AqBanking - Bug #256

gcttol and other tools do not show all available commands

16.02.2022 09:33 - rhabacker

Status:	Feedback	**Beginn:**	16.02.2022
Priorität:	Normal	**Abgabedatum:**	
Kategorie:	Gwenhywfar		
Betriebssystem:	FreeBSD, Linux, MacOSX, Windows, andere	**Anwendung:**	andere, aqbanking-cli
AqBanking-Version:	6.4.0	**Version der Anwendung:**	6.4.0

Beschreibung

Followup from https://www.aquamaniac.de/rdm/issues/254:

> When I looked more closely at the sources of gwenhywfar, I discovered additional command line applications where not all available commands/tests are displayed, such as: gcttool. It occurred to me that these applications could benefit from a commonly available interface for defining commands or tests.

Historie

#1 - 16.02.2022 09:43 - rhabacker

- Datei 0002-Refactor-gwentest-to-use-GWEN_Funcs_xxx-functions.patch wurde hinzugefügt

- Datei 0003-Refactor-gcttool-to-use-GWEN_Funcs_xxx-functions.patch wurde hinzugefügt

- Datei 0001-Add-GWEN_Funcs_xxx-functions.patch wurde hinzugefügt

Patches appended

#2 - 16.02.2022 09:56 - martin

- Status wurde von New zu Feedback geändert

Nice work, thanks!

One question: In gcttool/Makefile.am you changed added this by adding "$(top_builddir)/src/base/libbase.la":

```
gct_tool_LDADD=$(top_builddir)/src/$(gwenhywfar_internal_libname) $(top_builddir)/src/base/libbase.la
```

Why is this necessary? Libgwenhywfar.so already contains libbase.a?

Regards
Martin

#3 - 16.02.2022 10:01 - rhabacker

Without that library added on Linux I get:

/bin/sh ../../libtool --tag=CC   --mode=link gcc -g -O2 -Wall -Wdeclaration-after-statement -Wall -Wdeclaration-after-statement -Wall -Wdeclaration-after-statement   -o gct-tool activatekey.o changepin.o create.o genkey.o showkey.o showuser.o update.o setsignseq.o setkey.o hashtree.o checktree.o showpasswords.o main.o ../../src/libgwenhywfar.la -lssl -lcrypto
libtool: link: gcc -g -O2 -Wall -Wdeclaration-after-statement -Wall -Wdeclaration-after-statement -Wall -Wdeclaration-after-statement -o .libs/gct-tool activatekey.o changepin.o create.o genkey.o showkey.o showuser.o update.o setsignseq.o setkey.o hashtree.o checktree.o showpasswords.o main.o ../../src/.libs/libgwenhywfar.so -L/usr/lib64 -lgcrypt -ldl -lgpg-error -lgnutls -lssl -lcrypto -pthread
/usr/lib64/gcc/x86_64-suse-linux/7/../../../../x86_64-suse-linux/bin/ld: main.o: in function `main':
/home/ralf.habacker/src/gwenhywfar-build/tools/gcttool/../../../gwenhywfar/tools/gcttool/main.c:244: undefined reference to `GWEN_Funcs_Find'
/usr/lib64/gcc/x86_64-suse-linux/7/../../../../x86_64-suse-linux/bin/ld:
/home/ralf.habacker/src/gwenhywfar-build/tools/gcttool/../../../gwenhywfar/tools/gcttool/main.c:230: undefined reference to `GWEN_Funcs_Usage_With_Help

#4 - 16.02.2022 10:10 - martin

- Status wurde von Feedback zu Closed geändert

Ah, I see the problem (fixed in GIT).

In funcs.c you need to include "config.h", otherwise GWENHYWFAR_API is not properly declared and the symbol not exported.

I guess the idea is to have the new funcs module available to other tools (like aqbanking-cli) external to gwen, for this to work the symbols must be exported.

Also, there were some compiler warnings about discarded "const", fixed that, too.

Thanks for the work, that will surely improve the tools!

#5 - 16.02.2022 10:28 - martin

- Status wurde von Closed zu Feedback geändert

Hmm, maybe we can add some shortcuts for the macros?

"GWEN_Funcs_Entry_DB_NODE_Args_Help" already takes half of a editor line width ;-)

What About something like
- GWEN_FE_DAH for GWEN_Funcs_Entry_DB_NODE_Args_Help
- GWEN_FE_DA for GWEN_Funcs_Entry_DB_NODE_Args
- GWEN_FE_D for GWEN_Funcs_Entry_DB_NODE

Or GWEN_FNENTRY_DAH etc...

What do you think?

#6 - 16.02.2022 10:29 - martin

Also, normally we have macros in all-caps in gwen/aqbanking...

#7 - 16.02.2022 12:04 - rhabacker

> Also, normally we have macros in all-caps in gwen/aqbanking...

This macros could be be easily converted to real functions either inlined as shown below

```
/* Checks if a command variant exists */
#define GWEN_Funcs_Has_Call(func) (func)>func1
#define GWEN_Funcs_Has_Call_Args(func) (func)>func2
#define GWEN_Funcs_Has_Call_DB_NODE_Args(func) (func)>func3
+inline int GWEN_Funcs_Has_Call(const GWEN_FUNCS *func) { return func->func1 != NULL; }
+inline int GWEN_Funcs_Has_Call_Args(const GWEN_FUNCS *func) { return func->func2 != NULL; }
+inline int GWEN_Funcs_Has_Call_DB_NODE_Args(const GWEN_FUNCS *func) { return func->func3 != NULL; }

/* Functions to call a specified command */
#define GWEN_Funcs_Call(func) (func>func1)()
#define GWEN_Funcs_Call_Args(func,a,b) (func>func2)(a,b)
#define GWEN_Funcs_Call_DB_NODE_Args(func,a,b,c) (func>func3)(a,b,c)
+inline int GWEN_Funcs_Call(const GWEN_FUNCS *func) { return func->func1(); }
+inline int GWEN_Funcs_Call_Args(const GWEN_FUNCS *func, int argc, char **argv) { return func->func2(argc, arg
v); }
+inline int GWEN_Funcs_Call_DB_NODE_Args(const GWEN_FUNCS *func, GWEN_DB_NODE *node, int argc, char **argv) {
return func->func3(node, argc, argv); }
```

or as real functions, which would make them more binary safe.

#8 - 16.02.2022 12:12 - martin

And how about the shorter element macros proposed above?

#9 - 16.02.2022 12:28 - rhabacker

- Datei 0001-Convert-some-GWEN_FUNC_.-macros-to-real-functions.patch wurde hinzugefügt

- Datei 0002-Add-shortcuts-to-GWEN_Func_Entry-macros.patch wurde hinzugefügt

see appended patches

#10 - 16.02.2022 12:31 - rhabacker

Would it be possible to use c++ here ? There would be some nice features available like polymorph functions and constructors as shown in the following uncomplete example.

```
#include <stdio.h>
#include <string.h>

typedef int (*func1)(void);
typedef int (*func2)(int, char**);
typedef int (*func3)(void*, int, char**);

class Funcs {
public:
    const char *_name;
    const char *_help;
    func1 _func1;
    func2 _func2;
    func3 _func3;
    Funcs() : _name(NULL), _func1(NULL) {}
    Funcs(const char *name, func1 func, const char *help = NULL) : _name(name), _func1(func), _func2(NULL), _f
unc3(NULL), _help(help)  {}
    Funcs(const char *name, func2 func, const char *help = NULL) : _name(name), _func1(NULL), _func2(func), _f
unc3(NULL), _help(help) {}
    Funcs(const char *name, func3 func, const char *help = NULL) : _name(name), _func1(NULL), _func2(NULL), _f
unc3(func), _help(help) {}

    static const Funcs * get(const Funcs *funcs, const char *name)
    {
        for(const Funcs *f = funcs; f->_name; f++) {
            if (strcmp(f->_name, name) == 0)
                return f;
        }
        return NULL;
    }

    int call() const
    {
        fprintf(stderr, "%s %s\n", __PRETTY_FUNCTION__ , _help);
        return _func1();
    }

    int call(int argc, char **argv) const
    {
        fprintf(stderr, "%s %s\n", __PRETTY_FUNCTION__ , _help);
        return _func2(argc, argv);
    }

        int call(void *node, int argc, char **argv) const
    {
        fprintf(stderr, "%s %s\n", __PRETTY_FUNCTION__ , _help);
        return _func3(node, argc, argv);
    }

};

int testfunc1(void) { return 0; }
int testfunc2(int argc, char **argv) { return 0; }
int testfunc3(void *func, int argc, char **argv) { return 0; }
int testfunc4(void *func, int argc, char **argv, int something) { return 0; }

Funcs funcs[] = {
        Funcs("entry1", testfunc1, "help1"),
        Funcs("entry2", testfunc2),
        Funcs("entry3", testfunc3, "help3"),
        // uncomment to see how the compiler catches that unsupported constructor
        //Funcs("entry4", testfunc4),
        Funcs(),
};

int main(int argc, char **argv)
{
    const Funcs *func = Funcs::get(funcs, "entry1");
    if (func)
        func->call();
    func = Funcs::get(funcs, "entry2");
    if (func)
```

```
        func->call(argc, argv);
    func = Funcs::get(funcs, "entry3");
    if (func)
        func->call(NULL, argc, argv);
    return 0;
}
```

The disadvantage is that all main source files of command line tools (where main() is included) must be migrated to C++ before.

#11 - 18.02.2022 00:28 - martin

Definately no. I want to stay with C, I really want to avoid the complications that arise from using C++ especially in such libraries as gwen.

Years ago I used to write in C++ (first participating in OpenHBCI, then writing OpenHBCI-TNG). I encountered many obscure problems especially when using libraries which could not be solved at that time, that's why I wrote the whole AqBanking family in C.

Probably those old problems have since been solved in gcc and stdc++, but I don't want to rewrite the existing and basically nicely working code.

#12 - 24.02.2022 23:03 - rhabacker

- Datei 0001-gct-tool-restore-showing-global-options-when-using-h.patch wurde hinzugefügt

#13 - 24.02.2022 23:51 - rhabacker

- Datei 0001-gct-tool-restore-showing-global-options-when-using-h.patch wurde gelöscht

#14 - 24.02.2022 23:52 - rhabacker

- Datei 0001-gcttool-restore-showing-global-options-when-using-he.patch wurde hinzugefügt

After adding similar support to gsa I noticed a small issue in gcttools, which is fixed by this patch.

#15 - 24.02.2022 23:56 - rhabacker

- Datei 0002-Attempt-to-fix-creating-translations.patch wurde hinzugefügt

- Datei 0003-Add-command-tool-msghack-which-is-not-packaged-at-le.patch wurde hinzugefügt

- Datei 0004-Add-missing-german-translation-for-gcttool.patch wurde hinzugefügt

Fixes for gcttool transations applied.

Patch 0002* is required to regenerate po/de.po with

```
make merge
```

for out of source builds. Patch 0003* is required at least on openSUSE to regenerate de.po as msghack is not available as package.

#16 - 25.02.2022 00:04 - rhabacker

- Datei 0001-gcttool-use-GWEN_FUNCS-shortcuts.patch wurde hinzugefügt

For gcttool use GWEN_Funcs related shortcuts

#17 - 25.02.2022 00:17 - rhabacker

- Datei 0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch wurde hinzugefügt

Refactor gsa tool to use GWEN_Funcs_xxx functions

#18 - 25.02.2022 00:18 - rhabacker

- Thema wurde von gcttol does not show all available commands zu gcttol and other tools do not show all available commands geändert

#19 - 09.08.2022 09:07 - rhabacker

> 0001-gcttool-restore-showing-global-options-when-using-he.patch

I think that this required patch was not applied to git master.

e43961fb7 tools/gct-tool/main.c (aquamaniac 2005-11-03 00:20:45 +0000 225) if (GWEN_Args_Usage(args, ubuf, GWEN_ArgsOutType_Txt)) {

35ce2dfde tools/gct-tool/main.c (aquamaniac 2005-07-23 09:39:42 +0000 226) fprintf(stderr, "ERROR: Could not create help string\n");
35ce2dfde tools/gct-tool/main.c (aquamaniac 2005-07-23 09:39:42 +0000 227) return 1;
35ce2dfde tools/gct-tool/main.c (aquamaniac 2005-07-23 09:39:42 +0000 228) }

The called function GWEN_Args_Usage() only appends to the given ubuf argument and depends on the following lines to output the string, which are provided by the mentioned patch.

06c85b206 tools/gcttool/main.c (Ralf Habacker 2022-02-24 22:40:02 +0100 229) fprintf(stderr, "%s\n", GWEN_Buffer_GetStart(ubuf));
06c85b206 tools/gcttool/main.c (Ralf Habacker 2022-02-24 22:40:02 +0100 230) GWEN_Buffer_free(ubuf);

0001-gcttool-use-GWEN_FUNCS-shortcuts.patch
0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch

These patches were also not applied.

#20 - 10.08.2022 17:26 - rhabacker

- Datei 0002-Attempt-to-fix-creating-translations.patch wurde gelöscht

#21 - 10.08.2022 17:26 - rhabacker

- Datei 0003-Add-command-tool-msghack-which-is-not-packaged-at-le.patch wurde gelöscht

#22 - 10.08.2022 17:26 - rhabacker

- Datei 0004-Add-missing-german-translation-for-gcttool.patch wurde gelöscht

#23 - 10.08.2022 17:26 - rhabacker

The translation related patches has been moved to https://www.aquamaniac.de/rdm/issues/265

#24 - 10.08.2022 20:10 - martin

The following patches where rejected bei git:
- 0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch
- 0001-gcttool-use-GWEN_FUNCS-shortcuts.patch

Mostly because of changes in de.po.

#25 - 17.08.2022 21:18 - rhabacker

- Datei 0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch wurde hinzugefügt

Rebased patch

#26 - 17.08.2022 21:18 - rhabacker

- Datei 0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch wurde gelöscht

#27 - 17.08.2022 21:20 - rhabacker

- 0001-gcttool-use-GWEN_FUNCS-shortcuts.patch

Seems to be applied with https://www.aquamaniac.de/rdm/projects/gwenhywfar/repository/revisions/80863a328553ed54fa62ad9720e68545afba45c7

#28 - 17.08.2022 21:22 - martin

New patch applied to git, thanks!

Regards
Martin

Dateien

0002-Refactor-gwentest-to-use-GWEN_Funcs_xxx-functions.patch	10,6 KB	16.02.2022	rhabacker
0003-Refactor-gcttool-to-use-GWEN_Funcs_xxx-functions.patch	7,74 KB	16.02.2022	rhabacker
0001-Add-GWEN_Funcs_xxx-functions.patch	8,28 KB	16.02.2022	rhabacker
0002-Add-shortcuts-to-GWEN_Func_Entry-macros.patch	912 Bytes	16.02.2022	rhabacker
0001-Convert-some-GWEN_FUNC_.-macros-to-real-functions.patch	2,45 KB	16.02.2022	rhabacker

0001-gcttool-restore-showing-global-options-when-using-he.patch	795 Bytes	24.02.2022	rhabacker
0001-gcttool-use-GWEN_FUNCS-shortcuts.patch	2,67 KB	24.02.2022	rhabacker
0001-Refactor-gsa-tool-to-use-GWEN_Funcs_xxx-functions.patch	6,7 KB	17.08.2022	rhabacker