Amigamain

Matthias Rustler

COLLABORATORS			
	TITLE : Amigamain		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Matthias Rustler	January 13, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Amigamain

1.1 About

```
                Generic.tg is an input file for TemGen. It creates the files  ↩
                   amigamain.c and amigamain.h.
You can switch on/off several common Amiga features like:
* Command line parameters (ReadArg)
* Tooltype parameters
* open shared libraries
* forbid start from Workbench or Shell
* localisation


                TemGen

                How to use

                Switches

                Configuration functions

                Functions

                struct Config

                Application exit

                Copyright

                Author
```

1.2 TemGen

```
The template generator 'TemGen' has its own documentation.

Some quick notes:
Commands start with '@' as first character in a line.
```

Some characters (#, $, ") have a special meaning. If they should appear in the
resulting code they must be prefixed with '\'. Example '\#include <stdio.h>'.

1.3 How to use

Copy the TemGen binary bin/tg to your search path.

Make a copy of generic.tg and open it in an editor.
Ajust the switches and functions to your needs. The switches are at the
beginning and the functions at the end of the #?.tg file.

Run 'tg' with the file. The files amigamain.c and amigamain.h are created.
Normally you shouldn't never change this files manually.

If you want locale support you have to make a copy of generic.cd and add
your strings. Don't remove the existing strings. Create #?_strings.h
for example with 'catcomp app.cd CTFILE app_strings.h'.

Create at least one #?.c file with the application code.

Build the binary.

1.4 Switches

You can switch on/off several features. Most of them are bool values. Don't
forget to prefix true and false with a '$' character.

@ $version = "MyApp 0.1 (1.1.2004)"
Version string (without '\VER:')
Use "" if you don't want a version string

@ $application = ["APP_run()", "APP_clean()", "app.h"]
The first function is called in 'main' after the libraries have successfully
been opened. The second function is called in the cleanup function.
The third entry is the name of file which is included with '#include "..."'.

@ $forbidwb = $false
Set this to $true if you don't want the application to start from Workbench

@ $forbidcli = $false
Set this to $true if you don't want the application to start from Shell

@ $opencon = $true
If you start your application from Workbench your input/output to stdin, stdout,
Output() and Input() would go to NIL:. $true opens con: windows in auto mode
for stdin, stdout and Output(). This means that up to 3 con: windows are opened.
This is ugly but it's indended as a backdoor if a Workbench application
prints to console.

@ $delay = 200
The con: windows created with $opencon are closed in the cleanup function.
The delay time in 1/50 seconds gives the user some time to read the content

```
of the con: windows before they are closed

@ $message_request = $true
Set this to $true if the message functions shall open an EasyRequest.

@ $message_output = $true
Set this to $true if the message functions shall print to Output().

@ $request_title = "APP Problem"
This text is shown as title in the Easyrequest of the message functions
if you don't have locale support.

@ $tooltypes = $true
$true inserts code to read tooltype arguments

@ $readargs = $true
$true inserts code to read command line arguments

@ $locale = $true
$true switches on locale support.

@ $loc_builtin = "english"
@ $loc_header = "app_strings.h"
@ $loc_catalog = "app.catalog"
This are additional parameters for locale support.
```

1.5 Configuration functions

```
@ $library( Basevariable, Libraryname, Minversion , Basetype )
Inserts code to open and close a shared library.
Use "" for Basetype if it's "struct Library *"

Examples:
@ $library( "WorkbenchBase" , "workbench.library" , 40 , "" )
@ $library( "LocaleBase"    , "locale.library"    , 40 , "LocaleBase" )
@ $library( "TimerBase"     , ""                  , 0  , "")
The last one only creates the base variable.

Note:
Most available libraries are already in generig.tg. Just remove the
commend char '#' to switch them on.

@ $add_config( type, variable name, default value)
Adds a member to struct Config.
Type can be "L" (LONG), "B" (BOOL) or "S" (STRING)

Examples:
@ $add_config("L", "xvalue", 5)
@ $add_config("B", "print", "TRUE")
@ $add_config("S", "pubscreen", "")

@ $add_tooltype( tooltype , config variable, type)
Querys the icon for a tooltype. If found it is stored in struct Config.
You have to create the config variable with $add_config. The type
of the tooltype must be the same as the type of the config variable.
```

```
Tooltypes are case sensitive.

Examples:
@ $add_tooltype("PUBSCREEN", "pubscreen" , "S")
@ $add_tooltype("PRINT"    , "print"     , "B")
@ $add_tooltype("XVALUE"   , "xvalue"    , "L")

@ $add_arg( arg, config variable, type)
Adds a single entry to the readarg template. If found it is stored
in struct Config. You have to create the config variable with $add_config.
The type of arg must be the same as the type of the config variable.

Examples:
@ $add_arg( "P=PUBSCREEN/K", "pubscreen", "S")
@ $add_arg( "XVALUE/N"     , "xvalue"   , "L")
@ $add_arg( "PRINT/S"      , "print"    , "B")

@ $finish_arg()
This function must be called after the last $add_arg
```

1.6 Functions

```
The resulting amigamain.c contains this functions:

LONG show_request( char *title, char *text, char *button, ... );
Opens an EasyRequest.

Example:
show_request("Error", "Couldn't open %s %ld", "OK", "foo.library", 32);

LONG show_request_args( char *title, char *text, char *button, va_list ap );
Like show_request, but with va_list

void vmessagef(char *format, va_list ap);
Like vprintf, but you can adjust with switches whether output goes to
Easyrequest and/or Output().

void messagef(char *format, ... );
Like printf, but you can adjust with switches whether output goes to
Easyrequest and/or Output().

void messagef_loc(LONG msgid, ...);
Like messagef, but with msgid as parameter. This function is only created
if you set $locale to $true.

char *strcpy_malloc(const char *s );
Allocates some RAM and copies the string s to it.

Important Note
All vararg parameters must be 32 bit. Use %ld instead of %d.
```

1.7 Config

amigamain.h contains the struct Config. The following entries are always created:

BOOL all_libraries_open:
Readonly; TRUE after all libraries have successfully been opended

BOOL start_from_wb
Readonly; TRUE if application was started from Workbench

struct Window *reqwin
Read/Write: reference window for Easyrequest

BOOL message_request
Read/Write: TRUE if message functions shall open EasyRequest

BOOL message_output
Read/Write: TRUE if message functions shall print to Output()

Additional entries are created with $add_config

The variable 'config' is created in amigamain.c

1.8 Exit

The cleanup function is used with 'atexit'. If you want to exit your application just call 'exit(EXIT_SUCCESS);' and the cleanup function is automatically called. Query config.all_libraries_open to avoid calling library functions if the libraries couldn't be opened.

1.9 Copyright

TemGen is open source (GPL) by Marek Letowski

generic.tg and the examples are public domain.

1.10 Author

Please report bugs, changes and feature requests to:

Matthias Rustler
MAIL: mrustler@t-online.de
WWW: http://mrustler.bei.t-online.de