GAdoc

A program to extract the autodocs from source files

User-Manual
Version 1.0

Gerhard Leibrock, 1995

1 Disclaimer

GAdoc, Feb 1995

Although this piece of software was developped with greatest care, the author rejects any liability or responsibility for these or any other consequences from the use of `GAdoc` or whatsoever. This includes, but is not limited to, damage to your equipment, to your data, personal injuries, financial loss or any other kinds of side effects.

`NO warranty`

There is no warranty for this software package. Although this piece of software was developped with greatest care, it is impossible to exclude the possibility of errors.You are therefore using this material at your `cvery own risk`. The author cannot be made responsible for any damage which is caused by using this software package.

It is even possible, that this software will not run at all on your system.

2 Requirements

Amiga computer family

The minimal configuration for running this program is at least 1MB of free RAM. If you got a compiled version of the program, you need at least OS 2.04, too, because it got compiled using gcc263 and this is an 2.x only compiler.

Others

To recompile GAdoc, you need an ANSI/C compiler, GNU/C works. Just type in “make” (using gcc) and wait.

If you want to use this program with MS-DOS, you have to alter the file extexsions, of the generated programs, otherwise the program might not work at all.

Using this piece of software with U*IX computers caused no problem with SOLARIS and NetBSD. Use it at your own risk on other architectures.

3 Installation

Just copy the executable to somewhere in your path.

4 What are autodocs?

Every programmer knows about the problems of software documentation or problems that arise from team work.

After having finished a project or a part of it, you are glad that it works at all, but what about your partner or your boss? They also want to have the functions documented, if possible printed on paper.

Now you have to sit down again and write down all that you kow about your functions. But what happens after you changed some parts of your functions, or added some ...? Your documentation will have to be rewritten.

That's really a problem that also affects big firms, not only hobby programmers. There exist several systems, that allow you to write your program and documentation at the same time, like cweb does. cweb is great and very powerful, but most of the time, programmers need something more simple, this is the time that autodocs enter your life.

Autodocs are easy to use within your source code, no matter which language you use, you only need a language, that allows you to use blocks of comments.

4.1 Starting a project

Every source file should start with an autodoc, that describes your project and gives GAdoc information about the author, the copyright holder etc.

Here is an example:

```
/****h* project/Intro ************************************************
*
* NAME
*  project
```

```
*
* COPYRIGHT
*  No name software, inc.
*
* FUNCTION
*   This project is a project about projects.
*
* AUTHOR
*  Gerhard Leibrock
*
* VERSION
*  1.0
*
* NOTES
*  First project about how to make a project.
*
*****************************************************************************
*/
```

The first character gets ignored, then the line is checked for 4 asterisks followed by "h*" and a space.

After this "begin" keyword follows the name of the project, a slash ("/") and a short description (Could also be "project/TheNameOfTheGame").

The keywords have the following meaning:

`NAME` What's the name of your software?

`COPYRIGHT`
Who holds the copyright?

`FUNCTION` What can be done with this program?

`AUTHOR` Who wrote the software?

`VERSION` Version number, remeber that the numbers should be read as real decimals, meaning a Version number of 1.4 (one-point-4) refers to an older release than a number of 1.11 (one-point-eleven).

`NOTES` Bugs, features, statements, etc.

Remember that this "header" must be specified before any other autodoc, otherwise, GAdoc will cease its operation and give you an error message.

It is impossible not to specify the header. GAdoc will fill out the author's name with Unknown, the version number with 0.0 and other vague values.

Also remember the order of the keywords, meaning NAME should be specified before NOTES, etc..

4.2 Function description

Here is an example of an autodoc for a function within C-source code:

```
/****** misc/ShowReq ****************************************
*
* NAME
*  ShowReq -- Shows a requester (Needs at least KS 1.3)
*
* SYNOPSIS
*   void = ShowReq( Title, Text);
*                     D0   D0
*
*   void ShowReq ( STRPTR, STRPTR);
*
* FUNCTION
*   Display a requester with title and some text.
*
* INPUTS
*   Title  - Title of the requester
*   Text   - Message for the user
*
* RESULT
*   None
*
* EXAMPLE
*   ShowReq("J EDGAR HOOVER", "I regret to say that we of the FBI are\n"
*                             "powerless to act in cases of oral-genital\n"
*                             "intimacy, unless it has in some way\n"
*                             "obstructed interstate commerce.");
*
*
* NOTES
*   Will not work on Amigas with Kickstart version < 1.3
*
* BUGS
*
*
* SEE ALSO
*   special/ShowTextAndPictureReq
```

```
*
******************************************************************************/
```

An autodoc for functions is recognized by the sequence of six asterisks (“******”), the first character gets ignored (Here the “/”). Then follows a blank and the name of the module followed by “/” and the functions name. The rest of the line gets ignored.

Here some valid examples:

```
/****** module/name ----------------------------
;****** module/name ----------------------------
******* module/name ----------------------------
```

This marks six asterisks as a special keyword, which should be used vers carefully.

After this “begin” statement follows the description of the functions. You can use some of the following keywords to describe it:

`NAME` Put in the function's name, followed by two minus (“–”) symbols and a one line description. You should write sentences, ending with a period (“.”).

`SYNOPSIS` It consists of three parts: The calling convention, the assembly registers and the function prototype. The line with the prototype should be “ready to compile”. Take care of this (This doesn't matter to GAdoc, but “autodoc” as shipped with the 3.1 Amiga Developer Update, requires this.

`FUNCTION` Describe what your function does, try to avoid jargon and use generally accepted english. Write as much as needed but as short as possible.

`INPUTS` Describe the parameter's valid range and tell the reader what happens, it you receive e.g. a NULL-pointer or something like that. Use the variable name specified in the SYNOPSIS line and use a minus (“-”) as a seperator (See example above).

`RESULT` Describe the return values and error conditions, as error messages ar also seen as a kind of return value.

`EXAMPLE` `Short` example of how to use your function. Test the example to avoid misunderstandings.

`NOTES` Hints, warnings, tricks: Talk about side effects (If they exist), etc.

`BUGS` If your function has bugs, please describe them. Otherwise ignore this.

`SEE ALSO` If your function needs assistance of other functions, or you want the user to scan e.g. include files, list them here. GAdoc recognizes references to other modules ad generates

a cross link! But therefore, the name before the "/" `must` be identical to the file, where the function is located.

It is very important, that you use the keywords in the order listed above. You might ignore some of them, so GAdoc will replace them with empty ones.

An autodoc end with at least three asterisks at the start of a line ("***").

Please do not use any tabs within your autodos since they are of no use with makeinfo or TEX.

4.3 Internal functions

If you do have functions, that should not be documented to others for any reason, you should mark them with "****i*" (four asterisks, "i", one asterisk). These functions will only be extracted if you specify the "-i" option with GAdoc.

The syntax for internal functions is identical to that of normal functions.

5 How to use GAdoc

GAdoc can be used from any shell and accepts the following parameters:

gadoc <source> <output> [-i -amiga -c]

`<source>`:
: Name of the source files

<output>: Name of the generated files <output>.menu, <output>.data (Existing files with identical names will be deleted without warning!)

`-i`
: Also extract internal autodocs

`-amiga`
: Include amiga support for texinfo file

`-c`
: Convert '`\*`' (Backslash asterisk) to '`/*`' and "`\*`" to '`*/`' (Important for C compilers, that do not allow nested comments, and this will happen, if you write down c-comments within your autodoc).

The lines for the autodocs are limited to 80 chars/line.

GAdoc reads the specified source file and generates two files with the extensions .menu and .data. The file with the .menu extension is considered to be the main part of the generated texinfo file.

5.1 The texinfo format

Texinfo is a documentation system that uses a single source file to produce both on-line information and printed output.

To produce on-line information like plain ascii text or hypertext format, you need a program called makeinfo.

It will generate a plain ascii file (Here: "output.doc"), if you invoke it like that:

```
makeinfo --no-headers -o output.doc output.menu
```

If you want to generate a gnu infoview file, you should invoke makeinfo with just the filename:

```
makeinfo output.menu
```

It will generate a file "output.guide", as this name is specified in output.menu.

If you want to generate amiga specific files, use the "–amiga" flag, e.g. to generate the amiga hypertext format for amigaguide:

```
makeinfo --amiga output.menu
```

To generate a .dvi file with TEX, just call virtex with the following parameters:

```
virtex output.menu
```

Make sure, you have the texinfo available to use it with virtex. (If you use a special amiga version, you need amigatexinfo also, but I never needed it at all.)

6 Error messages

`-xy: Specified twice (See argument # n)`
: You specifiifed the parameter -xy twice.

`Unknown argument: <arg>`
: The argument <arg> that you specified in the command line is unknown to GAdoc. Remeber that arguments with a mines ("-") are only accepted in lower case letters.

`Could not open file ``<fname>''.`
: GAdoc could not open the requested file, maybe you should check its spelling (Some OS distinguish between lower case and upper case letters.)

`Error Line n: Either keyword <key> used twice or in wrong order.`
: Maybe you did specify a keyword more then one time in one function description, or you used in wrong order.

`Error: End of file occured during internal docs.`
: Maybe you forgot the three asterisks to end this internal documentation.

`Error Line n: Header specified after autodocs.`
: As stated in this file, the header is to be specified before any other autodoc.

`Error Line n: Second time header gets specified.`
: You did specify two header sections.

`Error: End of file occured during internal block.`
: During reading an autodoc for an internal function, an END OF FILE signal occured, maybe you forgot to specify the end marker.

Index

Table of Contents