope exits with a non 0 value if an error occurred, 0 otherwise.

SEE ALSO

cscope(1), gtags-parser(1), gtags(1), global(1), htags(1).

GNU GLOBAL source code tag system
(http://www.gnu.org/software/global/).

BUG

The second field of the output is almost <unknown> since GLOBAL doesn't recognize it. Command 2 (Find functions called by this function) is not implemented.

AUTHOR

Tama Communications Corporation.

HISTORY

The gtags-cscope command appeared in 2006.

Appendix A Copying This Manual

A.1 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.

A.1.1 ADDENDUM: How to use this License for your documents

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

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
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.

Option Index

-

F

Table of Contents