Table of Contents

1. Overview 2
  1.1. Introduction 2
  1.2. References 2
  1.3. Environment 2
  1.4. Folder structure 2
  1.5. Glossary 3
2. Knowledge 4
  2.1. File in Linux 4
3. Exercise 5
  3.1. Exercise 1 5
4. Solution notes 7
  4.1. C Compilation 7
5. Revision history 10

1. Overview

1.1. Introduction

This document provides information regarding the Linux Filesystem lesson.

1.2. References

Table 1-1: References

No.	Documents	Description
1	02_Linux_File_System.pdf	The lesson lecture

1.3. Environment

Table 1-2: Environment

Type	Component	Information
OS	Ubuntu	22.04.5 LTS
Kernel	Linux kernel	6.8.0-65-generic
GNU C Compiler	GCC	11.4.0
Build operation tool	GNU Make	4.3

1.4. Folder structure

Below is the folder tree.

```
<workdir>
    ├── doc                   --- All necessary documents
    │   └── Document.pdf
    ├── include               --- Location for headers
    │   ├── dl_error.h
    │   └── dl_filestat.h
    ├── Makefile              --- Makefile
    ├── sample                --- Sample application
    │   └── main.c
    └── src                   --- Location for library source files
        └── dl_filestat.c
```

Figure 1-1: Folder tree

1.5. Glossary

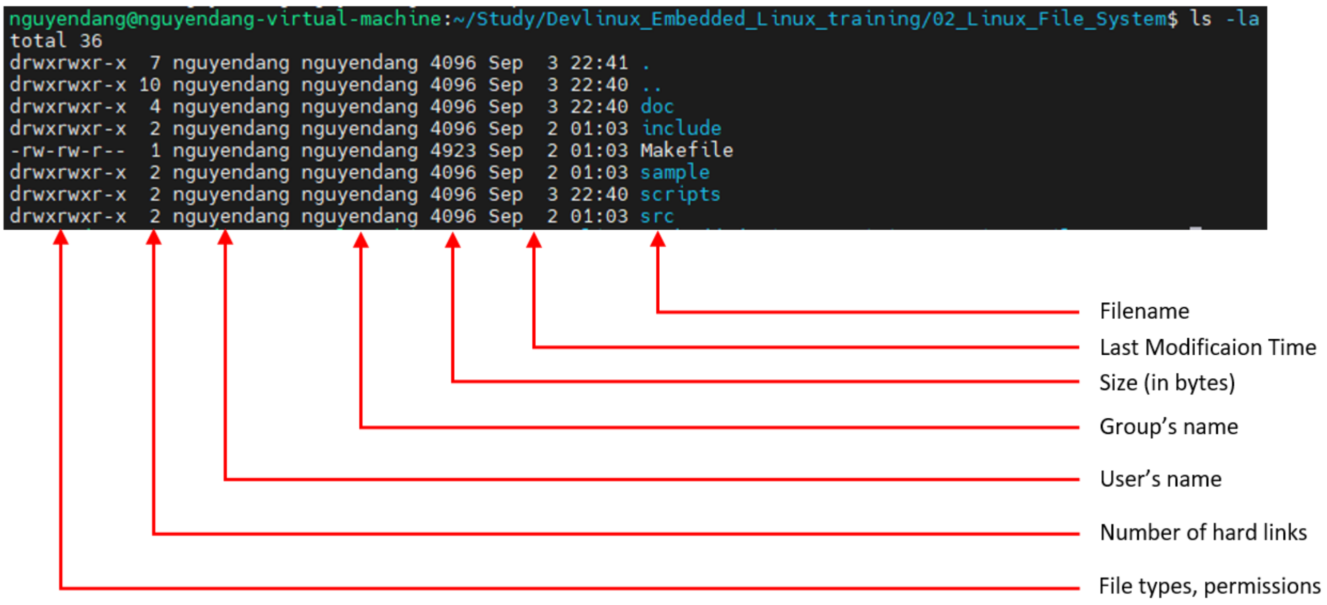
Table 1-3: Glossary

Abbr.	**Description**
ELF	Executable and Linkable Format

2. Knowledge

2.1. File in Linux

The Phisolophy of Linux: Everything is a file.



```
nguyendang@nguyendang-virtual-machine:~/Study/Devlinux_Embedded_Linux_training/02_Linux_File_System$ ls -la
total 36
drwxrwxr-x  7 nguyendang nguyendang 4096 Sep  3 22:41 .
drwxrwxr-x 10 nguyendang nguyendang 4096 Sep  3 22:40 ..
drwxrwxr-x  4 nguyendang nguyendang 4096 Sep  3 22:40 doc
drwxrwxr-x  2 nguyendang nguyendang 4096 Sep  2 01:03 include
-rw-rw-r--  1 nguyendang nguyendang 4923 Sep  2 01:03 Makefile
drwxrwxr-x  2 nguyendang nguyendang 4096 Sep  2 01:03 sample
drwxrwxr-x  2 nguyendang nguyendang 4096 Sep  3 22:40 scripts
drwxrwxr-x  2 nguyendang nguyendang 4096 Sep  2 01:03 src
```

Figure 2-1: File properties

Types of files in Linux:

- **Regular file**: Standard files such as text files or executable files.
- **Directory file**: A file that contains a list of other files (i.e., a folder).
- **Character device file**: A special file that represents character-based devices.
- **Block device file**: A special file that represents block-based devices.
- **Link file**: A file that represents (or points to) another file.
- **Socket file**: A file that represents a socket.
- **Pipe file**: A file that represents a pipe (used for inter-process communication).

3. Exercise

File Metadata Inspector

Every file and directory on a Linux filesystem not only contains data but also carries a large amount of metadata, stored in a structure called an inode. Metadata includes:

- File type
- Size
- Permissions
- Ownership information
- Important timestamps

System calls from the `stat()` family are the most accurate and efficient way for a C program to access this information without having to parse the output of shell commands such as `ls -l`.

In this exercise, you will build a small command-line tool named `filestat`, similar to the existing stat command in Linux, to read and display the key information of any given file.

3.1. Exercise 1

Write the program `filestat.c`:

1. The program must accept exactly one command-line argument: the path to a file or directory.
2. If no argument is provided, print usage instructions (for example: `Usage: ./filestat <file_path>`) and exit.
3. Use the system call `lstat()` to retrieve information about the object at the given path and store it in a `struct stat` variable.
4. The output must include the following:
    - **File Path**: The path entered by the user.
    - **File Type**: The type of the object. You must detect and display at least three main types:
        - **Regular File**
        - **Directory**
        - **Symbolic Link**

Hint: Use the macros `S_ISREG()`, `S_ISDIR()`, `S_ISLNK()` from `<sys/stat.h>` to check the `st_mode` field.

- **Size**: The file size, retrieved from the `st_size` field (displayed with the unit **bytes**).
- **Last Modified**: The last modification time of the file.
    - Retrieve the `time_t` value from the `st_mtime` field.

- Convert this timestamp into a human-readable date and time string.

Hint: Use the functions `ctime()` or `strftime()` from `<time.h>` to format the time.

4. Solution notes

This chapter provide notes and information about the solutions to Exercise.

4.1. C Compilation

The following table lists all compiler options used in the solutions.

Table 4-1: C complier options used

Purpose	**Option**	**Description**
General Warning and Error Control	-Wall	Enables a common set of important warnings about questionable code
	-Wextra	Enables extra warnings that are not included in -Wall
	-Werror	Treats all warnings as errors, stopping compilation if any warning appears
Pedantic Checks	-Wpedantic	Warns if your code uses non-standard GNU extensions that are not in ISO C
	-pedantic-errors	Like -Wpedantic but treats those warnings as errors
Type Conversion and Shadowing	-Wshadow	Warns if a local variable shadows (hides) another variable with the same name from an outer scope
	-Wconversion	Warns when implicit type conversions may change a value (e.g., float to int)
	-Wsign-conversion	Warns when a value changes sign due to conversion (e.g., unsigned to signed)
	-Wcast-function-type	Warns when casting between incompatible function pointer types (calling such a pointer is undefined behavior)
Precision and Formatting	-Wdouble-promotion	Warns when a float is promoted to a double implicitly
	-Wformat=2	Enables strict format string checks for functions like printf() and scanf()
	-Wfloat-equal	Warns on direct equality/inequality comparisons of floating-point values (fragile due to rounding)
	-Wformat-truncation=2	Warns when bounded printf-style functions (e.g., snprintf) may truncate output (level 2 = stricter)
	-Wformat-overflow=2	Warns when printf-style formatting may overflow the destination buffer (level 2 = stricter)

Purpose	**Option**	**Description**
Memory Safety	-Wnull-dereference	Warns when the compiler detects a dereference of a NULL pointer
	-Wcast-align	Warns if a pointer cast results in a stricter alignment requirement
	-Wcast-qual	Warns when casting away const or volatile qualifiers
	-Wcast-align=strict	Like -Wcast-align but warns whenever a cast increases the required alignment (strictest mode)
	-Wstringop-overflow=4	Warns when built-in string ops (e.g., strcpy, memcpy) may overflow the destination (level 4 = most strict)
	-Wstringop-truncation	Warns when string operations may silently truncate the result
	-Walloca	Warns about use of alloca() (stack allocation; easy to misuse and non-portable)
	-Walloc-zero	Warns on zero-size allocations (e.g., malloc(0)), which are implementation-defined and error-prone
Static Analysis	-fanalyzer	Runs GCC's static code analyzer to detect potential runtime bugs (e.g., NULL dereferences, memory leaks)
Optimization and Debugging	-Og	Optimizes for debugging: keeps code easy to debug while still optimizing slightly
	-g	Generates debug information for use with debuggers like gdb

Purpose	Option	Description
Code Safety and Correctness	-Wundef	Warns if an undefined macro is used in #if or #elif without being checked with #ifdef
	-Wstrict-prototypes	In C, warns if a function is declared without specifying argument types
	-Wmissing-prototypes	Warns if a global function is defined without a prior prototype
	-Wpointer-arith	Warns for suspicious pointer arithmetic, like arithmetic on void*
	-Wwrite-strings	Makes string literals have const type to prevent accidental modification
	-Wunreachable-code	Warns about code that will never be executed
	-Wunused	Warns about anything declared but never used
	-Wunused-parameter	Warns when a function parameter is unused
	-Wunused-but-set-variable	Warns when a variable is written to but its value is never read
	-Wlogical-op	Warns about suspicious logical operations (e.g., && vs &, always-true/false tests)
	-Wduplicated-cond	Warns when an if/else if chain repeats the same condition
	-Wduplicated-branches	Warns when different branches contain identical code
	-Wstrict-overflow=5	Warns when the compiler assumes signed overflow is undefined and optimizes based on that (level 5 = most strict)
	-Woverflow	Warns about compile-time constant arithmetic that overflows the destination type
	-Wredundant-decls	Warns when an entity is declared multiple times in the same scope
	-Wnested-externs	Warns on extern declarations placed inside functions (confusing linkage/style)
	-Wmissing-noreturn	Warns when a function that never returns should be marked _Noreturn
	-Wmissing-declarations	Warns when a global function is defined without a prior prototype in a header
	-Winline	Warns when a function marked/expected to inline is not inlined (e.g., too large/complex)

5. Revision history

Version	Date	Chapter	Content
0.01	Sep 3rd, 2025	All	Newly created