University of Zurich^UZH

Department of Informatics IfI
University of Zurich
Binzmühlestrasse 14
CH—8050 Zürich
Switzerland
URL: https://www.ifi.uzh.ch

Prof. Dr. Burkhard Stiller
Bruno Rodrigues
Rafael Ribeiro
Communication Systems Group
CSG
Phone: +41 44 635 46 81
E-mail: [lastname]@ifi.uzh.ch
URL: https://www.csg.uzh.ch

3121- Systems Software and Distributed Systems (SSDS)

E1 - Exercises Introduction and Organization

1 Introduction to Operating Systems

An Operating System (OS) is a software collection that manages computer hardware resources and provides common services for high-level application software to interact with the hardware [1]. It is a vital component in a computer system responsible for basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on disk, and controlling peripheral devices such as printers.

Processors run binary machine code, and assembly is the language that represents Machine Code. OS requirements became too arduous to implement purely in Assembly; therefore, using a high-level programming language can make the OS development task much easier.

The most popular language for writing OSs is C, which is used to perform the basic functionalities (Kernel) of Linux, Microsft Windows, and macOS. C was initially used for system development work. It was widely adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language.

2 Baseline Environment to Practical Exercises

We will use Ubuntu as OS configured in a VM since it is relatively more straightforward than other Operating Systems (OS). However, as indicated below, you can also compile C code on other major OSes such as Windows and macOS. This simple guide (not meant to be a definitive/self-included one) has the objective to prepare the baseline environment, perform the SSDS exercises, and introduce the C programming language's basic concepts.

2.1 Windows

The less preferred one since it may restrict some operations involving the creation of processes and the use of semaphores (it is possible to use Windows libraries, but I recommend using a VM to avoid messing with a "daily" use OS). Simple exercises may run fine with Dev-Cpp (`https://sourceforge.net/projects/dev-cpp/`) or any other editor.

- **Alternative 1**: download and install Dev-Cpp. Exercises requiring to create processes (*e.g.,* fork()) and semaphores may require further libraries [1].
- **Alternative 2**: use a VM with (suggested) Ubuntu:
    1. Download VirtualBox `https://www.virtualbox.org/wiki/Downloads`
    2. Download the Ubuntu image and deploy it on VirtualBox. `https://www.ubuntu.com/download/desktop`
    3. GCC is installed by default. Read/use the tutorial on how to compile C programs. `http://pages.cs.wisc.edu/~beechung/ref/gcc-intro.html`

2.2 Linux

Most distributions already have "gcc" built-in. Check whether it is installed by typing in your terminal:

```
$ which gcc
```

If gcc is not installed, install it by typing in the terminal:

```
$ sudo apt install gcc
```

To compile a C code, you can type in the terminal:

```
$ gcc myCode.c
```

If the program were compiled without error or warning, you would have a new file in the directory called a.out. To run the file, go to the same directory and type in the terminal:

```
$ ./a.out
```

Furthermore, if you want to specify the output, it is possible by doing the following:

```
$ gcc myCode.c -o hello
$ ./myCode
$ "Hello Code"
```

[1] `https://docs.microsoft.com/en-us/windows/win32/sync/using-semaphore-objects`

2.3 macOS

- **Alternative 1**: use the macOS terminal and gcc `https://www.cs.auckland.ac.nz/~paul/C/Mac/`
- **Alternative 2**: Installation of VirtualBox on macOS:
  1. Visit VirtualBox.org and go to Downloads.
  2. Select OS X hosts, which starts the download of `https://download.virtualbox.org/virtualbox/6.1.14/VirtualBox-6.1.14-140239-OSX.dmg`
  3. Run the installation (*i.e.,* double-click on the Virtualbox.pkg) and click through the installation
  4. You may need to allow the application access through the security settings

3 Practical Exercises

3.1 The first program, hello.c

The first has the objective to get you familiar with the development environment, the C syntax, and the GCC compiler. Create a code printing a simple “Hello World” message, and wait for a keyboard entry to terminate its execution. Save your code as “hello.c”.

3.2 Hello World via System Call

Note that the important is not the hello world itself, but how the operating system handles the high-level calls (”printf”), how the output is made, among other examples. Thus, **strace** conveniently shows these system calls using C syntax. Thus, output the result of the following command:

```
$ strace ./a.out
```

Action:

- Which line is the command that effectively prints the ”Hello World” message?
- What is the difference between the command and ”printf”?
- Using the POSIX standard, rewrite your code to print a ”Hello World” directly calling the system call (*i.e.,* without using ”printf,” *cf.* `https://www.man7.org/linux/man-pages/man2/write.2.html`).
- Compare the output of both ”straces.” Which is one is the most compact?

3.3 Hello World using inline Assembler

As a curiosity, it is also possible to write a Hello World at a low-level using the gcc inline assembler [2]. The asm keyword allows you to embed assembler instructions within C code (*cf.* . `https://gcc.gnu.org/onlinedocs/gcc/Basic-Asm.html#Basic-Asm`). You do not need to hand-in this example.

```c
int main(void) {
  register int    syscall_no  asm("rax") = 1;
  register int    arg1        asm("rdi") = 1;
  register char*  arg2        asm("rsi") = "Hello World!\n";
  register int    arg3        asm("rdx") = 14;
  asm("syscall");
  return 0;
}
```

4 Theoretical Exercises

1. A program is a set of instructions that acts as an intermediary between the user and the computer hardware. It can be a 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: ______________________________
   - Operating System: ______________________________
   - Kernel: ______________________________
   - Bootstrap: ______________________________

2. Relate the computer structures below with their corresponding options.
   Computer System Structure:

   (a) Hardware
   (b) Operating System
   (c) Application Programs
   (d) Users

   Corresponding Alternative:

   [__] Windows
   [__] PDF Reader
   [__] Kernel
   [__] ROM, EPROM, CPU, RAM
   [__] Mouse
   [__] Printer

[__] Linux

[__] Chrome Browser

[__] Bob

[__] Alice Computer

[__] Word

[__] Virtual Box

[__] Bob's Virtual Machine

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 correspondent 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 transfer control to another program temporally, 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

4. Today's computing systems use a multi-processing architecture 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.	Concept
	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
	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

5. Hierarchical protection domains are methods implemented by modern computing systems to prevent malicious behaviors (intentional or not) from damaging OS functionality or user data. Within the hierarchical protection, a mode bit is used to identify the process priority. Explain the differences and privileges between processes running the User and Kernel modes.

   - User mode: ____________________
   - Kernel mode: ____________________

6. (Book's question): Give two reasons why caches are useful. What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

7. (Book's question): In a multi-programming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems:

   (a) What are two such problems? ____________________

   (b) Can we ensure the same degree of security in a time-shared machine as in a dedicated machine? Explain your answer.

8. (Book's question): Distinguish between the client-server and peer-to-peer models of distributed systems. ____________________

9. (Book's question): Which of the following instructions should be privileged? Mark the alternatives.

   (a) Set value of timer. ____________________
   (b) Read the clock. ____________________

(c) Clear memory. ____________________

(d) Issue a trap instruction. ____________________

(e) Turn off interrupts. ____________________

(f) Modify entries in device-status table. ____________________

(g) Switch from user to kernel mode.____________________

(h) Access I/O device.____________________

10. (Book's question): Describe the differences between symmetric and asymmetric multi-processing. What are the three advantages and one disadvantage of multi-processor systems?

____

____

____

5 Submission Guidelines

- Submitted code must compile without error.
- Zip your code using the name pattern: ssds_ex+exercise+number_initials (*e.g.,* ssds_ex01_BR.zip)
- Submit the exercise to the research assistants e-mail addresses.

References

[1] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts*. Wiley Publishing, 8th edition, 2012.

[2] James Fisher. How to make a system call in C. https://jameshfisher.com/2018/02/19/how-to-syscall-in-c/, March 2020. Accessed: 2020-02-02.