test_harness.h

KISS unit testing for C

Thilo Fromm, 02/2011

Agenda

Scope

Common Tasks

test_harness.h

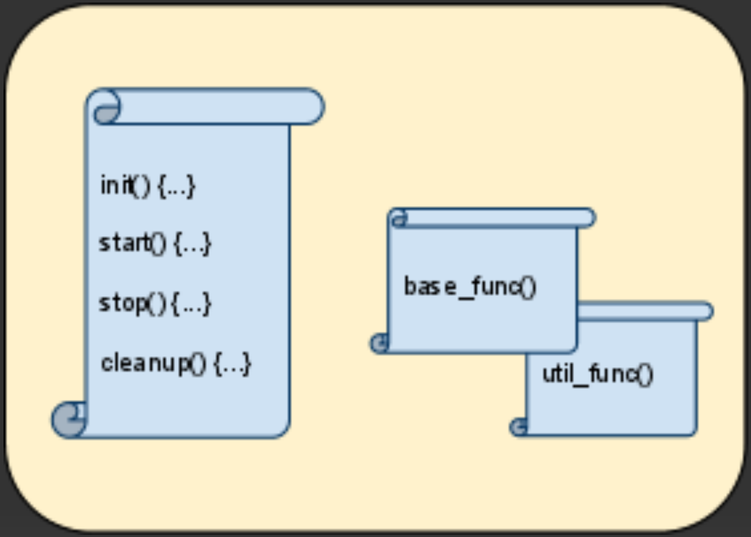
Q&A

Scope

What is a "Unit Test"?

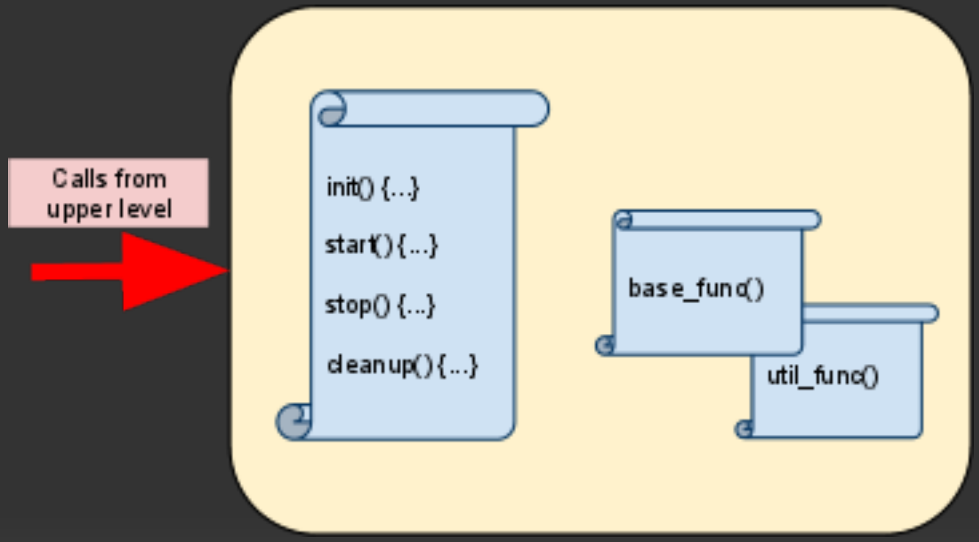
Scope

Unit: comprehensive set of specialized functions



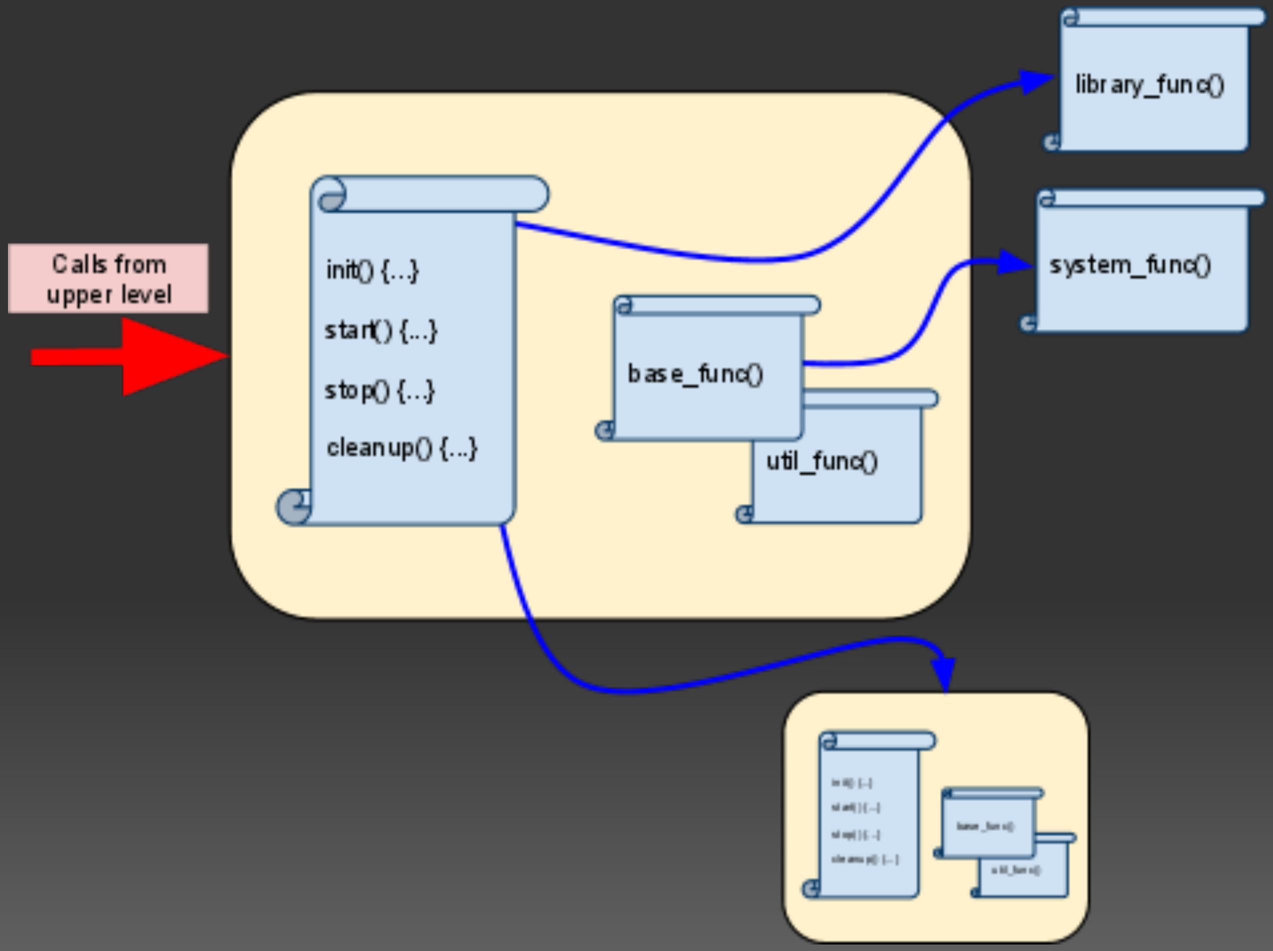
Scope

Unit acts on higher level stimuli



Scope

Unit utilizes lower level libs and other units



Common Tasks

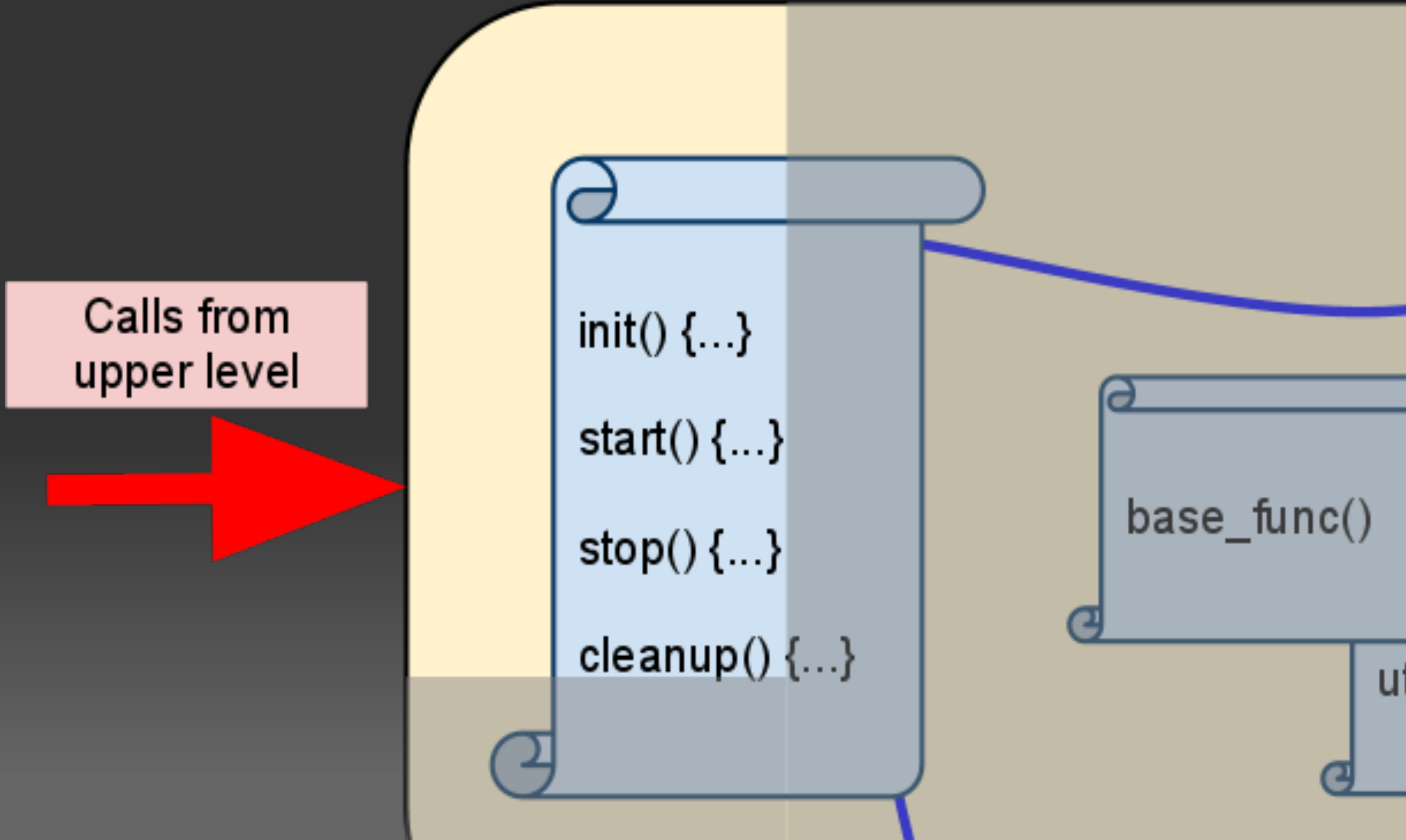
Things to provide in every unit test ever

Stimuli

Mocked back-end functions

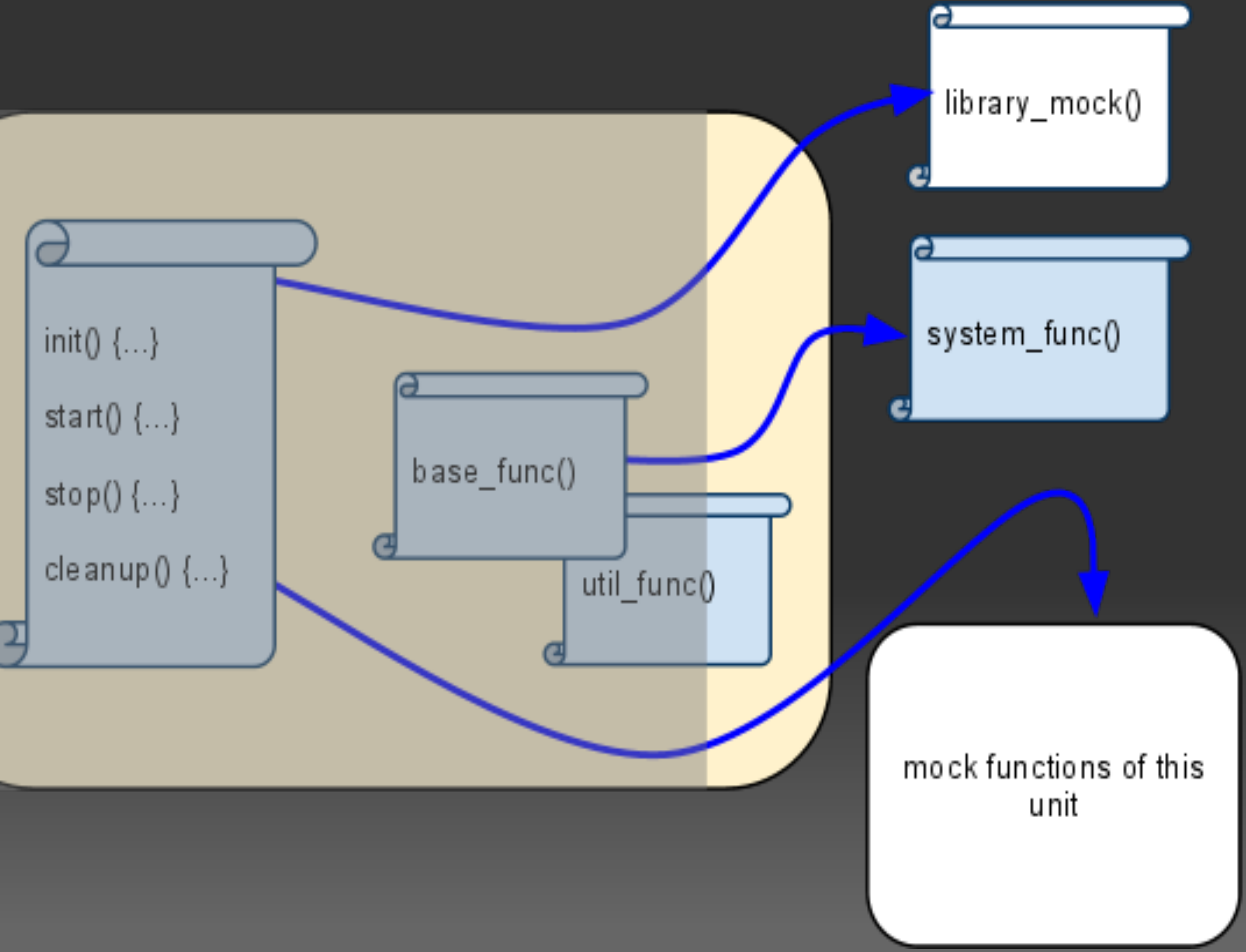
Common Tasks

Stimuli



Common Tasks

Mock Functions



test_harness.h

A collection of unit testing and mocking tools.

Implemented in the C preprocessor.

test_harness.h
Example unit test

```
#include <stdio.h>



int main(int argc, char ** argv)
{
    /* our main test routine */


    return 0;
}
```

test_harness.h
Access to higher level functions

```
#include <stdio.h>
#include "server_unit.c"


int main(int argc, char ** argv)
{
    /* our main test routine */


    return 0;
}
```

test_harness.h
Access to higher level functions' dependencies

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"

int main(int argc, char ** argv)
{
    /* our main test routine */


    return 0;
}
```

test_harness.h
Using a stimulus

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"

int main(int argc, char ** argv)
{
   /* our main test routine */
   int ret = server_unit_init("localhost", 12);

   return 0;
}
```

test_harness.h
test compile

```
:(.text+0xf): undefined reference to `library_func'
:(.text+0x1e): undefined reference to `sub_unit_func'
:(.text+0x2d): undefined reference to `sub_unit_tool_func'
collect2: ld returned 1 exit status
```

test_harness.h
Implementing a mock function

```
#include "test_harness.h"
```

test_harness.h
Implementing a mock function

```
#include "test_harness.h"

// int library_func( char * arg );
MOCK_1 (int,    library_func, char *);
```

test_harness.h
Implementing a mock function

```
#include "test_harness.h"

// int library_func( char * arg );
MOCK_1 (int,    library_func, char *);

// void* sub_unit_func( int val, uint32_t len, st..., int dbg);
MOCK_4(void*, sub_unit_func, int,uint32_t,struct netbuf *,int);
```

test_harness.h
Implementing a mock function

```
#include "test_harness.h"

// int library_func( char * arg );
MOCK_1 (int,    library_func, char *);

//void* sub_unit_func( int val, uint32_t len, st..., int dbg);
MOCK_4(void*, sub_unit_func, int,uint32_t,struct netbuf *,int);
```

// void sub_tool_func(int count, void * bytes);
MOCK_2V(sub_unit_tool_func, int, void *);

test_harness.h
Integration

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
    /* our main test routine */
    int ret = server_unit_init("localhost", 12);

    return 0;
}
```

test_harness.h
Cleanup

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1(int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
   char * server_name = "localhost";




   TEST_ASSERT( 42, server_unit_init(server_name, 12), int );
   return 0;
}
```

test_harness.h
Mock function call setup

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
   char * server_name = "localhost";
   MOCK_4_CALL( 0xbaba, sub_unit_func, 10, 99, NULL, 1);



   TEST_ASSERT( 42, server_unit_init(server_name, 12), int );
   return 0;
}
```

test_harness.h
Mock function call setup

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
   char * server_name = "localhost";
   MOCK_4_CALL ( 0xbaba, sub_unit_func, 10, 99, NULL, 1);
   MOCK_2V_CALL( sub_unit_tool_func, 12, DONT_CHECK_PARAM );


   TEST_ASSERT( 42, server_unit_init(server_name, 12), int );
   return 0;
}
```

test_harness.h
Mock function call setup

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
   char * server_name = "localhost";
   MOCK_4_CALL ( 0xbaba, sub_unit_func, 10, 99, NULL, 1);
   MOCK_2V_CALL( sub_unit_tool_func, 12, DONT_CHECK_PARAM );
   MOCK_1_CALL ( 42, library_func, server_name );

   TEST_ASSERT( 42, server_unit_init(server_name, 12), int );
   return 0;
}
```

test_harness.h
Mock function call setup

```
#include <stdio.h>
#include "server_unit.c"
#include "server_lib.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);
MOCK_4(void*, sub_unit_func, int, uint32_t, struct netbuf *,int);
MOCK_2V(sub_unit_tool_func, int, void *);

int main(int argc, char ** argv)
{
   char * server_name = "localhost";
   MOCK_4_CALL ( 0xbaba, sub_unit_func, 10, 99, NULL, 1);
   MOCK_2V_CALL( sub_unit_tool_func, 12, DONT_CHECK_PARAM );
   MOCK_1_CALL ( 42, library_func, server_name );
   MOCK_2V_CALL( sub_unit_tool_func, 12, server_name );
   TEST_ASSERT( 42, server_unit_init(server_name, 12), int );
   return 0;
}
```

test_harness.h

Mock functions w/ logic?

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);







int main(int argc, char ** argv)
{






    return 0;
}
```

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);

static void my_lib_func( char * arg ) {


}

int main(int argc, char ** argv)
{






    return 0;
}
```

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);

static void my_lib_func( char * arg ) {
    if (! strcmp(arg, "bloedsinn"))
        MOCK_RETVAL_OF( library_func ) = 23;
}

int main(int argc, char ** argv)
{






    return 0;
}
```

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);

static void my_lib_func( char * arg ) {
    if (! strcmp(arg, "bloedsinn"))
        MOCK_RETVAL_OF( library_func ) = 23;
}

int main(int argc, char ** argv)
{
    _library_func_cb = my_lib_func;




    return 0;
}
```

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);

static void my_lib_func( char * arg ) {
    if (! strcmp(arg, "bloedsinn"))
        MOCK_RETVAL_OF( library_func ) = 23;
}

int main(int argc, char ** argv)
{
    _library_func_cb = my_lib_func;
    MOCK_1_CALL ( 19, library_func, NULL );
    MOCK_1_CALL ( 00, library_func, DONT_CHECK_PARAM );


    return 0;
}
```

test_harness.h
Installing callbacks into mocked functions

```
#include "server_unit.c"
#include "test_harness.h"

MOCK_1 (int,    library_func, char *);

static void my_lib_func( char * arg ) {
    if (! strcmp(arg, "bloedsinn"))
        MOCK_RETVAL_OF( library_func ) = 23;
}

int main(int argc, char ** argv)
{
    _library_func_cb = my_lib_func;
    MOCK_1_CALL ( 19, library_func, NULL );
    MOCK_1_CALL ( 00, library_func, DONT_CHECK_PARAM );
    TEST_ASSERT ( 23, server_unit_init("bloedsinn", 12), int );

    return 0;
}
```

test_harness.h

```
MOCK_1 (int, blarf_func, char *)
```

test_harness.h

MOCK_1 (int, blarf_func, char *):

```
long _blarf_func_configured_calls = -1;
long _blarf_func_called_count     = -1;
```

test_harness.h

MOCK_1 (int, blarf_func, char *):

```
long _blarf_func_configured_calls = -1;
long _blarf_func_called_count     = -1;

void   (*_blarf_func_cb)(char*);
```

test_harness.h

MOCK_1 (int, blarf_func, char *):

```
long _blarf_func_configured_calls = -1;
long _blarf_func_called_count     = -1;

void  (*_blarf_func_cb)(char*);
```

char * _blarf_func_exp_arg0[MAX_NUM_FUNC_CALL];
int   _blarf_func_ret   [MAX_NUM_FUNC_CALL];

test_harness.h

MOCK_1 (int, blarf_func, char *):

```
long _blarf_func_configured_calls = -1;
long _blarf_func_called_count     = -1;

void  (*_blarf_func_cb)(char*);

char * _blarf_func_exp_arg0[ MAX_NUM_FUNC_CALL ];
int    _blarf_func_ret     [ MAX_NUM_FUNC_CALL ];

int blarf_func( char * arg0 )
{
   _blarf_func_called_count ++;
   check_params_and_callback();
   return _blarf_func_ret[ _blarf_func_called_count ];
}
```

Q&A

Fragen?