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)

E2 - Operating Systems Structures

1 Theoretical Exercises

1. The Linux directory structure is like a tree containing all the methods and data structures that an OS uses. It is accessible via the command "/" or via user interface (open a file manager on your Linux box and select Computer in the sidebar to show your system folders). This structure is the starting point for the file system hierarchy, being responsible for many services (e.g., I/O operations, resource allocation, file systems, and others). Complete the table below providing the description of these common top level directories.

   - /bin: is short for binaries and includes most executables for the users
   - /boot: this holds the files used for booting the os
   - /dev: is short for devices as linux sees everything either as a folder or device all devices can be accesed by writting to these files
   - /etc: This directory contains the config files for the OS
   - /lib: Contains the libraries which the kernel needs to boot
   - /sys: includes information about the system itself such as installed hardware
   - /proc: this folder contains runtime information, i.e. it contains control information for the kernel
   - /tmp: this folder contains temporary files such as temporary data from running prog.
   - /mnt: Here are all drives "mounted" this means if you have different drives or partitions you can access them through this folder
   - /home: Each user is assigned a user folder and these are contained in here

2. Regarding the **/proc** directory on Linux. Tick which file contains information about direct memory access channels.

    [__] /proc/cpuinfo

    [__] /proc/dma

    [__] /proc/ioports

    [__] /proc/help

    [__] /proc/modules

3. The Linux directory is organized into separate folders containing the top level directories. Based on the directories below:

    - /bin - host configuration and startup files
    - /boot - Static kernel and files needed to load the kernel
    - /dev - device inputs for terminals, disks, modems, etc.
    - /etc - essential commands required for minimal system operation

    Tick the option that contains those that have the content exchanged.

    [__] /bin and /etc

    [__] /dev and /etc

    [__] /bin and /boot

    [__] /dev and /boot

    [__] All options are correct.

4. Among the various functions of the kernel and its architectures, mark as True (T) or False (F) regarding the following sentences:

    [__] The kernel is the basis on which other parts of the operating system, device drives, and programs run

    [__] A micro kernel is not modular and difficult to port into new architectures

    [__] An operating system is considered monolithic when your kernel runs as multiple programs.

    [__] Monolithic kernels moves as much as possible from the "kernel space" into "user" space

    [__] Keep control of the files on disk; initialize and run programs concurrently; and allocate memory.

    The BIOS, CMOS and EPROM of a computing system correspond, respectively, to:

    [__] Software, software and hardware.

[__] Software, hardware and hardware.

[__] Hardware, hardware and software.

[__] Software, hardware and software.

[__] Hardware, software and hardware.

5. Tick the option which is NOT function of the Kernel in an operating system.

    [__] Accounting for system usage

    [__] Support for local and distributed networks

    [__] File-system management

    [__] Provide command line interface

    [__] I/O management

6. The operating system is a program or set of programs whose function is to manage system resources, as well as provide an interface between the computer and the user. In this regard, mark the INCORRECT alternative:

    [__] Process management is an operating system task.

    [__] Memory management is an operating system task.

    [__] The file system is defined and manipulated by the operating system.

    [__] Reproduce text and images using a master form or template

    [__] Managing the input and output devices is the task of the operating system.

7. The tendency of operating systems is to make the kernel smaller and simpler. The idea of providing the services of process management, file management, memory management, etc. processes in the user space, outside the core, is called:

    [__] Macro kernel

    [__] Monolithic kernel

    [__] Nano kernel

    [__] Micro kernel

    [__] Hybrid kernel

8. The communication of an application with the input and output subsystem of an operating system is established by means of:

    [__] Shell

    [__] Device drivers

    [__] System calls

[__] Scripting

[__] Batch

9. In operating systems, for an operation to execute a privileged instruction, the processor implements the access mode mechanism. This mechanism is basically divided into two access modes known as:

   [__] System calls and protected mode

   [__] Protected mode and cluster

   [__] User mode and kernel mode

   [__] Cluster and kernel mode

   [__] Asynchronous mode and user mode

10. One of the most important components of an Operating System is the Kernel. Tick the alternative below corresponding to the major goal of this component.

    [__] The Kernel represents the lowest abstraction level within the operating system, being responsible for bridging user applications with the hardware based on its drivers.

    [__] Responsible for bootstrapping the operating system, loading the core available services such as the scheduler and the file manager.

    [__] The Kernel belongs to the set of Graphical User Interface tools, assisting users in the application management

    [__] None of the alternatives above

11. Differently from physical machines, which relies on the user and kernel access modes, virtual machines only relies on the user mode as they can only be executed as an user application. Tick true or false.

    [__] True

    [__] False

12. The Linux operating system has a Monolith Kernel structure and a single user architecture, which increases its security. Tick true or false.

    [__] True

    [__] False

13. Which of the following is not an operating system?

    [__] Windows

    [__] Linux

    [__] Oracle

    [__] DOS

14. BIOS is used?

    [__] By the operating system

    [__] By the compiler

    [__] By the interpreter

    [__] By applications

15. Consider the Table below: Fill the table according to the correct Operating System (**Windows or Unix**) and category to which the system call belongs:

System Call	**Operating System**	**Category**
CreateProcess()		
pipe()		
getpid()		
chmod()		
ReadConsole()		
ioctl()		
WaitForSingleObject()		
chown()		
SetConsoleMode()		
fork()		

C1 Process control: *e.g.,* end, abort, create, terminate, allocate and memory.

C2 File management: *e.g.,* create, open, close, delete, read file etc.

C3 Device management

C4 Information maintenance

C5 Communication

C6 Protection

16. (Book's question): What are the five major activities of an operating system with regard to file management?

17. (Book's question): What are the three major activities of an operating system with regard to memory management?

18. (Book's question): What are the three major activities of an operating system with regard to secondary-storage management?

19. (Book's question): A system call is the way that a computer program requests a service from the kernel of the operating system it is executed on. Often, additional information is required when a system call is executed in which the exact type and amount of information may vary according to the particular operating system and system call. Thus, Describe three general methods for passing parameters to the operating system.

20. (Book's question): What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

21. (Book's question): What are the two models of inter-process communication? What are the strengths and weaknesses of the two approaches?

22. (Book's question): List at least three of the major categories of system calls.

2 Practical Exercises

The practical exercises for Module 02 - Operating Systems Structures *are not mandatory*. To evaluate in practice the operation of system calls, the example shown in `https://linuxhint.com/linux_system_call_tutorial_c/` is useful. In the example, 64 MB of data are copied from one file to another using standard read/write methods.

```
fread(buffer, sizeof(char), BUFFER_SIZE, fIn);
fwrite(buffer, sizeof(char), BUFFER_SIZE, fOut);
```

In the other example, system calls are used and the sendfile() method realizes the transfer.

```
sendfile(fOut, fIn, 0, BUFFER_SIZE);
```

2.1 Testing the performance of your OS to transfer files

To build the examples below, you will need the development tools installed on your distribution. On Debian and Ubuntu, you can install with:

```
apt install build-essentials
```

Then compile with:

```
gcc test1.c -o test1 && gcc test2.c -o test2
```

To run both and test the performance, run:

```
time ./test1 && time ./test2
```

2.1.1 test1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/file.h>
#include <sys/random.h>

#define BUFFER_SIZE 67108864
#define BUFFER_1 "buffer1"
#define BUFFER_2 "buffer2"

int main() {

    FILE *fOut, *fIn;

```

```c
    printf("\nI/O test with traditional glibc functions.\n\n");

    // Grab a BUFFER_SIZE buffer.
    // The buffer will have random data in it but we don't care about that.
    printf("Allocating 64 MB buffer:                     ");
    char *buffer = (char *) malloc(BUFFER_SIZE);
    printf("DONE\n");

    // Write the buffer to fOut
    printf("Writing data to first buffer:                ");
    fOut = fopen(BUFFER_1, "wb");
    fwrite(buffer, sizeof(char), BUFFER_SIZE, fOut);
    fclose(fOut);
    printf("DONE\n");

    printf("Copying data from first file to second:      ");
    fIn = fopen(BUFFER_1, "rb");
    fOut = fopen(BUFFER_2, "wb");
    fread(buffer, sizeof(char), BUFFER_SIZE, fIn);
    fwrite(buffer, sizeof(char), BUFFER_SIZE, fOut);
    fclose(fIn);
    fclose(fOut);
    printf("DONE\n");

    printf("Freeing buffer:                              ");
    free(buffer);
    printf("DONE\n");

    printf("Deleting files:                              ");
    remove(BUFFER_1);
    remove(BUFFER_2);
    printf("DONE\n");

    return 0;

}
```

2.1.2 test2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/file.h>
#include <sys/sendfile.h>
#include <sys/random.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/fcntl.h>

#define BUFFER_SIZE 67108864

```

```
int main() {

    int fOut, fIn;

    printf("\nI/O test with sendfile() and related system calls.\n\n");

    // Grab a BUFFER_SIZE buffer.
    // The buffer will have random data in it but we don't care about that
   .
    printf("Allocating 64 MB buffer:                         ");
    char *buffer = (char *) malloc(BUFFER_SIZE);
    printf("DONE\n");


    // Write the buffer to fOut
    printf("Writing data to first buffer:                    ");
    fOut = open("buffer1", O_RDONLY);
    write(fOut, &buffer, BUFFER_SIZE);
    close(fOut);
    printf("DONE\n");

    printf("Copying data from first file to second:          ");
    fIn = open("buffer1", O_RDONLY);
    fOut = open("buffer2", O_RDONLY);
    sendfile(fOut, fIn, 0, BUFFER_SIZE);
    close(fIn);
    close(fOut);
    printf("DONE\n");

    printf("Freeing buffer:                                  ");
    free(buffer);
    printf("DONE\n");

    printf("Deleting files:                                  ");
    unlink("buffer1");
    unlink("buffer2");
    printf("DONE\n");

    return 0;

}
```

Paste here the performance output of your OS:

3 Submission Guidelines

- Include ”SSDS” in the e-mail’s title.
- Any submitted code must compile without error.
- Zip your files using the name pattern: ssds_ex+exercise+number_initials (*e.g.,* ssds_ex01_BR.zip)
- Submit the exercise to the research assistant responsible for the task e-mail addresses.