test_harness.h

KISS unit testing for C

Thilo Fromm, 02/2011

Agenda

Scope

Common Tasks

`test_harness.h`

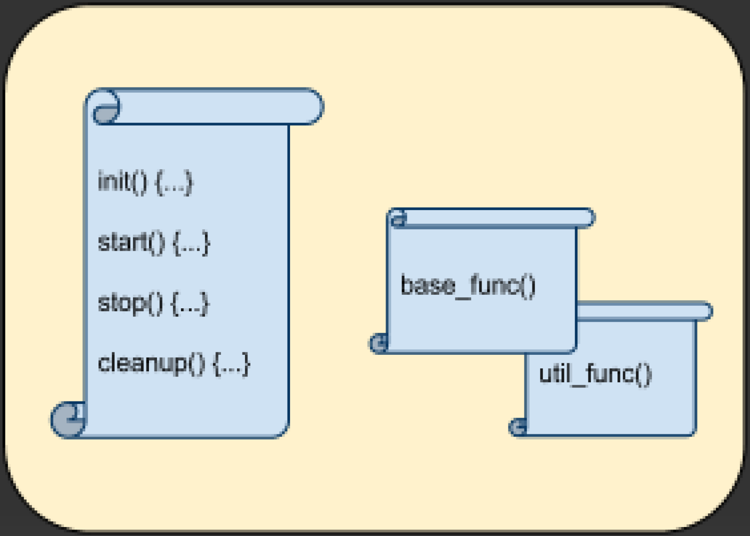
Q&A

Scope

What is a “Unit”?

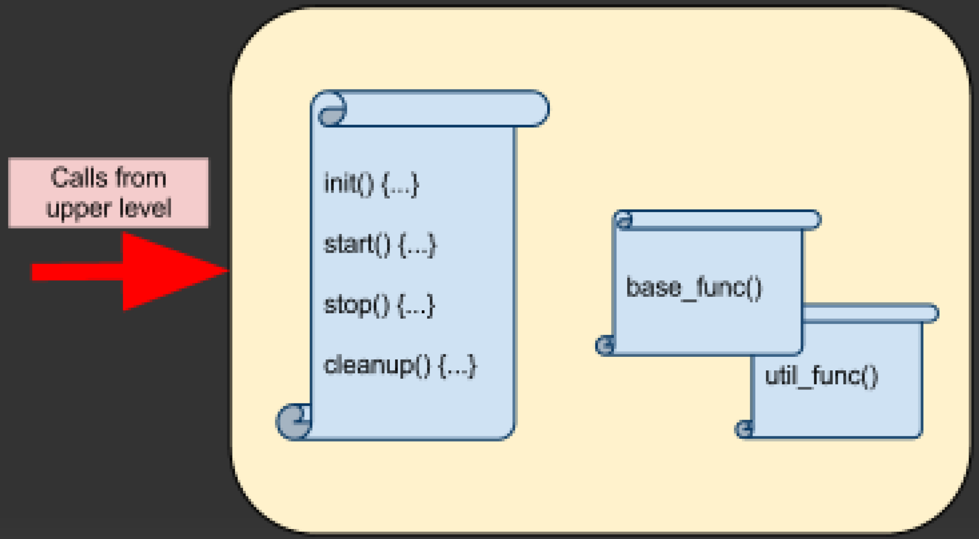
Scope

Unit: comprehensive set of specialized functions



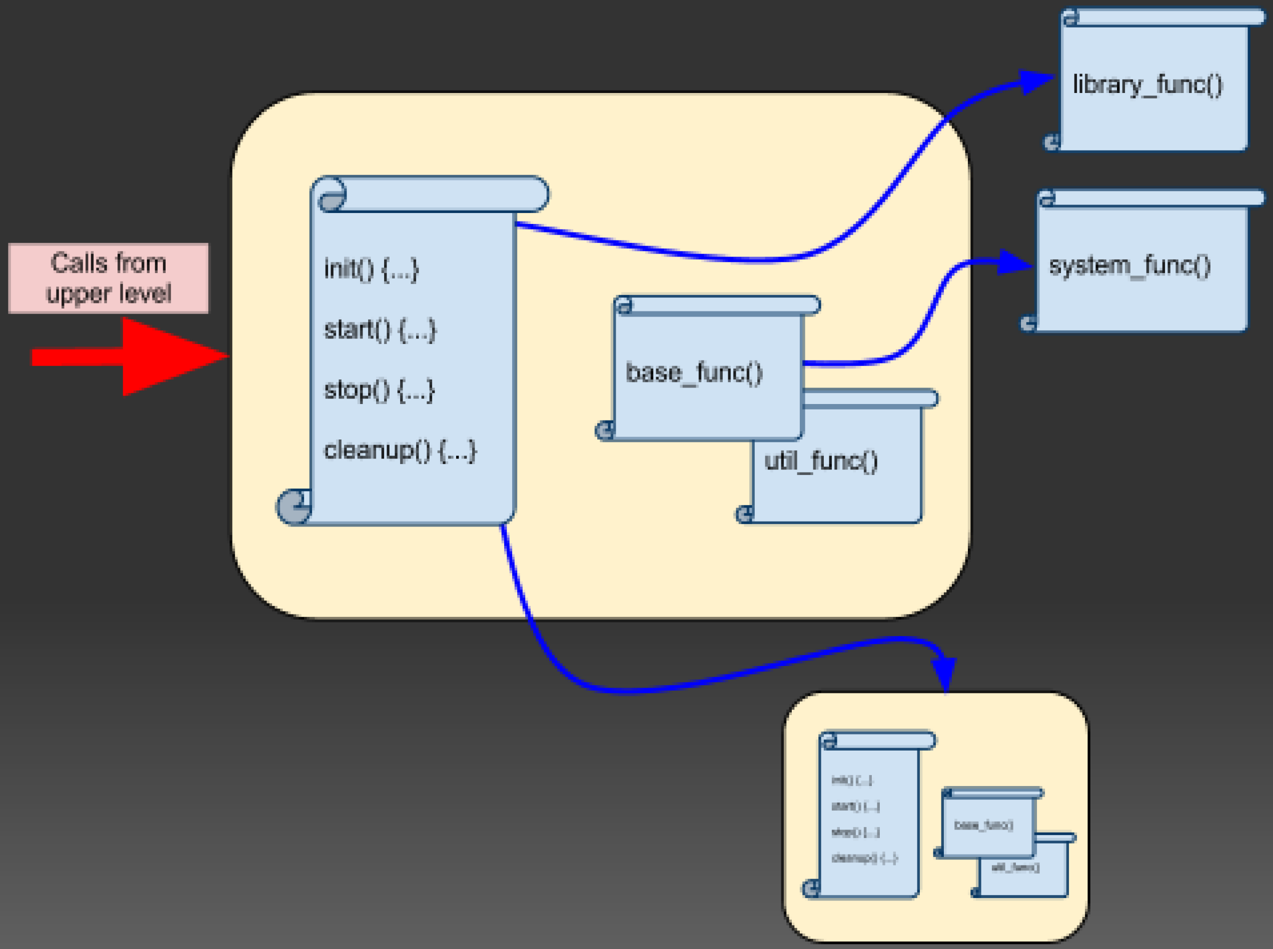
Scope

Unit acts on higher level stimuli



Scope

Unit utilizes lower level libs and other units



Scope

What is a “Unit”?

One Function (or a small set thereof)

- doing exactly one thing

- which can be separated

Common Tasks

Common Tasks

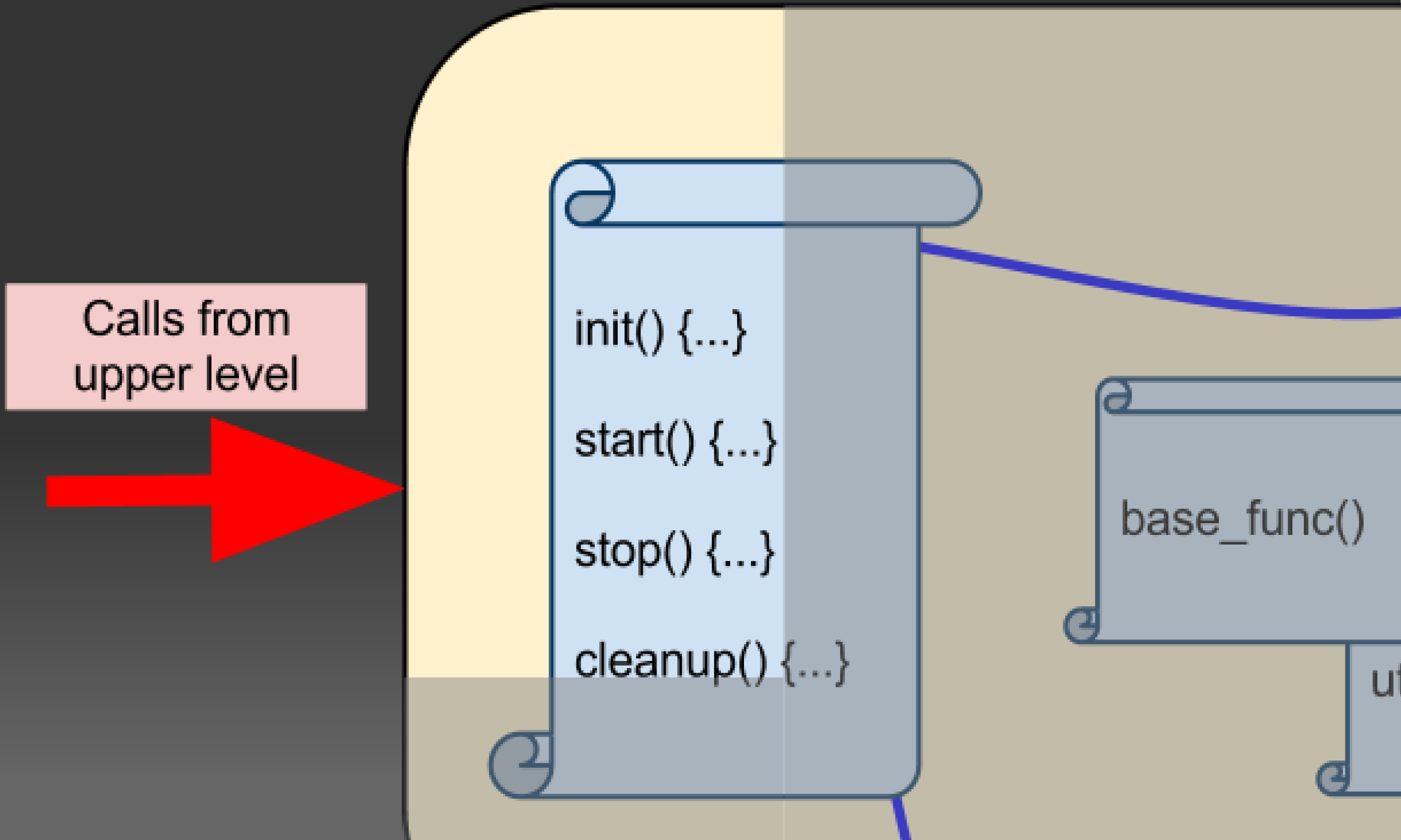
“Boilerplating”

What do I need in every new test I write?

How can I provide automation?

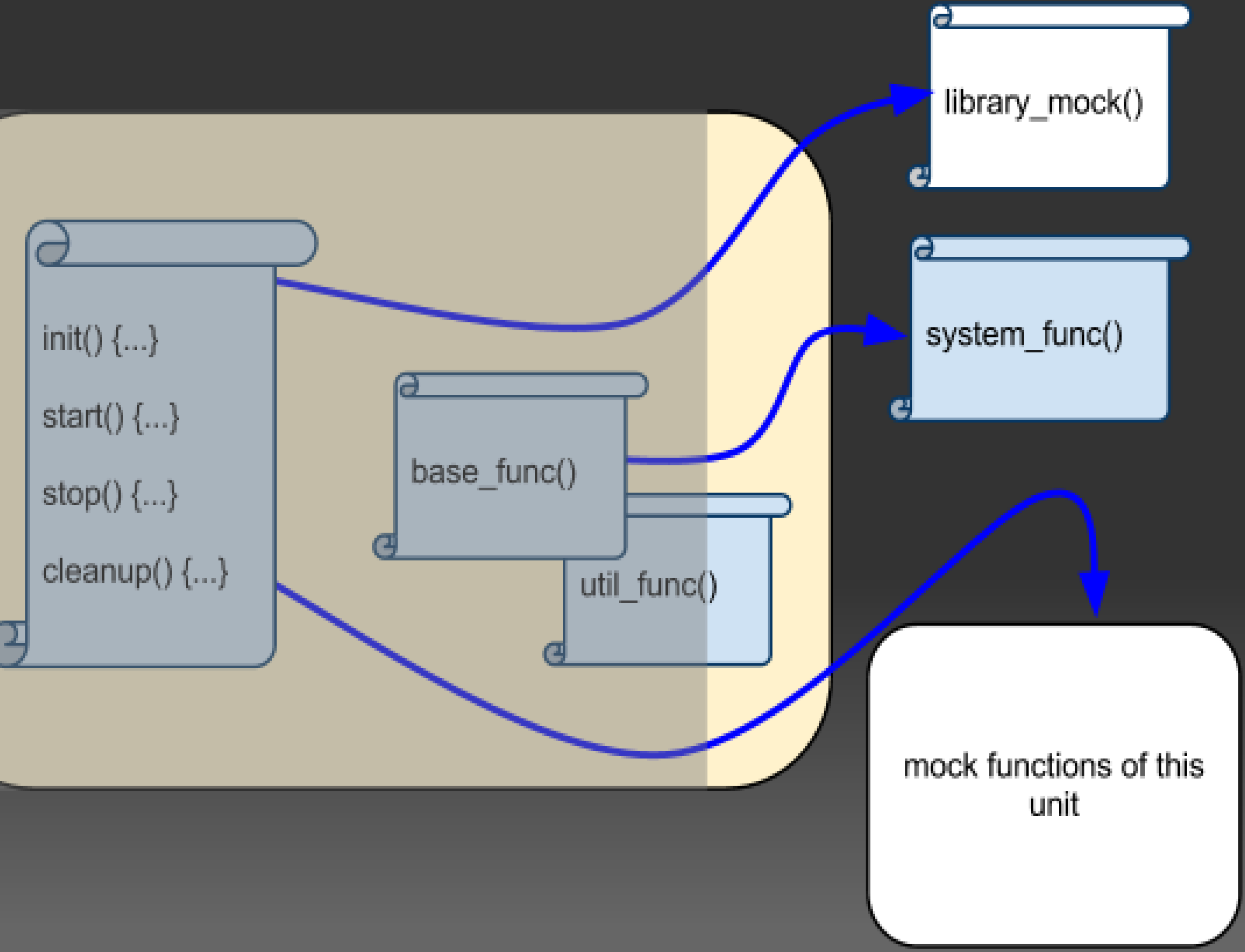
Common Tasks

Stimuli



Common Tasks

Mock Functions



introducing

`test_harness.h`

test_harness.h

A collection of unit testing and mocking tools.

Implemented in the C preprocessor.

`test_harness.h`

Let's do a quick unit test.

First, we include the Unit, and add a stimulus.

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

Now let's 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`

We need to mock functions used by the unit.

`test_harness.h` can help with this.

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

Mock 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)
{
    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

Need custom logic in mock functions?

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)
{
    MOCK_CB_SET( lib_func, 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)
{
    MOCK_CB_SET( lib_func, 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)
{
    MOCK_CB_SET( lib_func, 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_[n]( … )`

creates a mocked function.

test_harness.h

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

test_harness.h

MOCK_1 (int, example_func, char *):

```
    long _example_func_configured_calls = -1;
    long _example_func_called_count     = -1;
```

test_harness.h

```
MOCK_1 (int, example_func, char *):

    long _example_func_configured_calls = -1;
    long _example_func_called_count     = -1;

    void   (*_example_func_cb)(char*);
```

test_harness.h

```
MOCK_1 (int, example_func, char *):

    long _example_func_configured_calls = -1;
    long _example_func_called_count     = -1;

    void   (*_example_func_cb)(char*);

    char * _example_func_exp_arg0[ MAX_NUM_FUNC_CALL ];
    int    _example_func_ret     [ MAX_NUM_FUNC_CALL ];
```

test_harness.h

```
MOCK_1 (int, example_func, char *):

    long _example_func_configured_calls = -1;
    long _example_func_called_count     = -1;

    void   (*_example_func_cb)(char*);

    char * _example_func_exp_arg0[ MAX_NUM_FUNC_CALL ];
    int    _example_func_ret     [ MAX_NUM_FUNC_CALL ];

    int example_func( char * arg0 )
    {
        _example_func_called_count ++;
        check_params_and_callback( … );
        return
            _example_func_ret[ _example_func_called_count ];
    }
```

`test_harness.h`

`MOCK_[n]_CALL ( ... )` configures

Calls, Parameters, and return values

test_harness.h

```
MOCK_1_CALL (ret, example_func, param)
```

test_harness.h

```
MOCK_1_CALL (ret, example_func, param):

{
        _example_func_configured_calls ++;



}
```

test_harness.h

MOCK_1_CALL (ret, example_func, param):

```
{
        _example_func_configured_calls ++;
        _example_func_exp_arg0[ _example_func_configured_calls ]
                = param;

}
```

test_harness.h

MOCK_1_CALL (ret, example_func, param):

```
{
        _example_func_configured_calls ++;
        _example_func_exp_arg0[ _example_func_configured_calls ]
                = param;
        _example_func_ret[ _example_func_configured_calls] = ret;
}
```

`test_harness.h`

Stimuli: Access units by including the source

Mock functions: made easy w/ macros

Things not covered by

`test_harness.h`

and what to do about them

Things not covered by

`test_harness.h`

Code coverage – we use `gcov`

Graphics, statistics – we use `gcov` output :)

Q&A

Questions?