ABI Compatibility for

Dummies

By,

Sinny kumari

{#,@}ksinny

fedora

What is API?

Application **P**rogramming **I**nterface

- Set of interfaces provided by an application to interact with a software.
- **For example:**
  - printf(), scanf() by libc.so library from glibc

    API declared in stdio.h header file (#include <stdio.h>)

```
extern int printf (const char *__restrict __format, ...);
extern int scanf (const char *__restrict __format, ...);
```

  - std::cout, std::cin by libstdc++.so library from gcc

    API declared in ostream header file (#include <ostream>)

```
namespace std
 {
   operator<<(__ostream_type& );
 }
```

Effect of API change

Applications may result in compilation error

Example

/* My application app.c */

```
#include <stdio.h>
int main()
{ printf ("Hello\n");  return 0;}
```

Compiling application app.c

$ gcc app.c

Prototype of printf in older version of glibc

```
extern int printf (const char *__restrict __format, ...);
```

Prototype of printf in newer version of glibc

```
extern int printf (const char *__restrict __format, int
arg_length, ...);
```

app.c will need to update source and recompile for using latest glibc

What is ABI?

Application **B**inary **I**nterface

Interface between two program modules, one of which is often a library or operating system, at the level of machine code.

What constitutes ABI change?

- Changes in calling stack
- V-table changes
- Removed functions/variables
- Removed libraries
- Change in size of data types

How ABI differs from API

- API is a source code-based while an ABI is a binary interface.
- POSIX is an API, while the Linux Standard Base provides an ABI
- If API of an application changes, dependent application need to update source code and recompile it.
- If ABI of an application changes, dependent application don't need to update source code but may need to recompile application.

What is ABI incompatibility

An application needs recompilation to run correctly, due to change in ABI from older to newer version of interface being used by it.

Effect of incompatible ABI

Wrong output or crash while running application

Benefits of having compatible ABI

- Reduced maintenance time
- Reduced cost
- Lesser bugs and crash in application
- Flexibility to migrate application from older to newer version of your distro like Fedora, Ubuntu, OpenSUSE

ABI analysis tools for Linux binaries

- Binutils
  - readelf - Displays information about binary files
    - debug information (option --debug-dump)
    - Exported function/variables (option -s)
    - Architecture, binary type, endianness (option -h)
  - objdump
  - nm- Lists symbols from object files
    - Exported symbols (option -g)
- Elfutils

Tools checking C/C++ ABI compatibility

- API Compliance Checker - Checks backward binary and source-level compatibility of a C/C++ library. Written in perl
- Libabigail – C++ application providing set of APIs and tools to do ABI analysis on binaries
  - abidiff – ABI change between two binaries
  - abidw – XML representation of ABI relevant data from binaries
  - abicompat – Checks ABI compatibility of an application against linked libraries during their subsequent release.

Summary

- While designing a library, think wisely and don't change APIs exposed by it frequently.
- As a consumer of library, while developing an application use libraries having stable ABI.
- Run ABI analysis tools like ACC or abidiff before releasing newer version of a library
- Run abicompat tool before upgrading your application to newer version of libraries it's linked to.

References

- ABI Compatibility Checker - http://ispras.linuxbase.org/index.php/ABI_compliance_checker
- Libabigail - https://sourceware.org/libabigail/
- Elfutils - https://fedorahosted.org/elfutils/
- Binutils - www.gnu.org/software/binutils/

Thank You