Lab 1: Reading Material

Pointers and Memory in C

Lab 2 focuses on pointers in C. Pointers are a concept missing in some higher-level languages; in C, pointers are powerful but dangerous. The chapter Pointers and arrays of the wikibook C Programming provides an overview of concepts related to pointers in C. You should recall the lecture slides from last week regarding memory and pointers in C. In addition, Allocating memory dynamically is done by system calls, most conveniently through standard library functions such as malloc(). Read the Linux man page malloc(3) to understand how to use it.

Using GNU Debugger to locate fatal errors

GNU Debugger (gdb) is a free tool for debugging programs in C and other languages in Linux. gdb has a rich set of commands, and learning most of them would take significant time. However, for basic use of GDB, only a small subset of its capabilities is sufficient. The following GDB - intro helps to begin using the GNU Debugger and is sufficient for the debugging assignments of the System Programming Laboratory. For further reading, consider the GNU Debugger Manual.

Input and Output Format Conversions

The C Standard Library includes powerful facilities for input and output of primitive types (integer and floating point numbers, characters, strings, pointers). The facilities are described in man pages for fgets(3), printf(3) and scanf(3). Please read the man pages and use the appropriate functions in the lab assignments. Note that although we point out "scanf", you should avoid it, and use its "string" version "sscanf" instead (after using fgets to read an input line into a char array).

Ending a program

The C Standard Library includes the exit function, that ends the program without reaching the end of the main function. Please read its man page exit.(3)

String to number conversion

The C Standard Library includes functions such as atoi(3), that converts a string representation of a number to an int numeric representation. Please read the man page of atoi(3)