Lecturers:	Prof. Dr. Florina M. Ciorba	`florina.ciorba@unibas.ch`
	Dr. Osman Simsek	`osman.simsek@unibas.ch`
Assistants:	Thomas Jakobsche	`thomas.jakobsche@unibas.ch`
Tutors:	Reto Krummenacher	`reto.krummenacher@unibas.ch`
	Agni Ramadani	`agni.ramadani@unibas.ch`
	Tom Rodewald	`tom.rodewald@unibas.ch`

Exercise 1: OS-User Interface (10 points)

Given: March 8, 2024
Deadline: April 2, 2024, 10:00

Objectives

- Familiarize with Linux system calls through the command line
- Use system calls to implement basic Linux utilities
- Familiarize with the C language
- Parse input arguments of a program written in C

Tasks

- Task 1: Get familiar with C (3 points)
- Task 2: System calls and error handling (3 points)
- Task 3: Program linking and loading (3 points)
- Task 4: Compiling vs. Linking (1 point)

Instructions

- You can solve this exercises in teams of two.
- Submit the solution of each task with detailed comments that clarify your solution.
- Show your solution and upload it to https://adam.unibas.ch.
- Provide all deliverables as an archive file.
- In total, at least 65% of exercise points have to be obtained (with a min of 30% of each exercise).

Task 1: Get familiar with C (3 points)

Given the source code T1.c, implement a function that accepts the following three arguments: a pointer to an integer array, an integer representing the length of the given input array, and an integer value representing a target. The function must return -1 if the target integer does not exist in the input array. Otherwise, it must return the existence frequency of the target within the input array. The signature of the function must look as follows.

Listing 1: function signature

```c
int get_frequency(const int * input, int length, int target);
```

Once you implement get_frequency, compile T1.c with a GNU compiler and execute the program. Besides your modified T1.c (1.5 points), provide the compilation command (0.5 point) and the final output (1 point).

Task 2: System calls and error handling (3 points)

A user wants to implement software that accepts a path to a specific folder and a target file name. This software must
1) create the folder if it does not exist. Otherwise, it must alert the user.
2) create the link to the newly created folder.

- implement Task 2.1 and Task 2.2 using Linux bash commands. (1.5 points)
  Hint: start by using bash commands directly on your terminal, once you are sure, wrap them in a file, which we call bash script. For more details, see T2.sh.
- implement Task 2.1 and Task 2.2 in C. One should consider sys/stat.h and errno.h. Both contain all system calls you need for Task 2.1 and Task 2.2. (1.5 points)

Expected usage of your bash script

```
$ sh T2.sh /etc/temp/ test.txt
$ mkdir:  cannot create directory '/etc/temp':  Permission denied.
$ sh T2.sh temp/ test.txt
$ sh T2.sh temp/ test.txt
$ mkdir:  cannot create directory 'temp/':  File exists.
```

Expected usage of your C program

```
$ ./T2.o /etc/temp/ test.txt
$ cannot create directory '/etc/temp':  Permission denied.
$ ./T2.o temp/ test.txt
$ ./T2.o temp/ test.txt
$ cannot create directory 'temp/':  File exists.
```

Task 3: Program linking and loading (3 points)

A given makefile (Makefile) has three different entries; each of the three entries compiles T3.c. Execute the compilation results (t3_case1, t3_case2, and t3_case3), report the output in each case, and explain why you get different outcomes although T3.c did not change.

Task 4: Compiling vs. Linking (1 point)

Explain briefly what is the difference between compiling and linking for this exercise.