POSIX Compliance

Google Summer of Code Program 2020 Project Proposal

Eshan Dhawan
eshandhawan51@gmail.com
Jaypee Institute of Information Technology, Noida
NCR, India
+91 9897368739
#rtems IRC: eshandhawan51

Project Abstract.

Portable Operating System Interface (POSIX) is a set of standards that are defined by the IEEE society. It defines various APIs along with command-line interfaces and utility interfaces to maintain software compatibility between various operating systems.
POSIX Compliance provides maximum flexibility to write applications that can be executed on various platforms and increases the portability of code. The RTEMS developers community has kept it well compliant with various standards. But with the advancement of time and technology, its up-gradation becomes an interminable task.
POSIX Compliance will require to add various APIs and different interfaces as specified by the POSIX IEEE Standard 1003.1TM. They can either be ported from other platforms with a compatible license or will be written from scratch as required. There would be a need to write architecture-specific code as well.
In RTEMS, Newlib provides the C library including libm, so for improving POSIX Compliance, we would also be required to add various functions and headers to Newlib as well. Newlib also requires math headers also

Project Description.

With the advancement of technology there comes an indefinite task of keeping up with the latest standards. The developers' community of the Real-Time Executive for Multiprocessor Systems (RTEMS), has kept the systems well compliant with various standards like POSIX IEEE Standard 1003.1™, C99, C11, and various profiles of FACE and SCA. This enables RTEMS to execute on various devices, architectures and environments.

But with non-stop advancements in technology, it becomes an interminable task, these standards keep updating and hence creates a need for up-gradation of their compliance status too.

Different embedded systems have different sets of capabilities, some real-time embedded systems like small embedded systems need a limited set of operating system functionality while others use a surprising set of methods and reusable libraries. Having more POSIX increases the ease of porting libraries. Which eases building the systems. That's the goal -- "Make it easier to port software". The stronger the software base, the easier it is to port software.

The goal of the project is to update the compliance status of RTEMS with POSIX IEEE and

FACE general purpose profile. This shall require the addition of various functions, headers, documentation and test-suites. They shall be added as the project advances.

Header files and related functions that are yet to be implemented

- **fenv.h**:-This header is designed to support the floating-point exception status flags and directed-rounding control modes required by the IEC 60559:1989 standard, and other similar floating-point state information. It has an architecture-specific implementation and its current implementation status can be found here #https://docs.google.com/document/d/1uKyFjzfTyUTd-CBtc0LgEiE5xSsUjTBzaV_8F3Ri7cs/edit?usp=sharing

- **File descriptor functions** :- All *at() functions need to be implemented, RTEMS has its own File Descriptor Implementation. It cannot be ported from other systems. It has to be implemented from scratch. List of missing functions can be found at List of missing RTEMS File descriptors
  Some implementation can be taken from https://github.com/lattera/freebsd/blob/master/contrib/openbsm/bin/auditdistd

- **wordexp.h**:- It contains definitions for word-expansion types. It contains two functions.Since the header cant be completely supported by RTEMS. So it can either be ported with tweaks in FreeBSD,musl or be written from scratch.Depending on what is favourable.

- **ftw.h**:- It contains the definition of File tree traversal. This defines the functions 'ftw()' and 'nftw()'. ftw() was declared obsolete by POSIX-2008 but still needs to be added for backward compatibility. Its implementation can be taken from musl.
  musl implementation:
  ftw.h : https://git.musl-libc.org/cgit/musl/tree/include/ftw.h
  ftw.c: https://git.musl-libc.org/cgit/musl/tree/src/legacy/ftw.c
  nftw.c: https://git.musl-libc.org/cgit/musl/tree/src/misc/nftw.c

- **confstr() from <unistd.h>** - Method to get configurable variables. It is missing from FACE GPP. It can be ported from https://github.com/udp/freebsd-libc/blob/master/gen/confstr.c.

- ***Port aio.h to Newlib****:- This header defines real-time asynchronous input and output. The header has been already implemented completely in RTEMS (present in cpukit/include/aio.h) but now needs to be added to Newlib.*

- ***Add functions to math.h****:- M*ost of the functions are implemented in math.h but still some functions need to be ported. The functions can be ported from FreeBSD The list of missing functions can be found here: Math.h missing functions

- ***Add tests for clock_nanosleep() and timer_create()***:- These functions are defined in time.h . They are already implemented but tests need to be added for them.There need to be tests added for timer_create() and clock_nanosleep() with CLOCK_MONOTONIC. I will also add tests for other supported functions of time.h whose tests are not present in the test suite.

- **fmtmsg.h** - Contains definitions for message display structures. fmtmsg is the only method under this header file. It will be ported from FreeBSD.

https://github.com/freebsd/freebsd/blob/master/include/fmtmsg.h
https://github.com/freebsd/freebsd/blob/master/lib/libc/gen/fmtmsg.c

- **search.h** - I will ensure that all search capabilities in newlib are enabled (newlib/libc/search) and will review the search.h capabilities against POSIX for completeness and compliance.

- **sys/ipc.h** - Contains definitions for XSI interprocess communication access structure. 'ftok()' is the only method in this header file. It will be ported from FreeBSD.
https://github.com/freebsd/freebsd/blob/master/sys/sys/ipc.h
https://github.com/freebsd/freebsd/blob/master/lib/libc/gen/ftok.c

Project Deliverables

GSoC Timeline: https://developers.google.com/open-source/gsoc/timeline

June 1 (coding begins) :

When the coding period begins I will ensure :

- ➢ All the Required BSP's build with completed hello world task
- ➢ I will have all the appropriate sources identified from where the code is to be ported or be written from scratch (as required)
- ➢ I would make myself well versed with the coding conventions to be followed in RTEMS and Newlib.
- ➢ Will have the necessary repositories forked on GitHub where my mentors can track my progress.
- ➢ Get well versed with git commands
- ➢ Add a few tests for present headers

June 29 - July 3 (Phase 1 Evaluation) –

- ➢ Implement **fenv.h** for **ARM** and **AARCH64**
- ➢ Implement **fenv.h** to **SPARC** (Will require implementation from the base since there is no related folder in libm/machine)
- ➢ Add **confstr()** from **unistd.h**
- ➢ Implementation of **fmtmsg.h**

July 27-31(Phase 2 Evaluation) –

- ➢ Implement **ftw.h**
- ➢ Add **file descriptor** functions (List of missing RTEMS File descriptors)
- ➢ Enable the **search.h** features

August 24-31 (Final Evaluation) -

- ➢ Port or write from scratch various **Math.h** functions.
(detailed plan is shared in project schedule)
- ➢ Implement **wordexp.h**
- ➢ Add **sys/ipc.h**

September 8 (Final Results Announced)

- ➢ Refine my code with the mentor's feedback.
- ➢ Update documentation for the changes I have made.
- ➢ Update my blog and share my experience with others.

Post GSOC - I have a keen interest in OS and kernel and RTEMS has everything related to it. I will continue my work with this community and try to gain more knowledge. I will get more involved with old and new projects and will try to help new developers join in future.

Proposed Schedule

March 16 – March 31 (Application Period)

- Join RTEMS developers mailing list.
- Build and Install RTEMS for required architectures.
- Complete 'HELLO WORLD' task for those architectures.
- Draft Proposal for the summer and ask the experts to review it.
- Set deliverables and project schedule as well as complete the pending parts
- Further, improve the proposal as per the suggestions from the community
- Set up my GitHub account for development and fork the required repositories.

March 31 - May 4 (Acceptance Waiting Period)

- Learn how to port code from FreeBSD as well as repositories with a compatible license
- Learn how to port architecture-specific code
- Also, learn how to write architecture-specific codes from scratch
- Get familiar with the RTEMS Coding Conventions.
- Go through the previous project related work
- Take a rough idea about various standards from their documentation.
- Also, go through their implementation in FreeBSD and NetBSD.

May 4 - June 1 (Community Bonding Period)

- Join RTEMS and Newlib mailing lists and get involved with the mentors and other experts present.
- Get familiar with the process of contribution to the organizations.
- Plan my complete day to day tasks after complete discussion with the mentors.
- Implement some basic Macros, constants and integers to Newlib

June 1- July 3 (Phase 1

Fenv.h will be implemented for ARM, AARCH, SPARC and PowerPC and other available architectures. I will need to set up toolchains of the BSPs' for the purpose of testing the implementation.fenv has an architecture-specific implementation. So I will go through FreeBSD, NetBSD and musl to port code to Newlib.
I will also be implementing missing functions such as **confstr()** from **unistd.h** and which can be ported from FreeBSD or musl as well.It will be added to RTEMS libbsd.
I will also add fmtmsg.h both can potentially be ported from FreeBSD. fmtmsg.h has as single-function fmtmsg() that is not present in Newlib and need to be added.

July 3 – July 31 (Phase 2)

After the first phase, I plan to implement File Descriptor functions such as fstat(), fcmod() and other missing functions whose list can be found here: List of missing RTEMS File descriptors.
Since RTEMS has its own File Descriptor implementation. the functions have to be written from scratch.
Also,I plan to implement ftw.h it contains functions like ftw() and nftw(). They can be ported from various potential sources like FreeBSD, NetBSD and musl. its implementation also

contains a lot of dependencies among headers, which I would try to decrease as much as possible.
I will also ensure to enable the search.h functionality in Newlib (newlib/libc/search). I will review the search.h capabilities against POSIX for completeness and compliance.

July 31 – August 31 (Final Phase)

In this phase, I will implement methods missing from math.h which are not present in Newlib. Firstly I will add the function prototype of missing to math.h then they can be ported to newlib from various sources. For the list of missing functions and there sources details can be seen in the document: Math.h missing functions.
I also plan to implement wordexp.h it can't be completely supported by RTEMS but if we remove a few test cases then it might work. For that, it either needs to be implemented from scratch or be ported with major tweaks.
Newlib also lacks ipc.h implementation it can be ported from FreeBSD or musl to Newlib.

Future Improvements

- Enhance support for POSIX mutex types
- Support for sigaction() sa_flags Values
- mq_open lacks the support of the "mode" argument
- sem_open lacks the support of the "mode" argument

Continued Involvement

I have a keen interest in kernel and OS as well as like to study various computer architectures and embedded systems.
I would definitely like to continue to contribute and learn from the RTEMS community. I believe I can do this by keeping track with the ongoing mailing lists and submitting patches as well.
I would also be delighted to get involved and learning and developing hardware proximity code and device drivers.
Eventually, I would also be interested in being on the mentoring side of GSOC Projects mentoring future students, Once I have learned enough.

Conflict of Interests or Commitment

I might have final exams in the month of May but the dates cannot be confirmed due to the recent outbreak of COVID-19 virus. Other than that I don't have any engagement during the project period.

Major Challenges foreseen

- Understanding the FreeBSD and NetBSD and musl code
- Understanding the header dependencies among header files
- Time Management
- Categorization of feasible functions and priority

References

- Linux man pages

- POSIX Wikipedia
- https://pubs.opengroup.org/onlinepubs/9699919799/
- https://devel.rtems.org/wiki/GSoC/GettingStarted
- RTEMS Documentation
- https://lists.rtems.org/pipermail/devel/

Relevant Background Experience

- I made a project to manage a criminal database written completely in C. Also I made a project to calculate cab fares using data structures and algorithms in C.

- RTEMS contributions:
  → https://lists.rtems.org/pipermail/devel/2020-March/058727.html : being reviewed
  → https://lists.rtems.org/pipermail/devel/2020-March/058755.html

Personal

I am Eshan Dhawan, I am a Second-year student pursuing Bachelors in Computer Science Engineering from JIIT (Jaypee Institute of Information Technology), Noida, INDIA.
I was introduced to RTEMS by one of my seniors who had previously contributed to the organization. I have some theoretical knowledge related to operating systems and embedded systems. Also have worked with the raspberry pi and Arduino nano for various college projects.

Experience

Free Software Experience/Contributions (optional):

- I am an active member of JIIT OPEN-SOURCE DEVELOPERS CIRCLE (JODC) the open-source society of our college. I am also one of the few volunteers of the society https://github.com/JIITODC/JODC-TEAM/tree/master/2019-2020

- I also have been a part of the open-source community and have contributed to other projects as well

  - CCEXTRACTOR Contributions (core software), contributed in c.
    Solved issues in code
    https://github.com/CCExtractor/ccextractor/pulls?q=is%3Apr+author%3Aeshandhawan51

  - Sample-Platform (Test platform for CCEXTRACTOR),contributed in shell script
    Solved an issue in installation which led to an infinite loop, as well as helped enhance the login.
    https://github.com/CCExtractor/sample-platform/pulls?q=is%3Apr+is%3Aclosed+author%3Aeshandhawan51

Language Skill Set

- C/C++(4 years, advanced)
- JAVA (5 years , advanced)
- LINUX SHELL SCRIPTING (6 months, beginners)
- PYTHON (2 years , intermediate)

Reference Links and Web URLs (optional):

- GitHub : https://github.com/eshandhawan51
- LinkedIn : https://www.linkedin.com/in/eshan-dhawan-798359192/
- Freenode IRC : eshandhawan51 @ #rtems