Files and Filesystems

How a computer permanently stores your data

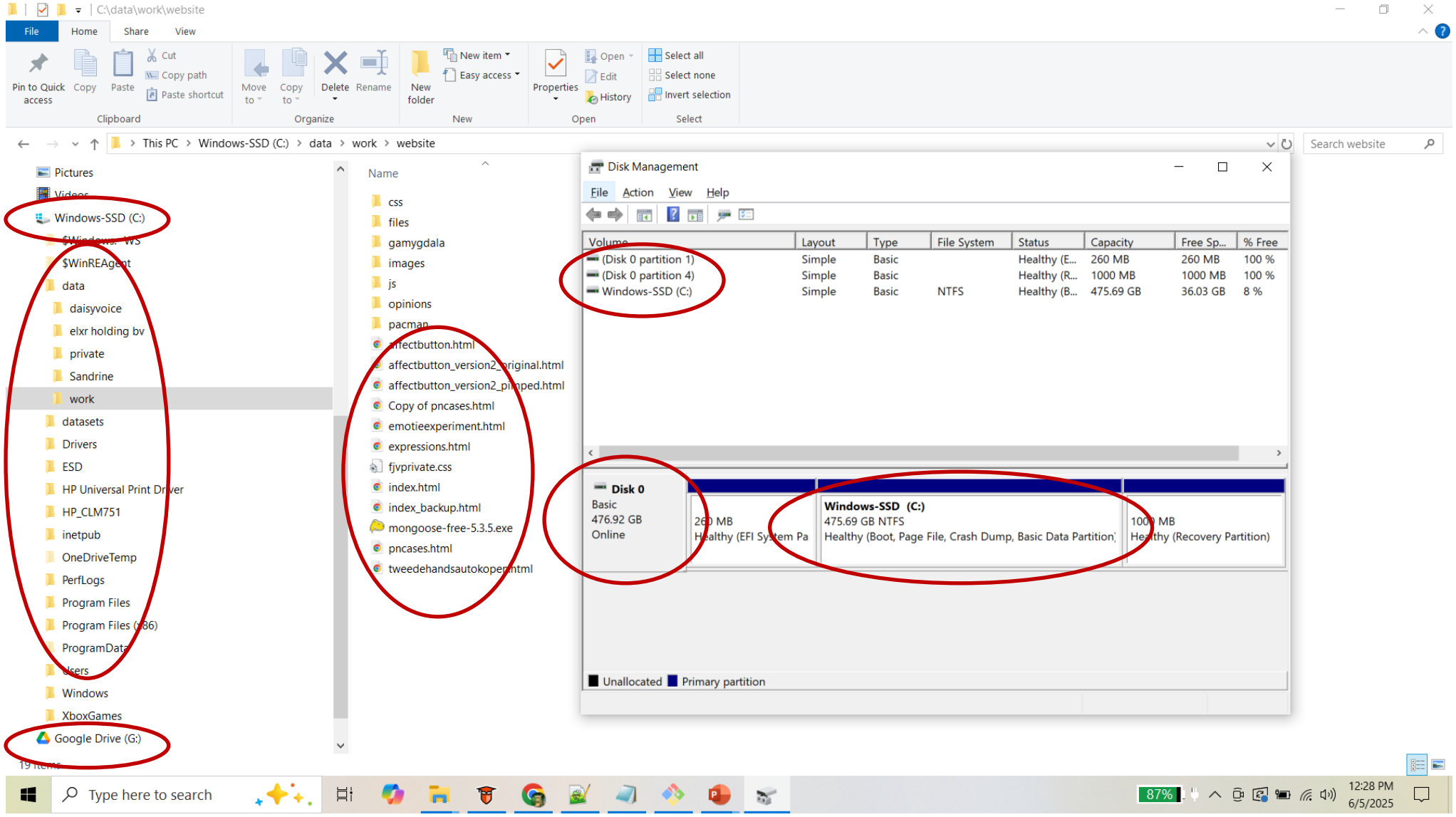
Learning goals

- Understand the file system structure: directories, files, links, file and folder permissions (read, write, execute).
- Understand the difference between file types (executables/data)
- Understand text/character encoding (ASCII, UTF-8, Unicode)
- Explain basic lossless file compression principle.

File systems

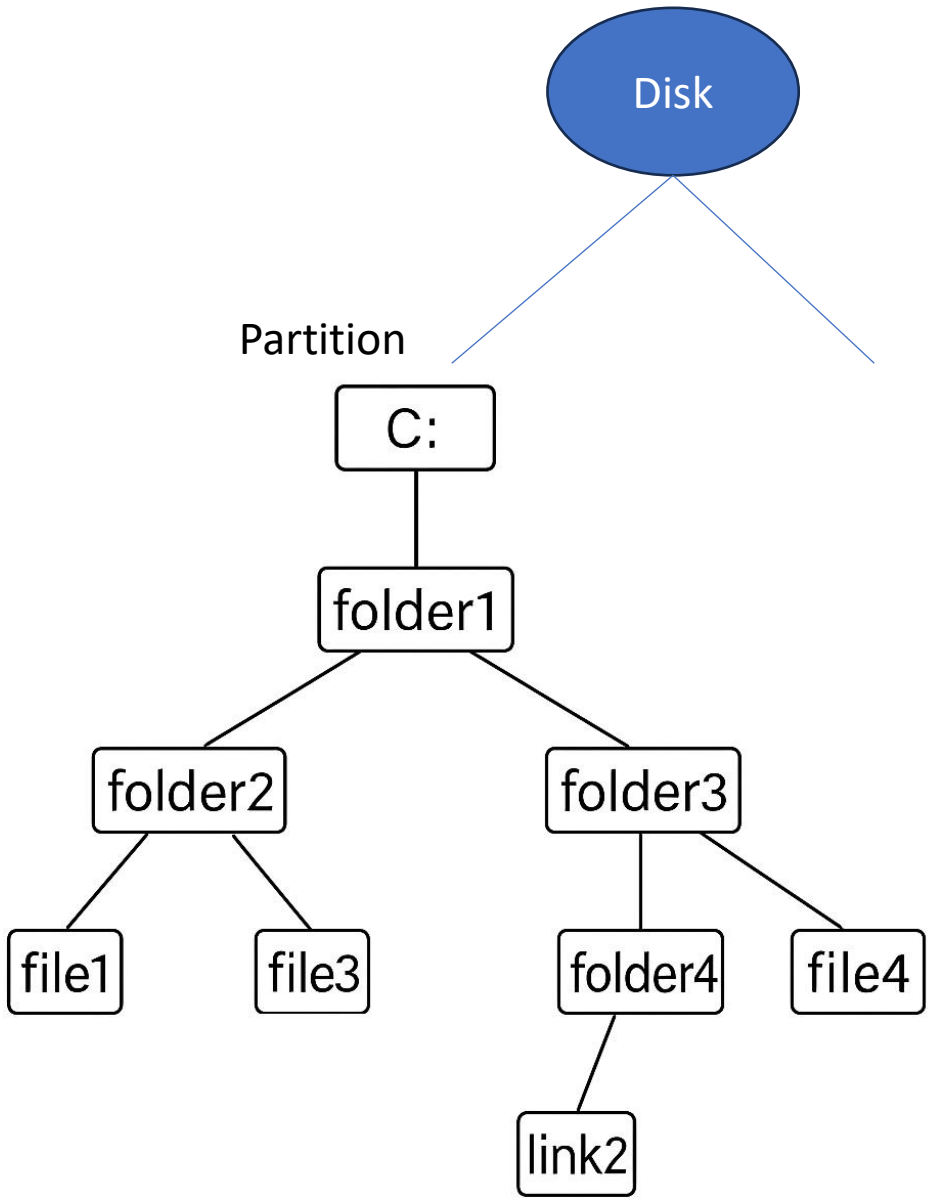
- To store data your computer uses *Hard disk (HDD) or Solid state disks (SSD)*
  - Non volatile storage
  - a physical device that retains data even when the power is off.
- A disk has *partitions*
  - a section of the physical disk that can be assigned a *Drive* letter (like C: or D:)
- A *drive* is a logical representation of storage space
  - It does need to be tied to a physical device
  - you can create drives in RAM memory or on USB sticks.
  - or create a drive that links to your Google cloud storage (Google Drive)
- A drive contains a *file system* stored at the beginning of the partition
  - Defines how files are stored and organized, consist of
  - *File System Metadata*
    - *File allocation information (where file data is on disk)*
    - *Directory structure (how files/folders are organized)*
    - *File attributes (timestamps, permissions, size, etc.)*
- Lets have a look! (with the application *Disk Management*)

File systems



Files and Directories

- A directory is a Tree
  - Think of a family tree
  - It contains subdirectories, files, and links.
  - Discrete Mathematics: the Tree!
- The directory structure
  - A root node: the drive (e.g. C:)
  - Inner nodes: folders/directories
  - Leaves: files and links/shortcuts
- Folders and files have permissions
  - Read: you may open the file to view it
  - Write: you may write to the file to change it
  - Execute: the file is an application

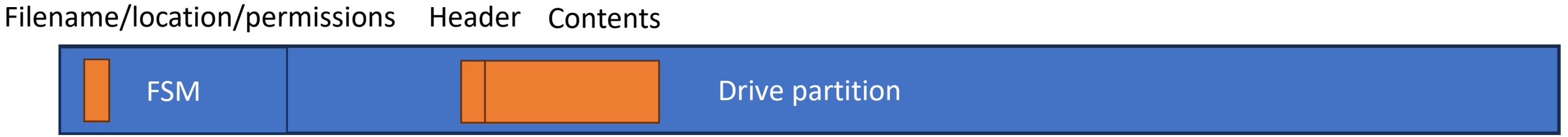


File types and encoding: how is the data stored

- A file always contains bytes; but we have conventions about how to interpret this data.
- Most data in a file is either:
  - Character encodings
    - Examples:
      - .txt – plain text
      - .html – web page source
      - .csv – comma-separated values
    - Data is encoded using a character encoding (e.g., UTF-8)
    - Can be opened and read in a text editor (of terminal)
  - CPU instructions (executables)
    - Examples: .exe in windows
    - Not readable in a text editor (shows symbols or "garbage")
    - Data are CPU instructions
    - Loaded by the OS and then "run" by CPU
    - Header with information about the program
  - Something else
    - .bmp, .gif – image formats
    - .mp3, .wav - music
    - Not readable in a text editor (shows symbols or "garbage")
    - Requires a specific program to interpret
    - Header with information about the data

File structure

- Filename, location in file system, location on disk
  - Stored in the *File System Metadata* (FSM, beginning of the drive partition)
- File type
  - Stored at start of the file (the header of the file)
- File contents
  - Stored after the header/metadata



File extensions

- Filenames have extensions
  - raw text: .txt
  - uncompressed image: bmp
  - word document: docx
  - application: .exe
- The file type (in windows) is not the extension (.xxx)
  - There is a difference, the extension is really just for Windows (or linux) to
    - Choose the application to open the file with (associated application)
    - Show an icon in the GUI (for the user to visually see the file type)
- The real file type is
  - defined by the file's internal structure and header
  - is stored as part of the file itself.
  - Windows (or Linux) does not read that header, but applications do.
  - You can test this: rename a .jpg to .txt and Windows won't complain until an app opens it.

Text files and character encoding

- Encoding = A system to represent characters as numbers (bytes).
- Computers store everything as numbers — encodings map text to bytes.
- Common Encodings:
- ASCII (American Standard Code for Information Interchange)
  - Early standard (1960s)
  - 128 characters: A–Z, a–z, digits, punctuation
  - Fits in 7 bits
  - "A" = 65, "a"=97, ENTER = 13, SPACE=32
- Unicode
  - Universal standard
  - Supports all languages, symbols, emojis
  - Each character has a unique code point (e.g., U+03A9 = Ω)
- UTF-8 (most common today)
  - A way to store Unicode using 1–4 bytes per character using "control" bytes
  - Backward compatible with ASCII
  - Efficient for English, flexible for global use

Text files and character encoding: UTF-8

Number of Bytes	First Byte Pattern	Follows with	Example
1 byte	0xxxxxxx	—	ASCII (e.g. A = 01000001)
2 bytes	110xxxxx	10xxxxxx	é (U+00E9) → 11000011 10101001
3 bytes	1110xxxx	10xxxxxx x2	€ (U+20AC) → 11100010 10000010 10101100
4 bytes	11110xxx	10xxxxxx x3	🙂 (U+1F642) → 11110000 10011111 10011001 10000010

File compression

- Files take up a lot of space on your disk or when sent over the network.
- To safe space, people invented *compression* algorithms
  - Typical compressed file types include **zip, rar, jpeg**
- There is lossless and lossy compression
  - Lossless: the compression does not throw away information to approximate the original data
  - Lossy: the compression may throw away information to approximate the original data, in order to safe MUCH more space.
- We only cover the basics of *lossless compression*

Lossy compression (image example)

- File size: 1598kB

Lossy compression (image example)

- File size: 638kB

Lossy compression (image example)

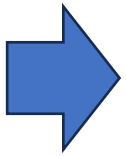
- File size: 283kB

Lossless compression

- Look for (large of frequently) repeating patterns
- Enumerate the patterns (make an index)
- Save the patterns and their sequence

The red ball bounces. The red ball bounces again. The red ball always bounces. The red ball never stops bouncing. The red ball is round. The red ball is fast. The red ball bounces and bounces and bounces.

Size=204 bytes



1: The red ball x 7
2: bounces x 6
3: again x 1
4: always x1
5: never stops x 1
6: bouncing x 1
7: is x2
8: round x 1
9: fast x1
0: and x2

The red ball
bounces
again
always
never stops
bouncing
is
round
fast
and
12123142156178179120202

Size= 86 bytes