GAdoc

A program to extract the autodocs from source files

User-Manual
Version 1.2

Gerhard Leibrock, 1995

1 Copyright

GAdoc, March 1995

© 1995 by Gerhard Leibrock

This software is subject to the "Standard Amiga FD-Software Copyright Note". It is Cardware as defined in paragraph 4.c. If you like it and use it regularly please send a postcard to the author. For more information please read "AFD-COPYRIGHT" (Version 1 or higher).

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.

If you want to use the `-aicon` option, you should use Amiga OS 3.0 at least, since this icon calls multiview, which gets shipped with OS 3.0 and later versions.

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 what you know about your functions. But what happens after you change some parts of your functions, or add 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 where 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
*  Its not the size of the ship, its the size of the waves.
*  LITTLE RICHARD
*
*
***/
```

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

Note: Because GAdoc supports the usage of IDs that can be written instead of the 2nd and 3rd asterisk, it is also possible to begin the autodoc entry with something different from "/****h*", e.g. it can begin with "/*IT*h*", if you choose "IT" as your favourite ID, maybe to indicate a header written in italian. These special autodoc entries can only be extracted using the -s argument. This flag can also be used to group special functions, e.g. an ID like "WD" could specify the function as a special Window function, or something like that.

The two asterisks at the 2nd and 3rd position can be seen as the default ID.

After this "begin" keyword you have to specify the name of the project followed by 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..

You should take a look at the GAdoc source code, which gets delivered with gadoc.

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    D1
*
*   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 through 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 very carefully.

Note: Because GAdoc supports the usage of IDs that can be written instead of the 2nd and
            3rd asterisk, it is also possible to begin the autodoc entry with something different from
            "******" (6 asterisks), e.g. it can begin with "/*IT***", maybe to indicate an entry
            written in italian. These special autodoc entries can only be extracted using the `-s`
            argument. This flag can also be used to group special functions, e.g. an ID like "WD"
            could specify the function as a special Window function, or something like that.

            The two asterisks could be seen as the "default ID".

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 des-
            cription. You should write sentences, ending with a period (".").

SYNOPSIS   It consists of three parts: The calling convention, the assembly registers and the func-
            tion 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 misunderstan-
            dings.

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 ends 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.

`Note:`   Because GAdoc supports the usage of IDs that can be written instead of the 2nd and 3rd asterisk, it is also possible to mark the autodoc entry with something different from "****i*", e.g. it could look like "*IT*i*", maybe to indicate a header written in italian. These special autodoc entries can only be extracted using the `-s` argument. The two asterisks could be seen as the "default ID".

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 -c -s<id> -amiga -aicon]

`<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

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

-s<id>   If you want to write the documentation in different languages, than you can specify a
                special ID, for which every autodoc entry gets checked. If the entry matches the two
                character ID, than the autodoc gets extracted, otherwise it won't get touched. This
                special ID must be specified as the third and fourth character of the beginning line of
                the autodoc.

                                Standard   Using "GB" as ID
                                1234567   1234567

                    internal   ?****i*   ?*GB*i*
                    usual   ?******   ?*GB***
                    header   ?****h*   ?*GB*h*

                    ? := any character

                E.g. you want to extract all autodocs written in german, so you could use "DE" as ID
                for each german autodoc entry. Then you can call GAdoc with the option "-sDE" and
                all autodocs that have "DE" will be extracted.

                Default is "**". This makes it easy for you to write multi-language autodocs and keep
                them together in one file.

-amiga   Include amiga support for texinfo file

-aicon   If this flag is specified, gadoc will create an icon with the name "<output>.guide.info".
                This file is only useful for people that do use an Amiga computer, otherwise the created
                icon will be worthless. After creating the guide file with makeinfo, you will be able to
                use the icon from workbench by just clicking on it and it will be loaded into multiview
                (Workbench 3.0 required at least).

    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 includes 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 specifed 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 minus ("-") 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 the keywords 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.`
>   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