Optional Lab
Exercises in C programming on Unix

Magnus Grandin
magnus.grandin@it.uu.se

January 21, 2015

These exercises are completely optional and are intended to help you get started with C-programming, or as a refresher if you have seen it before. Look at the lecture notes from the introductory lecture on C-programming for examples and guidelines on how to solve the exercises.

Part 1: Hello world and more

Do these excercises in order and extend your program as you go. Compile your program with `gcc -o hello_world hello_world.c`

(a) Write a program that prints "Hello World!" on the screen.

(b) Put the print command in a function that is called from the main function.

(c) Pass an `int` and a `double` as function parameters and print them too.

(d) Write a function that recursively generates the Fibonacci series up to a given number.

(e) Create an array a where $a_i = i$ and the number of elements is determined by the number passed to the function. Copy a to another array b and set $b_i = i^2$. Print all elements of the array.

(f) Compute the square root of a number (using the function *sqrt()* in *math.h*). Print the result. **Note:** `-lm` must be added to the compile command in order to use the math functions.

Part 2: Matrix multiplication

Let two dense matrices be given, $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$, $A, B \in \mathbb{R}^{n \times n}$, i.e. $i, j = 1, 2, \ldots, n$. Let C be the product of the two matrices, $C = AB$. The elements of $C = \{c_{i,j}\}$ are given by

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}, \quad i, j = 1, 2, \ldots, n$$

Write a program that computes the matrix-matrix product above and prints the result on the screen. Use a struct to represent a matrix type and try to separate the program into at least one function other than the main function. Allocate the matrices dynamically.

Hint: You will have to write 3 nested for-loops.

Question: In which order should we iterate over the elements in the matrices for best efficiency? Think about how data is laid out in memory and how the matrices are stored.

Part 3: Arrays, pointers and memory

The purpose of this exercise is to practice proper use of pointers and memory allocation. Your task is to debug a program that we have *intentionally broken*. Look at the code, and try to compile and run the program. Pay special attention to the way memory is used, how pointers are allocated, and so on. This excercise also gives you an idea of how to structure a program into functions and files, and introduces you to makefiles.

1. Download the file **C-lab-files.tar.gz**.
2. Unpack the archive with the command **tar -xvzf C-lab-files.tar.gz**.
3. The program is now in the folder called "Exercise3".
4. Compile the program by running **make** and try to run it.
5. Fix the code so as to remove the compiler warnings and runtime errors.

Hint: You can compile with the -g flag and use the debuggers gdb or ddd to help you debug. Ask your instructor for further help on using the debuggers.