FreeBSD GSOC 2024 Proposal

Getz Mikalsen

2024-04-01

Contents

Note: This proposal is also available in the following formats:
ASCII Text: `https://dflund.se/~getz/gsocproposal.txt`
HTML: `https://dflund.se/~getz/gsocproposal.html`

1 General Information

Name Getz Mikalsen
Email getz@sdf.org
Phone +46737301192
IRC getz at libera.chat
Matrix @getz:envs.net
Timezone UTC+1

I can dedicate a significant amount of my time during the summer. With no other major obligations, I plan to work around 40 hours per week on the proposed tasks and objectives.

As for the timeline, I will align my schedule with the suggested Google Summer of Code program dates. The project will aim to commence on the official coding start date.

I am committed to maintaining open communication with my mentor and the FreeBSD community throughout the project's duration. This will ensure that I stay on track, address any challenges promptly, and incorporate feedback effectively. Regular progress updates and discussions will be a priority to facilitate a smooth and successful collaboration as suggested by GSOC.

1.1 Biography

For the past four years, I have been an active FreeBSD user, running it on my personal laptop. Currently, I serve as a System Administrator at our University's Computer Society, Datorföreningen vid LU/LTH. In this role, I manage a diverse range of computer systems, including legacy machines like VAX, Sun, and SGI, as well as modern servers running FreeBSD, Debian, and OpenBSD. My responsibilities encompass system upgrades, security patch installation, troubleshooting, and backup and recovery procedures.

As an Embedded Electronics Engineering student pursuing a Master's Programme at Lund University (LTH) in Sweden [1], I have gained extensive knowledge in processor design, computer architecture, and I am currently implementing the RISC-V base ISA as a soft core on an FPGA. Furthermore, I have an interest in reverse engineering and actively participate at crackmes.one (previously crackmes.de). Reading and writing assembly code is fun and at times tedious, -O2 is a mess.

Mentor Robert Clausecker <fuz@FreeBSD.org> volunteers to mentor the project.

We have discussed the project on #freebsd.soc on libera.chat

2 Porting amd64 libc SIMD enhancements to arm64 (aarch64)

2.1 Project Description

The proposed project aims to enhance the performance of the FreeBSD libc library by expanding its performance enhancements to the aarch64 archi-

tecture. This will be done by introducing NEON and groundwork for SVE support for optimized string manipulation functions.

This will involve addressing the lack of the pmovmskb instruction in NEON through clever techniques. Furthermore, the project will lay the groundwork for future SVE support by introducing an `"aarch64_archlevel.h"` header file to handle different architectural levels. The scope of the project will be limited to porting to NEON due to time constraints and lack of SVE capable hardware.

I currently have access to NEON capable hardware in the form of an M1 Macbook and two Raspberry Pi 5.

2.1.1 libc Porting and Optimization

The initial phase of the project (before midterm evaluation) is suggested to concentrate on porting and optimizing smaller libc functions from the *src/lib/libc/amd64/string* directory, such as strlen. These functions will serve as a foundation for optimizing other string manipulation routines, including strchrnul, strcat, and related functions.

2.2 Technical Approach

As an initial step, I have begun porting the strlen function to the aarch64 architecture with NEON optimizations. The amd64 implementation leverages the PCMPEQB, PMOVMSKB, and BSF instructions to efficiently scan the string bytes. For aarch64, I discovered the equivalent optimization technique using the SHRN, FMOV, RBIT, CLZ, and bit shifting instructions, as used in the ARM "Optimized Routines" codebase [2]. Important to keep in mind is to avoid the costly moves back and forth to GPRs.

2.3 Significance for FreeBSD

Enhancing the libc library's support for aarch64 architectures is crucial for ensuring FreeBSD's continued relevance and performance on modern ARM-based systems. As the adoption of ARM processors in server and embedded environments continues to grow, optimizing the libc library for these architectures will enable FreeBSD to deliver optimal performance and efficiency on a broader range of hardware platforms.

I am committed to becoming a long-term FreeBSD contributor, actively maintaining and further improving the optimized code and hope that this GSOC project will be a good starting point.

2.4 Applicant Background

As an Embedded Electronics Engineering student pursuing a Master's Programme at Lund University (LTH) in Sweden, I am passionate about performance optimization and low-power computing. Single Instruction Multiple Data (SIMD) technologies like NEON and SVE offer significant potential for improving the efficiency of critical software components, such as the libc library, on ARM-based systems. By leveraging these instruction set extensions, we can unlock substantial performance gains while minimizing power consumption, which is crucial in the embedded and mobile computing domains.

I'm currently involved in a research project with a group at LTH creating an in-memory computing platform using memristor arrays. This project ends before the summer.

I previously maintained of a branch of GerdaOS [3], for the JioPhone1, a featurephone released in India. Until I broke the device while debugging. :D I have completed relevant courses in computer architecture, digital design, operating systems and effective C programming. I am confident in my ability to contribute effectively to the proposed project.

I have a strong background in programming languages such as C, C++, Rust, and shell scripting (z{sh}, sed, awk). Additionally, I have hands-on experience with patching FreeBSD and am familiar with the workflow, "git pull && make buildworld && make buildkernel".

Im familiar with x86-64 and MIPS assembly which positions me well to grasp ARM/NEON concepts quickly.

3 Deliverables

Introduce a new ARCHLEVEL flag for the aarch64 architecture, allowing selection between base, NEON, and SVE instruction set extensions. This will involve creating an `aarch64_archlevel.h` header file to manage the different levels of architectural support.

Update the `simd(7)` manual page to document the new SIMD-accelerated string functions for aarch64, including details on the ARCHLEVEL flag and its usage.

And port the functions with already existing SIMD implementations for amd64 to aarch64.

All development and contributions will be hosted in a publicly accessible fork of the FreeBSD source code repository (freebsd-src git repository). The primary deliverable functions for this project are the following:

```
Functions to be Ported
----------------------

HEADER    FUNCTION  NOTES
string.h  stpcpy    String copy functions
          stpncpy
          strcat
          strncat
          strcpy
          strncpy
          strlcpy
          strlcat
          strchrnul
          strrchr
          strcspn
          strspn
          strpbrk
          strsep    String tokenisation functions
          strcmp    String comparison functions
          strncmp
          memcpy    Memory copy functions
          memccpy
          memset    Memory initialisation functions
          memchr    Memory search functions
          memrchr
          memcmp    Memory comparison function
          strlen    String length
          timingsafe_bcmp Constant time functions
          timingsafe_memcmp

Additional functions that will benefit from these enhancements:

strings.h index           *legacy, through strchr
          rindex         *legacy, through strrchr
          bcmp            *legacy, through memcmp
          bcopy           *legacy, through memmove
          bzero           *legacy, through memset
          explicit_bzero  through memset
          memset_s        through memset
          mempcpy         through memmove
```

```
memcpy          through memmove
strdup          through memcpy, strlen
strtok          through strtok_r
```

3.1 Testing

Performance benchmarking will be carried out the same way fuz did when he wrote libc SIMD enhancements for amd64 using his strfperf [4] utility. Functionality tests are located in *src/lib/libc/tests/string*, If further tests are needed they will be written.

For example `/src/lib/libc/tests/string/memcmp_test.c` has support for custom memcmp function. Helping development tremendously.

4 Project Schedule

The submitter is based in Copenhagen, Denmark.

I intend to continue studying ARM intrinsics ahead of the start date and provide a proof of concept SIMD port for strlen(3) for the first week. Then gather feedback and investigate the need for additional unit tests for the ported functions.

This work will build from the same algorithmic ideas developed in the SIMD enhancened functions provided by fuz last year [5].

4.1 Timeline

350 hours are allocated for the project as per the GSOC guidelines for a large project.

I request the coding start date to be moved up from May 27 to May 31st due to collisions with an University exam on the 30th

May 1-26 Community bonding

May 31 Coding start

July 8 Midterm goals reached

July 12 Midterm eval deadline

August 26 - September 2 Final GSoC contributor evaluations

[1] `https://www.lunduniversity.lu.se/lubas/i-uoh-lu-TAEEE`

[2] `https://github.com/ARM-software/optimized-routines`

[3] `https://archive.luxferre.top/gerdaos`

[4] `https://github.com/clausecker/strperf`

[5] `http://fuz.su/~fuz/freebsd/2023-04-05_libc-proposal.txt`