LLVM Libc: Status, Challenges and Future Plans

Siva Chandra Reddy
Google LLC
1. What is LLVM libc
2. Current Status
   a. Functions
   b. Loader
   c. Infrastructure
3. Challenges
4. Future Plans
LLVM libc

→ A greenfield libc developed with certain goals
  ◆ Sanitizer friendly
  ◆ Implemented in C/C++ source code without assembly
  ◆ Moduler

Full list of features and goals is available here:
http://llvm.org/docs/Proposals/LLVMLibC.html
Status: Code Metrics

Development was kicked off a year ago
- Over 10,000 lines of code from 200+ commits by 17 different contributors
- Over a 100 libc functions implemented
- Made a start to build a fully functional static-pie ELF loader
- Build, unittest, CI and other infrastructure setup
Functions from math.h

➢ Basic floating point operations
  ○ fabs[f|l], fdim[f|l], fmax[f|l], fmin[f|l]

➢ Nearest integer functions
  ○ ceil[f|l], floor[f|l], trunc[f|l], round[f|l]

➢ Floating point manipulation functions
  ○ copysign[f|l], frexp[f|l], logb[f|l], modf[f|l]

➢ Exponential functions
  ○ Single precision floating point versions of the exponential functions expf and exp2f

➢ Trigonometric functions
  ○ Single precision floating point versions of trigonometric functions sinf, cosf and sincosf

➢ Quotient and Remainder functions
  ○ remquo[f|l], remainder[f|l]

➢ Power functions
  ○ sqrt[f|l], hypotf

Credits: The exponential and trigonometric functions are adaptations of Arm’s contribution from their AOR project:
Memory Functions: `bzero`, `memcpy`, `memset`  
- Optimized for the statistically significant subset of inputs

Null terminated string functions: `memchr`, `memrchr`, `strcat`, `strchr`, `strcmp`, `strcpy`, `strcspn`, `strlen`, `strnlen`, `strpbrk`, `strrchr`, `strspn`, `strstr`, `strtok`, `strtok_r`

<table>
<thead>
<tr>
<th>Function</th>
<th>% of calls with size ≤ 128</th>
<th>% of calls with size ≤ 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>memcpy</td>
<td>96%</td>
<td>99%</td>
</tr>
<tr>
<td>memset</td>
<td>91%</td>
<td>99.9%</td>
</tr>
</tbody>
</table>
Status: Functions

● Functions from threads.h
  ○ call_once, mtx_init, mtx_lock, thrd_create, thrd_join (all for Linux)
  ○ Port them to non-linux platforms very soon.

● Functions from signal.h
  raise, sigaction, sigaddset, sigdelset, sigemptyset, sigfillset, signal, sigprocmask (all for Linux)

● Functions from ctype.h
  All ctype.h functions for the default locale
Miscellaneous Functions

Linux implementations of

- abort
- assert
- errno
- _Exit
- fwrite
- POSIX functions `mmap` and `munmap`
Loader

- A start has been made to build a static-pie ELF loader.
- Just enough has been built to be able to test thread functions.
- Near term goal is to build a full static-pie ELF loader.
Build Infrastructure

- Libc specific CMake rules have been implemented
- The rules are one of the core components which make the libc implementation modular.
- Libc targets added via libc specific CMake rules have Python like fully qualified names.
  - libc.src.math.sinf
Status: Infrastructure

Collection of Utils

- Unit test framework
- Standalone library of C++ utilities
- Template library of floating point operations
- MPFR as reference for math function testing
  - MPFR is a C library for multi-precision floating point operations: [http://mpfr.org](http://mpfr.org)
Status: Infrastructure

Header generation framework

- Standards are encapsulated in table gen files
- Each platform defines an API file
- The header generation tool reads the API file and generates the header files containing only the API listed in the API file
- Provides the pick the choose ability for header files

Note: A platform is a combination of the target OS + target machine architecture.
Clang-tidy checks

➢ Clang-tidy checks specific to LLVM libc have been implemented
➢ Protect against including undesired system headers
➢ Protect against polluting global namespace
➢ They run as part of the build and hence alert developers about deviant code at development time.
➢ They run on the public CI builders.
Redirectors

- A concept built for enabling the `mix-with-other-libcs` feature
- Redirectors are essentially wrapper functions in LLVM libc which intentionally call into the system libc
- Useful when an implementation of a particular function is not available yet in LLVM libc.

Status: Infrastructure
Challenges

➔ Ability to mix with other libcs
◆ Challenge: Avoid header file mixup
◆ Challenge: Avoid symbol mixup
◆ Challenge: Namespace pollution

Solution: libc specific clang-tidy checks

➔ Ability to pick only parts of LLVM libc
◆ Challenge: Avoid pulling in parts not required

Solution: Header generation + build system + redirectors
In the next one year

1. Complete the math library; that is, have an implementation for all functions coming from `math.h`.
2. Likewise, complete the null-terminated strings library.
3. Implement the API from `stdio.h` and `stdlib.h` at least for Linux.
4. Setup public CI builders for non-x86/non-linux platforms.
5. Implement full static-pie ELF loaders on Linux.
6. Finish the standard threads library (`threads.h`).
7. Setup infrastructure to compare the results and performance of the math functions with similar functions from other popular libcs.
8. Run LLVM libc fuzz tests on OSS fuzz:
   
   [https://github.com/google/oss-fuzz](https://github.com/google/oss-fuzz)
Thank You