Config

Adam Dawes

COLLABORATORS			
	TITLE : Config		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Adam Dawes	August 26, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Config

1.1 Config v2.0

```
Config Library v2.0

   by Adam Dawes

7th September, 1996
```


1.2 Introduction

Well, I hate Windows as much as the next man, but occasionally I stumble across a good idea hidden away within the operating system.

There are a couple of functions buried in there for reading and writing configuration files, and they actually make things very easy. I decided I'd had enough of messing around with config files on my Amiga, so I've ported the functions.

The idea is that the configuration files take a definitive structure which the config.library functions can understand. Each config file is split in

to a number of "Sections" (which are stored in the config file as a keyword inside square brackets). In each of these sections are a number of "Items", each of which contains an actual data item. The items are local to the section that contains them, so it's perfectly legal to use one item name in several sections, they'll all be treated separately.

The beauty behind the functions is you don't have to worry about creating files or scanning through them. Even when it comes to reading data, you don't have to care if the config file exists or not as you provide a default value to use if the file/section/item cannot be located. Everything is automated within the config.library functions.

1.3 Using the Config Functions

There are 4 public functions within the config.library, as follows:

```
WriteConfig()
WriteConfigNumber()
ReadConfig()
ReadConfigNumber()
```

Select one of the functions for further details. For examples of these functions in use, please see the supplied ConfigTest.c or ConfigTest.s sourcecode.

All necessary include files for using the library in C are supplied within the Config archive. I've also converted the main include file to assembly language. If anyone feels like converting the include files for other languages, please contact me!

1.4 WriteConfig()

```
Function:
    int WriteConfig(char *filename, char *section, char *item, char *data);

Parameters:
    filename = the name of the config file (eg, "S:MyConfig.cfg")
    section = the name of the section to add config data to
    item = the item within the section to contain the config data
    data = the actual data itself

Details:
    You do not need to worry about anything at all when calling this
    function. If the config file specified doesn't exist, it'll be
    created for you. If the section specified doesn't exist, that will
    be created. If the item doesn't exist, it'll be added to the
    appropriate section; if it does exist, the previous data will be
    replaced by the new data.

    If all goes well, the function will return CFG_WRITE_SUCCESS. If
    something goes wrong (out of memory, or the specified file
    cannot be written to), CFG_WRITE_FAIL will be returned.
```

1.5 WriteConfigNumber()

```
Function:
    int WriteConfigNumber(char *filename, char *section, char *item, long data);

Parameters:
    filename = the name of the config file
    section = the name of the section to add config data to
    item = the item within the section to contain the config data
    data = a long integer to be written to the file

Details:
    This function is more useful when reading/writing numbers to the config
    file. It is actually only a small front-end to the WriteConfig()
    function.

    If you need to write numbers that aren't longs (floats, for instance),
    just make your own function that sprintf()'s the number to a text
    buffer, and then calls WriteConfig() with the buffer as its data
    parameter.

    As before, this will return CFG_WRITE_SUCCESS is all is well, or
    CFG_WRITE_FAIL if something goes wrong.
```

1.6 ReadConfig()

```
Function:
    int ReadConfig(char *filename, char *section, char *item, char *buffer,
                   int bufsize, char *def);

Parameters:
    filename = the name of the config file
    section = the name of the section to read config data from
    item = the item within the section that contains the config data
    buffer = an empty character array ready to receive the config data
    bufsize = the maximum length of chars than can be written to the buffer
    def = a string that will be written to the buffer if the requested
          config item cannot be found

Details:
    Use ReadConfig() to get data back from your config file. It will look
    for the specified section/item in the specified file.

    If the file cannot be opened, the section cannot be located, or the
    item within the section is not found, the function will copy the
    default string to your buffer, and return CFG_READ_SUCCESS. Therefore,
    you don't need to care at all about whether the config file exists.
    Just tell the function what you want to receive if your requested data
    cannot be found.

    The function will return CFG_READ_SUCCESS if everything is ok, or
    CFG_READ_FAIL if the string to be returned is larger than the supplied
    buffer.
```

1.7 ReadConfigNumber()

```
Function:
    long ReadConfigNumber(char *filename, char *section, char *item,
         long def);

Parameters:
    filename = the name of the config file
    section = the name of the section to read config data from
    item = the item within the section that contains the config data
    def = a long value that will be returned if the requested config item
          cannot be found

Details:

    As with WriteConfigNumber(), this is just a front end to the
    ReadConfig() function.

    This will return as a long value, the number located in the data item
    specified. If the data item cannot be found, your default value will be
    returned instead.

    This function is assumed never to fail. If the item to be returned is
    actually textual, instead of a number, 0 will be returned.
```

1.8 Requirements

```
In order to use config.library, you will need to be running V36+ of the
Amiga operating system. This means that any program using config.library
also requires V36+. If your program is designed to work on earlier versions
of the OS, you will have to think about an alternative to config.library.
```

1.9 Legal Stuff

```
Config.library and its associated files are not public domain. They may be
distributed freely as long as no unreasonable charge is imposed. They may
not be included within any commercial package without express written
permission from the author; the exceptions from this are the Aminet CDs and
Fred Fish's collections.

Config.library itself may be freely distributed within any PD, freeware or
shareware packages. If you wish to use config.library in a commercial
package, contact me first.

I do not accept responsibility for any damage done to your system or data
lost, directly or indirectly, as a result from using this program or any of
its associated files. You use the program entirely at your own risk. Of
course if you *do* experience problems then I'll do what I can to sort them
out, and please let me know so that I can try to cure them in a future
release.

If you decide to include config.library in your own programs, please
```

consider giving me a mention in the documentation. It'd also be nice if you sent me an email so I can see how far config.library is being used.

1.10 History

v1.0 (25.5.96)

Initial release, as C sourcecode.

v1.01 (29.5.96)

Fixed a memory allocation problem that could cause some nasty crashes. Version 1.0 of this code should *not* be used as it may result in disk unvalidation. Please recompile any programs that use Config 1.0 with the new Config code. Thanks to Mike Gooder for noticing it!

Cleaned up a couple of minor code inconsistencies that caused warnings to occur when compiling with SAS/C. Should be clean now.

v2.0 (7.9.96)

Config has been completely restructured as a shared library, config.library. This means that any future updates can be incorporated in to existing programs without having to rerelease the programs themselves.

This also opens up the possibilities of using config.libraries in languages other than C. If you wish to convert the include files to any other languages (Amiga E, BASIC, etc.) then please contact me.

1.11 To Do...

Whilst config.library is fully functional, there's always room for improvements! On my ToDo list are (amongst others) the following:

- Config file caching. Currently, each read/write operation involves at least 1 file open. This can result in a fair amount of disk access when reading/writing large numbers of config items.

  The solution to this is two new functions, one to cache the whole config file in memory, and one to flush it from memory back to disk. This will, unfortunately, mean that programs using config.library will have to be recompiled in order to use these extra functions. Of course, using the existing functions without caching the config file will act exactly as it does currently, so there will be no compatibility issues.

- Functions to delete an item/section from the config file.

To the existing users of Config with whom I've spoken in the past: my apologies for not implementing these features yet.. I hope this will show that I haven't abandoned this project, it just takes me a while to get things done! ;)

1.12 Acknowledgements

My thanks to the following people, without whom config.library would not be what it is today:

- Andreas Kleinert (for his example library code (CExecLib.lha))
- Bret McGee (for his invaluable help with getting my .library to compile)

Also my thanks go to Mike Gooder, Dirk Holtwick and Christian Kemp, for their suggestions for the future of config.library.

1.13 Contacting the author

Please do write to me if you like config.library or if you have any problems with it or suggestions for a new version. I can't promise to reply quickly if you write via snail-mail, but I will do my best to always reply to email messages. I can be contacted at:

InterNet
Adam@beachyhd.demon.co.uk

http://www.pavilion.co.uk/rda/adam

FidoNet
Adam Dawes@2:441/93.5

SnailMail
Adam Dawes
47 Friar Road
Brighton
BN1 6NH
England