Section 1 / Example of calling some common C runtime functions

This chapter gives an example of calling the most common C I/O functions from assembly language.

There are, by the way, two broad types of functions within the C runtime. Some are implemented largely in the C runtime itself. Others that exist in the C runtime act as wrappers for functions implemented within the OS itself. These are called "system calls".

For the purposes of calling functions in the C runtime, there is no practical difference between these two types. Note however, there are ways of calling system calls directly using the `svc` instruction. We will cover this way of making system calls as well. See here.

Low level file operations

This example program makes use of `open()`, `close()`, `read()`, `write()` and `lseek()`. These are implemented in the C runtime as wrappers for system calls.

The program will

- create a file,
- write a small amount of text to it,
- rewind (seek) back to the beginning of the file,
- read back and print the contents of the file and then
- close the file.

A lot of error checking is also implemented (frankly speaking: until we got bored writing the example).

Doing all of these ballooned the example program to about 200 lines. As such we won't explain the code line by line but, in compensation, the code is liberally commented.

Here is just a bit:

```
/*  off_t lseek(int fd, off_t offset, int whence);
*/
seek_zero:
        stp     x29, x30, [sp, -16]!
        mov     w0, fd          // file descriptor
        mov     x1, xzr         // beginning of file
        mov     w2, wzr         // SEEK_SET - absolute offset
        bl      lseek
        ldp     x29, x30, [sp], 16
```

```
ret
```

Calling this function rewinds the read / write “head” to position 0.