Mini-libc

Objectives

- Learn about the structure and functionalities provided by the standard C library
- Accommodate with the syscall interface in Linux
- Gain a better understanding of strings and memory management functions
- Learn how the standard C library provides support for low-level input/output operations

Statement

Build a **minimalistic** standard C library implementation for Linux systems (named **mini-libc**), that can be used as a replacement for the **system libc** (glibc in Linux). The goal is to have a minimally functional libc with features such as string management, basic memory support and POSIX file I/O.

The implementation of mini-libc will be **freestanding**, i.e. it will not use any outside library calls. It will be implemented on top of the system call interface provided by Linux on an `x86_64` architecture. Any function you require, that is typically part of libc, you will have to implement. You can reuse functions that you implement in other parts of the mini-libc.

In case you are using a macOS device with ARM64 / Aarch64, you will have to install an `x86_64` virtual machine.

Support Code

The support code consists of three directories:

- `src/` is the skeleton mini-libc implementation. You will have to implement missing parts marked as `TODO` items.
- `samples/` stores use-cases and tests of mini-libc.
- `tests/` are tests used to validate (and grade) the assignment.

System call invocation is done via the `syscall()` function defined in `src/syscall.c`. That itself makes a call to the architecture-specific call in `src/internal/arch/x86_64/syscall_arch.h`; hence the dependency on the `x86_64` architecture.

API and Implementation Tasks

The application programming interface (API) of the C standard library is declared in a number of header files. Each header file contains one or more function declarations, data type definitions and macros. For your minimal implementation, the following header files are of interest:

- `<string.h>`: defines string-handling functions

  For this assignment, you will have to implement the following functions: `strcpy()`, `strcat()`, `strlen()`, `strncpy()`, `strncat()`, `strcmp()`, `strncmp()`, `strstr()`, `strrstr()`, `memcpy()`, `memset()`, `memmove()`, `memcmp()`.

- `<stdio.h>`: defines printing and I/O functions

  For this assignment, you will have to implement `puts()`.

- `<unistd.h>`, `<sys/fcntl.h>` and `<sys/stat.h>`: define I/O primitives

  For this assignment, you will have to implement the following functions: `open()`, `close()`, `lseek()`, `stat()`, `fstat()`, `truncate()`, `ftruncate()`.

  You will also have to implement the `nanosleep()` and `sleep()` functions.

- `<stdlib.h>` and `<sys/mman.h>`: define memory allocation functions

  For this assignment, you will have to implement the following functions: `malloc()`, `free()`, `calloc()`, `realloc()`, `reallocarray()`, `mmap()`, `mremap()`, `munmap()`.

  For managing memory areas, a basic list structure is provided in `include/internal/mm/mem_list.h` and `mm/mem_list.c`.

- `<errno.h>` and `errno.c`: declare and define the integer variable `errno`, which is set by system calls and some library functions in the event of an error to indicate what went wrong.

Some tests do not build. This is intentional. You will have to add the missing features to make those tests compile, that is

- the `time.h` header
- the declaration and the implementation of `puts()`
- the declaration and the implementation of `nanosleep()` and `sleep()`
- the update of the libc `Makefile` to build the source code files implementing `puts()`, `nanosleep()` and `sleep()`

> ❗ ❗ **Pay attention** to which functions have to modify the `errno` variable.

Building mini-libc

To build mini-libc, run `make` in the `src/` directory:

```
student@so:~/.../content/assignments/mini-libc$ cd src/

student@so:~/.../assignments/mini-libc/src$ make
```

To build samples, enter the `samples` directory and run `make`:

```
student@so:~/.../content/assignments/mini-libc$ cd samples/

student@so:~/.../assignments/mini-libc/samples$ make
```

Testing and Grading

The testing is automated. Tests are located in the `tests/` directory.

```
student@so:~/.../assignments/mini-libc/tests$ ls -F
Makefile        graded_test.inc.sh  run_all_tests.sh*   test_so_file_create.sh*
grade.sh*       io/                 test_fstat.sh*      test_so_file_delete.sh*
graded_test.c   memory/             test_ftruncate.sh*  test_lseek.sh*
graded_test.h   process/            test_ls.c           test_malloc.sh*
```

To test and grade your assignment solution, enter the `tests/` directory and run `grade.sh`. Note that this requires linters being available. The easiest is to use a Docker-based setup with everything installed, as shown in the section "Running the Linters". When using `grade.sh` you will get grades for checking correctness (maximum `90` points) and for coding style (maxim `10` points). A successful run will provide you an output ending with:

```
### GRADE


Checker:                                                    90/ 90
Style:                                                      10/ 10
Total:                                                     100/100


### STYLE SUMMARY
```

Running the Checker

To run only the checker, use the `make check` command in the `tests/` directory:

```
student@so:~/.../assignments/mini-libc/tests$ make check
make[1]: Entering directory '.../
rm -f *~
[...]
test_mmap_perm_ok                       ......................... failed ...  0
test_mmap_perm_notok                    ......................... failed ...  0
test_mmap_perm_none                     ......................... failed ...  0

Total:                                                                      0/100
```

Some files will fail to build, it's expected. This is because there are missing files or missing functions that cause build errors. You'll need to add those files and implement those functions for the build error to disappear.

Obviously, most tests will fail, as there is no implementation. Some tests don't fail because the missing implementation equates to the bad behavior being tested not happening.

Each test is worth a number of points. The total number of points is `900`. The maximum grade is obtained by dividing the number of points to `10`, for a maximum grade of `90`.

A successful run will show the output:

```
student@so:~/.../assignments/mini-libc/tests$ make check
[...]
test_strcpy                             ......................... passed ...  9
test_strcpy_append                      ......................... passed ...  9
test_strncpy                            ......................... passed ...  9
test_strncpy_cut                        ......................... passed ...  9
test_strcat                             ......................... passed ...  9
test_strcat_from_zero                   ......................... passed ...  9
test_strcat_multiple                    ......................... passed ...  9
test_strncat                            ......................... passed ...  9
test_strncat_cut                        ......................... passed ...  9
test_strcmp_equal                       ......................... passed ...  9
test_strcmp_same_size_less              ......................... passed ...  9
test_strcmp_same_size_greater           ......................... passed ...  9
test_strcmp_diff_size_less              ......................... passed ...  9
test_strcmp_diff_size_greater           ......................... passed ...  9
test_strncmp_equal_size_equal           ......................... passed ...  9
test_strncmp_diff_contents_equal        ......................... passed ...  9
test_strncmp_diff_size_equal            ......................... passed ...  9
test_strchr_exists                      ......................... passed ... 11
test_strchr_exists_twice                ......................... passed ...  9
test_strchr_not_exists                  ......................... passed ...  1
test_strrchr_exists                     ......................... passed ... 11
test_strrchr_exists_twice               ......................... passed ...  9
test_strrchr_not_exists                 ......................... passed ...  1
test_strstr_exists                      ......................... passed ... 11
test_strstr_exists_twice                ......................... passed ... 13
test_strstr_not_exists                  ......................... passed ...  1
test_strrstr_exists                     ......................... passed ... 11
test_strrstr_exists_twice               ......................... passed ...  9
test_strrstr_not_exists                 ......................... passed ...  1
test_memcpy                             ......................... passed ... 11
test_memcpy_part                        ......................... passed ...  9
test_memcmp_equal_size_equal            ......................... passed ...  9
test_memcmp_diff_contents_equal         ......................... passed ...  9
test_memcmp_diff_size_equal             ......................... passed ...  9
test_memset                             ......................... passed ...  9
test_memset_part                        ......................... passed ...  9
test_memmove_apart                      ......................... passed ...  9
test_memmove_src_before_dst             ......................... passed ...  9
test_memmove_src_after_dst              ......................... passed ...  9
test_open_non_existent_file             ......................... passed ...  8
test_open_invalid_access_mode           ......................... passed ...  8
test_open_file_as_directory             ......................... passed ...  8
test_open_directory_for_writing         ......................... passed ...  8
test_open_force_invalid_creation        ......................... passed ...  8
test_open_close_existent_file           ......................... passed ...  8
test_open_close_create_file             ......................... passed ...  8
test_open_read_write_only_mode          ......................... passed ...  8
test_open_write_read_only_mode          ......................... passed ...  8
test_lseek_invalid_fd                   ......................... passed ...  8
test_lseek_invalid_whence               ......................... passed ...  8
test_lseek_invalid_offset               ......................... passed ...  8
test_lseek_set                          ......................... passed ...  8
test_lseek_cur                          ......................... passed ...  8
test_lseek_end                          ......................... passed ...  8
test_lseek_combined                     ......................... passed ...  8
test_truncate_read_only_file            ......................... passed ...  8
test_truncate_invalid_size              ......................... passed ...  8
test_truncate_directory                 ......................... passed ...  8
test_truncate_non_existent_file         ......................... passed ...  8
test_truncate_file                      ......................... passed ...  8
test_ftruncate_read_only_file           ......................... passed ...  8
test_ftruncate_invalid_size             ......................... passed ...  8
test_ftruncate_directory                ......................... passed ...  8
test_ftruncate_bad_fd                   ......................... passed ...  8
test_ftruncate_file                     ......................... passed ...  8
test_stat_non_existent_file             ......................... passed ...  8
test_stat_regular_file                  ......................... passed ...  8
test_fstat_bad_fd                       ......................... passed ...  8
test_fstat_regular_file                 ......................... passed ...  8
test_puts                               ......................... passed ... 15
test_open_close_create_file             ......................... passed ... 10
test_open_close_read_byte               ......................... passed ... 10
test_ftruncate                          ......................... passed ... 10
test_truncate                           ......................... passed ... 10
test_fstat                              ......................... passed ... 10
test_stat                               ......................... passed ... 10
test_sleep                              ......................... passed ... 20
test_nanosleep                          ......................... passed ... 20
test_mmap                               ......................... passed ...  8
test_mmap_bad_fd                        ......................... passed ...  8
test_mmap_bad_flags                     ......................... passed ...  8
test_mremap                             ......................... passed ...  8
test_malloc                             ......................... passed ...  8
test_malloc_two                         ......................... passed ...  8
test_malloc_access                      ......................... passed ...  8
test_malloc_memset                      ......................... passed ...  8
test_malloc_memcpy                      ......................... passed ...  8
test_calloc                             ......................... passed ...  8
test_realloc                            ......................... passed ...  8
test_realloc_access                     ......................... passed ...  8
test_realloc_memset                     ......................... passed ...  8
test_realloc_array                      ......................... passed ...  8
test_malloc                             ......................... passed ... 10
test_multiple_malloc                    ......................... passed ... 10
test_malloc_free                        ......................... passed ... 10
test_multiple_malloc_free               ......................... passed ... 10
test_malloc_free_sequence               ......................... passed ... 10
test_malloc_perm_ok                     ......................... passed ... 10
test_malloc_perm_notok                  ......................... passed ... 10
test_mmap                               ......................... passed ... 10
test_mmap_munmap                        ......................... passed ... 10
test_mmap_perm_ok                       ......................... passed ... 10
test_mmap_perm_notok                    ......................... passed ... 10
test_mmap_perm_none                     ......................... passed ... 10

Total:                                                                     90/100
```

Running the Linters

To run the linters, use the `make linter` command in the `tests/` directory. Note that the linters have to be installed on your system: `checkpatch.pl`, `cpplint`, `shellcheck` with certain configuration options. It's easiest to run them in a Docker-based setup with everything configured:

```
student@so:~/.../assignments/mini-libc/tests$ make lint
(...)
cd ... && checkpatch.pl -f checker/*.sh tests/*.sh
(...)
cd ... && cpplint --recursive src/ tests/ checker/
(...)
cd ... && shellcheck checker/*.sh tests/*.sh
```

Behind the Scenes

For a fine grained approach, build tests and ignore errors (due to missing source code and header files) by using:

```
student@so:~/.../assignments/mini-libc/tests$ make -i
```

Then run the tests, either individually via executable files and scripts:

```
student@so:~/.../assignments/mini-libc/tests$ ./test_lseek.sh
test_lseek                              ......................... passed ...  10

student@so:~/.../assignments/mini-libc/tests$ ./test_memory
test_mmap                               ......................... passed ...  8
test_mmap_bad_fd                        ......................... passed ...  8
[...]
```

Or run them all via the `run_all_tests.sh` script:

```
student@so:~/.../assignments/mini-libc/tests$ ./run_all_tests.sh
test_strcpy                             ......................... passed ...  9
test_strcpy_append                      ......................... passed ...  9
test_strncpy                            ......................... passed ...  9
[...]
```

Resources

- GNU libc manual
- musl implementation of the standard C library for Linux-based systems
- Syscall interface in Linux - Linux man pages online
- glibc implementation of the standard C library for Linux-based systems