Prof. Dr. Burkhard Stiller, University of Zurich, Binzmühlestrasse 14, CH-8050 Zürich
Telephone: +41 44 635 6710, Fax: +41 44 635 6809, stiller@ifi.uzh.ch
Bruno Rodrigues, Telephone: +41 44 635 4381, rodrigues@ifi.uzh.ch
Sina Rafati, Telephone: +41 44 635 6702, rafati@ifi.uzh.ch

Exercises in SSDS

Exercise 1

Hand out date: Feb 19, 2020
Submission until and discussion: Feb 20, 2020 08:15

Submission Guidelines

- Submitted code must compile without errors.
- Zip your code using the name pattern: ssds_ex**+exercise+number_initials**. For instance, the file of Exercise 01 for Bruno Rodrigue would be named: sssds_ex01_BR.zip
- Submit the exercise to the reseach assistants e-mail addresses.

Introduction to C

C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope, and recursion. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, language constructs that map efficiently to machine instructions, and to require minimal run-time support. Despite its low-level capabilities, it was designed to encourage cross-platform programming.

It was initially used for system development work (UNIX), particularly the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of its use are:

- Operating systems
- Language compilers
- Assemblers
- Text editors
- Data bases

1. Theoretical Exercises

Concepts and Definitions

1.1) A program is a set of instructions that acts as an intermediary between the user and the computer hardware. It can be an user application, an Operating System (OS), Kernel, etc. In this regard, (a) what are the differences between an application, OS, Kernel and Bootstrap and (b) how these programs relate to each other?

- Application:
- OS:
- Kernel:
- Bootstrap:

1.2) Relate the computer structures below with their corresponding options.

Computer System Structure:
[1] Hardware
[2] Operating System
[3] Application Programs
[4] Users

Corresponding option:
[ ] Windows
[ ] PDF Reader
[ ] Kernel
[ ] ROM, EPROM, CPU, RAM
[ ] Mouse
[ ] Printer
[ ] Linux
[ ] Chrome Browser
[ ] Bob
[ ] Alice Computer
[ ] Word
[ ] Virtual Box
[ ] Bob's Virtual Machine

1.3) A computer structure may have one or more applications concurrently requesting hardware resources. To efficiently make use of hardware, the OS use mechanisms such as **interrupts (1)**, **traps (2)** and **system calls (3)**. Regarding their concepts and functionality, mark the respective number 1, 2 or 3 for each of the concepts below with their correspondend meaning in the table's first column.

No.	Concept
	Triggered by user applications and handled in the Kernel
	Triggered by hardware components (e.g., keyboard, mouse, I/O ports) and handled in the Kernel causing a processor to temporally transfer control to another program, or function
	Software-generated signaling caused either by an error or a user request
	Examples: fork(), open(filename.txt), printf("hello world")
	Also known as exceptions being typically triggered by problems such as division by zero and invalid memory access
	Programming interfaces to the services provided by the OS

1.4) Today's computing systems use a multi-processing architecture in order to increase efficiency and reliability. However, these systems can be implemented in two types, namely **asymmetric (1)** and **symmetric (2)**, depending on the goal of an application/use case. Regarding these multi-processing types, tick their correspondent characteristic in the table's first column.

No.	Chacarteristic
	CPUs are treated differently with respect to I/O access or Kernel code execution
	Typicaly an OS is used and this is a single instance that runs on all the CPUs

No.	Chacarteristic
	Each core has its own copy of a kernel, which could be different or identical to one the other core is executing
	CPUs have full access to all I/O devices and Kernel code execution.
	Mostly implemented in application-specific hardware, such as sensors
	Use multiple CPUs

1.5) Hierachical protection domains are methods implemented by modern computing systems to prevent malicious behaviors (intentional or not) to damage OS functionality or user data. Within the hierachical protection, a mode bit is used to identify the process priority. Explain the differences and priviledges between processes running the **User and Kernel** modes.

- User mode:
- Kernel mode:

2. Practical Exercises

2.1) Looping structures and if-else statements

Based on the code of "hello.c", create an array of 100 positions statically allocated. Using a looping structure (for, while, do-while), insert in the even positions of the array the position in the array and string "Even", and in the odd positions the string "Odd". In the end, print the array using another looping structure. Save your code as "static_loop.c". The output of your code should be:

1 Odd
2 Even
...
99 Odd
100 Even

2.2) Dynamic Memory Management

The C programming language provides several functions for memory allocation and management. These functions can be found in the <stdlib.h> header file (*cf.* Table 1).

Table 1: C Memory Management Methods

Function	Description	Syntax
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space	ptr = (cast-type*) malloc(byte-size)
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory	ptr = (cast-type*)calloc(n, element-size);
free()	deallocate the previously allocated space	free(ptr);
realloc()	Change the size of previously allocated space	ptr = realloc(ptr, newsize);

Modify the code of Exercise 3 to use dynamic memory allocation. At the end of the code, release the allocated memory using free().

2.3) Recursion in C

The C language supports recursion, i.e., a function to call itself. However, while using recursion, it is required to be careful to define an exit condition from the function. Otherwise, it will go into an infinite loop. One of the main examples of recursion in C is the Fibonacci sequence. A Fibonacci sequence is defined as:

$$fn = fn\text{-}1 + fn\text{-}2$$

In which: $f0 = 0$, $f1 = 1$, $f2 = 1$, $f3 = 2$, $f4 = 3$, $f5 = 5$, $f6 = 8$, ...

In this exercise you will write a C program to print the Fibonacci series until a fN number using recursion.