Generating and Contributing Test Cases for C Libraries from Client Code: A Case Study

Ahmed Zaki, Arindam Sharma, Cristian Cadar

SOFTWARE RELIABILITY GROUP

IMPERIAL

Motivation

Writing test cases for software libraries is non-trivial

Test cases might not represent real-world uses

Clients as a reference

- Clients of a library provide a rich source of usages for APIs
- Infeasible to integrate a full function from a client
  - Other library dependencies
  - Client specific dependencies

Objective

- Automatically generate test cases for APIs that
  - Have no dependencies except on the target library and system headers
  - Self contained (single file)
  - Readable

Library
Libiconv

Client
GnuPG

```
1  int set_native_charset (const char *newset){
2    const char *full_newset;
3    if (!newset)  {
4      static char codepage[30];
5      unsigned int cpno;
6      const char *aliases;
7      cpno = GetConsoleOutputCP ();
8      if (!cpno) cpno = GetACP ();
9      sprintf (codepage, "CP%u", cpno );
10     newset = codepage;
11     for (aliases = ("CP936" "\0" "GBK" "\0"...);
       *aliases; aliases += strlen (aliases) + 1,
       aliases += strlen (aliases) + 1){
12         if (!strcmp (codepage, aliases) || (*
       aliases == '*' && !aliases[1])){
13           newset = aliases + strlen (aliases) + 1
       ;
14           break;
15         }
16     }
17     newset = nl_langinfo (CODESET);
18   }
19   full_newset = newset;
20   if (strlen (newset) > 3
        && !ascii_memcasecmp (newset, "iso", 3))  {
21       newset += 3;
22       if (*newset == '-' || *newset == '_')  {
23           newset++;
24       }
25   }
26   if (!*newset
       || !ascii_strcasecmp (newset, "8859-1")
       || ...) {
27       active_charset_name = "iso-8859-1";
28       no_translation = 0;
29       use_iconv = 0;
30   } else{
31       iconv_t cd;
32       cd = iconv_open (full_newset, "utf-8");
33       if (cd == (iconv_t)-1){
34         handle_iconv_error (full_newset,
       "utf-8", 0);
35         return -1;
36       }
37
38       iconv_close (cd);
39       cd = iconv_open ("utf-8", full_newset);
40       if (cd == (iconv_t)-1){
41         handle_iconv_error ("utf-8",
       full_newset,0);
42         return -1;
43       }
44       iconv_close (cd);
45       active_charset_name = full_newset;
46       no_translation = 0;
47       use_iconv = 1;
48   }
49   return 0;
50 }
```

```c
int set_native_charset (const char *newset){
  const char *full_newset;
  if (!newset)  {
    static char codepage[30];
    unsigned int cpno;
    const char *aliases;
    cpno = GetConsoleOutputCP ();
    if (!cpno) cpno = GetACP ();
    sprintf (codepage, "CP%u", cpno );
    newset = codepage;
    for (aliases = ("CP936" "\0" "GBK" "\0"...);
    *aliases; aliases += strlen (aliases) + 1,
    aliases += strlen (aliases) + 1){

        if (!strcmp (codepage, aliases) || (*
    aliases == '*' && !aliases[1])){

          newset = aliases + strlen (aliases) +
    ;

          break;
        }
    }
    newset = nl_langinfo (CODESET);
  }
  full_newset = newset;
  if (strlen (newset) > 3
     && !ascii_memcasecmp (newset, "iso", 3))  {
        newset += 3;
        if (*newset == '-' || *newset == '_')  {
            newset++;
        }
  }
  if (!*newset
    || !ascii_strcasecmp (newset, "8859-1")
    || ...) {
        active_charset_name = "iso-8859-1";
        no_translation = 0;
        use_iconv = 0;
  } else{
        iconv_t cd;
        cd = iconv_open (full_newset, "utf-8");
        if (cd == (iconv_t)-1){
          handle_iconv_error (full_newset,
     "utf-8", 0);
          return -1;
        }

        iconv_close (cd);
        cd = iconv_open ("utf-8", full_newset);
       if (cd == (iconv_t)-1){
          handle_iconv_error ("utf-8",
    full_newset,0);
          return -1;
       }
        iconv_close (cd);
        active_charset_name = full_newset;
        no_translation = 0;
        use_iconv = 1;
  }
  return 0;
}
```

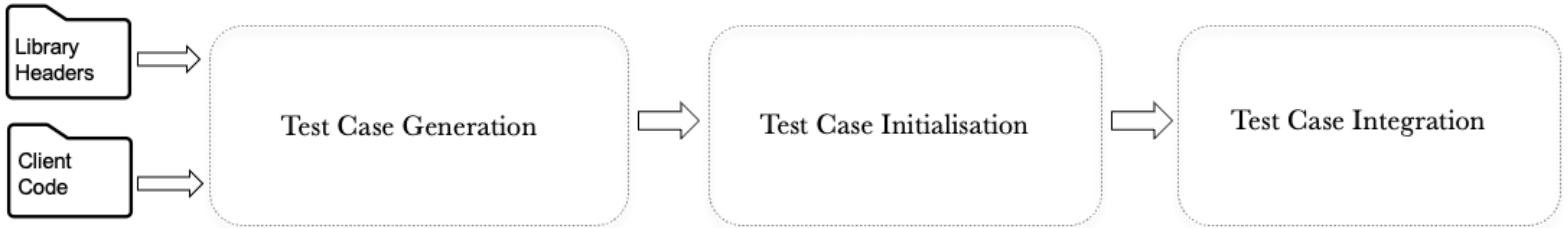
```c
int Test_set_native_charset() {
  // begin function parameters
  const char *newset = NULL;
  // end function parameters
  const char *full_newset;
  if (!newset) {
    newset = nl_langinfo(CODESET);
  }
  full_newset = newset;
  if (strlen(newset) > 3 && !ascii_memcasecmp(
    newset, "iso", 3)) {
    newset += 3;
    if (*newset == '-' || *newset == '_') {
    newset++; }
  }

  if (!*newset || !ascii_strcasecmp(newset, "
    8859-1") ||
      !ascii_strcasecmp(newset, "646") || !
    ascii_strcasecmp(newset, "ASCII") ||
      !ascii_strcasecmp(newset, "ANSI_X3.4-1968")
    ) {
  } else {
    libiconv_t cd;
    cd = libiconv_open(full_newset, "utf-8");

    if (cd == (libiconv_t)-1) {
      return -1;
    }
    libiconv_close(cd);
    cd = libiconv_open("utf-8", full_newset);

    if (cd == (libiconv_t)-1) {
      return -1;
    }
    libiconv_close(cd);
  }
}
```

APISlicer


Library Headers
Client Code
Test Case Generation
Test Case Initialisation
Test Case Integration

Test Case Generation

- Function Selection – one test case per client function that uses an API
  - Primitive Client
  - Primitive API
- Slicing
  - Reaching Definitions + Use-Def chain ≅ (PDG)
- Synthesis
  - Fetch type definitions
  - Construct a test case

Test Case Initialisation

```
1  int Test_set_native_charset() {
2    // begin function parameters
3    const char *newset = NULL;
4    // end function parameters
5    const char *full_newset;
6    if (!newset) {
7      newset = nl_langinfo(CODESET);
8    }
9    full_newset = newset;
10   if (strlen(newset) > 3 && !ascii_memcasecmp(
       newset, "iso", 3)) {
11     newset += 3;
12     if (*newset == '-' || *newset == '_') {
       newset++; }
13   }
14
15   if (!*newset || !ascii_strcasecmp(newset, "
       8859-1") ||
16       !ascii_strcasecmp(newset, "646") || !
       ascii_strcasecmp(newset, "ASCII") ||
17       !ascii_strcasecmp(newset, "ANSI_X3.4-1968")
       ) {
18   } else {
19     libiconv_t cd;
20     cd = libiconv_open(full_newset, "utf-8");
21
```

Instrument Client Functions

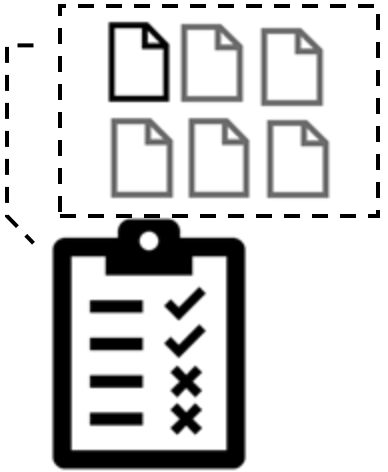
Execute Client Test Suite

Collect Inputs

Test Case Integration

Understand Library
Testing Framework

Integrate Test Case

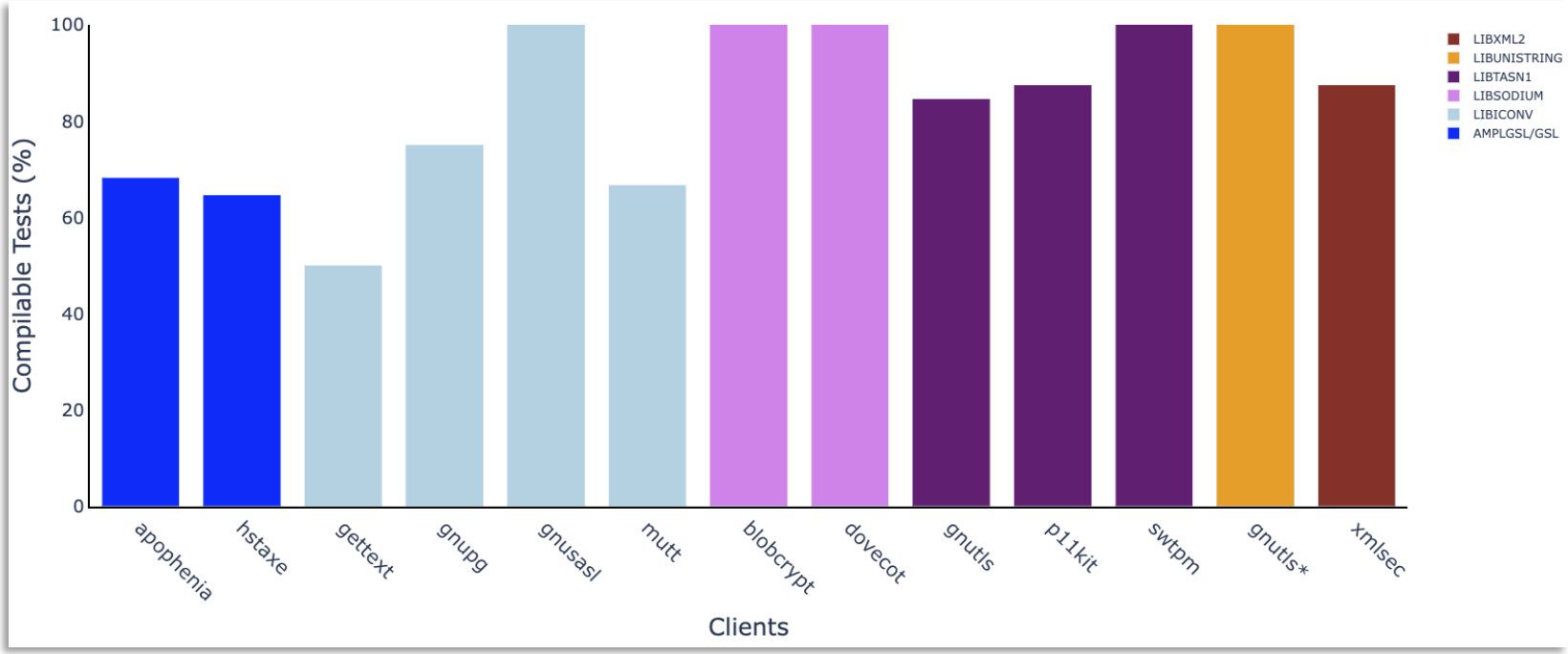


Case Study

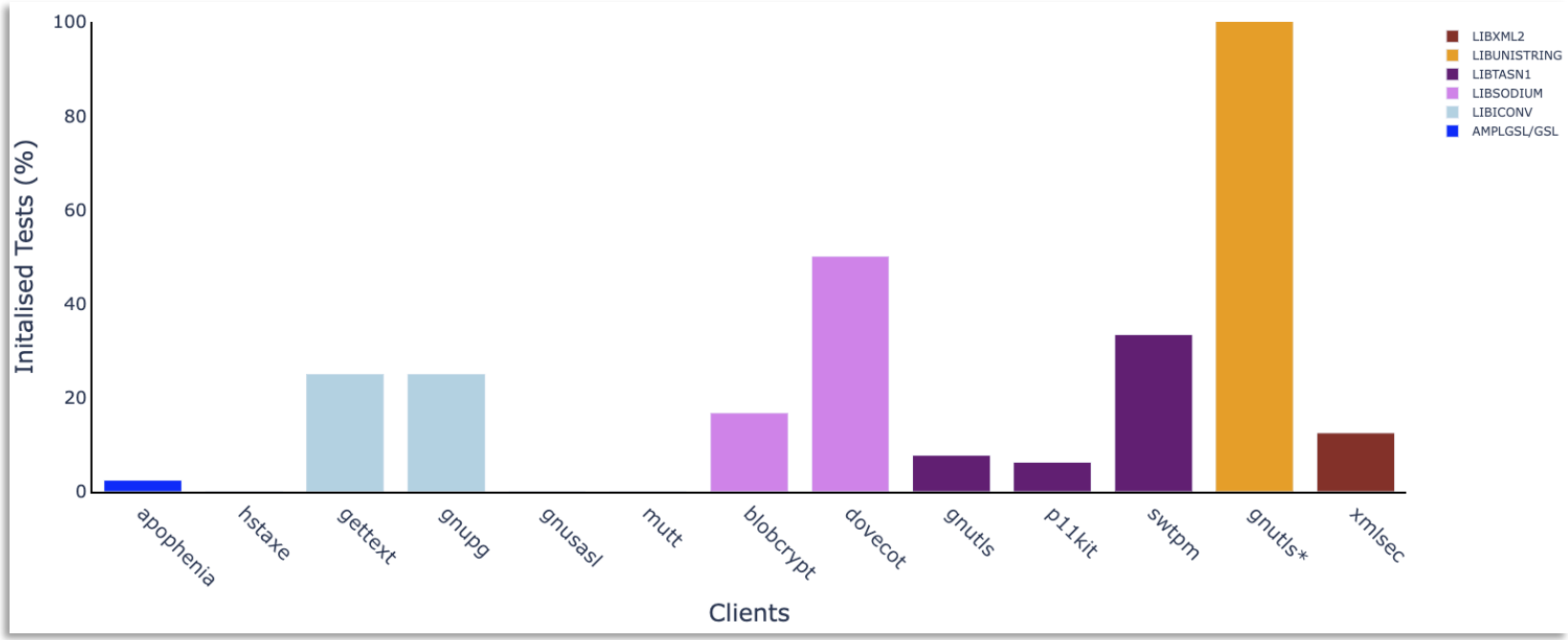
- 7 popular libraries
- 12 clients

Library	LoC	Client
AMPLGSL	71,547	APOPHENIA HSTAXE
GSL	66,130	APOPHENIA HSTAXE
LIBICONV	8,895	MUTT GNUSASL GETTEXT GNUPG
LIBSODIUM	10,289	DOVECOT BLOBCRYPT
LIBTASN1	4,322	GNUTLS P11KIT SWTPM
LIBUNISTRING	8,289	GNUTLS
LIBXML2	117,351	XMLSEC

Compilable Tests from Client Functions



Client Test-Suite Limitations



Developer Feedback

…we still have the licensing issue, contributions should be assigned to FSF and under the libtasn1 license.

Libtasn1

Thanks. Indeed, your test case takes a different code path in lib/unictype/categ_and_not.

Libunistring

…it would be much more useful to run the whole test suite of downstream projects with the latest libxml2 code. Some downstream projects already do that. But I'd also consider to test a few projects within libxml2's CI setup…

Libxml2

These tests don't test anything that is not already exercised by the current test suite

Libsodium

Thank you for your contribution, much appreciated!

GSL

Lessons

Generally, the number of generated test cases that compile was close to the number of candidate client functions.

Relying on client test suites to gather input values severely limits approach.

Increasing coverage is more important for library developers than realistic test cases.