System Programming in C

A system call can be defined as a request to the operating system to do something on behalf of the program. During the execution of a system call, the mode is change from user mode to kernel mode (or system mode) to allow the execution of the system call.

The kernel: the core of the operating system program in fact has control over everything. All OS software is trusted and executed without any further verification.

Unlike processes in user mode, which can be replaced by another process at any time, a process in kernel mode cannot be arbitrarily replaced by another process but only can be suspended by interrupt or exception.

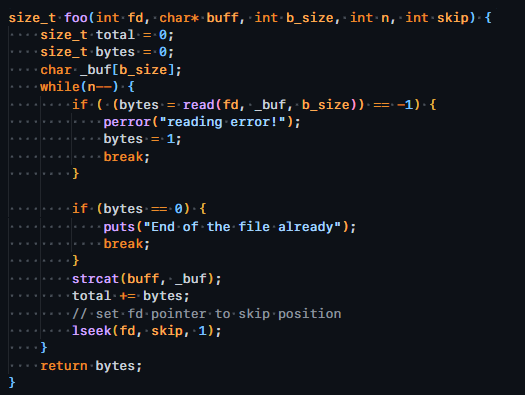
Unix system calls are primarily used to manage the file system or control processes or to provide communication between multiple processes.

A subset of the system calls include:

* creat(), open(), close() – managing I/O channels
* read(), write() – handling input and output operations
* lseek() – for random access of files
* link(), unlink() – aliasing and removing files
* stat() – getting file status
* access(), chmod(), chown() – for access control
* exec(), fork(), wait(), exit() – for process control
* getuid() – for process ownership
* getpid() – for process id
* signal(), kill(), alarm() – for process control
* chdir() – for changing working directory
* mmap(), shmget(), mprotect(), mlock() – manip-ulate low level memory attributes
* time(), gettimer(), settimer(), settimeofday(), alarm() – time management functions
* pip() – for creating inter-process communication

Exercise:

1. foo function



2.

References:

1. [Lecture 24: System Programming in C](https://www.cs.cmu.edu/~guna/15-123S11/Lectures/Lecture24.pdf)
2. source code: <https://github.com/PhanNgocTrieu/Advanced>