jgromero@inf.uc3m.es

Universidad
Carlos III de Madrid
www.uc3m.es

Lesson 8
Advanced topics

Programming

Grade in Industrial Technology Engineering

Outline

1. External data storage
   a. Files
   b. Databases
2. Dynamic memory management
3. Programs used in Engineering

Outline

1. **External data storage**
   a. **Files**
   b. **Databases**
2. **Dynamic memory management**
3. **Programs used in Engineering**

Data storage on the hard disk

Permanent storage of program results

Data stored in the main memory of the computer is lost when the program finishes

Read data stored in a file, rather than asking the user to introduce it from the keyboard

Data in files with a proper format can be read and assigned to the variables and structures of the program

Dealing with data larger than the memory size

Data is stored in the hard disk, which has larger storage capabilities, and it is processed in parts

Outline

1. **External data storage**
   a. **Files**
   b. **Databases**
2. **Dynamic memory management**
3. **Programs used in Engineering**

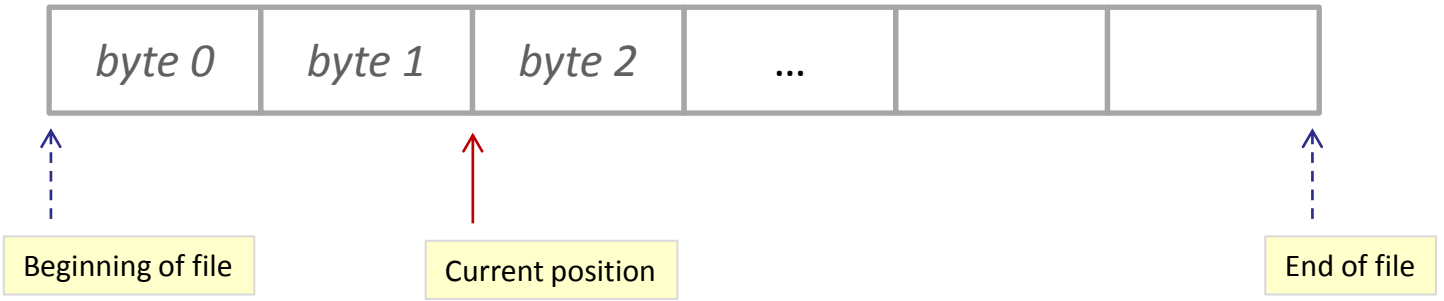
File

Data stored in a permanent read and write computer peripheral

Usually, files are stored on a hard disk

Sequence of bytes

Position in the file



C standard library provides functions for the management of *streams*

A stream is an abstract representation of any external source or destination for data –including disk files

Operations

> Open the file
> Read from file
> Write to file
> Close the file

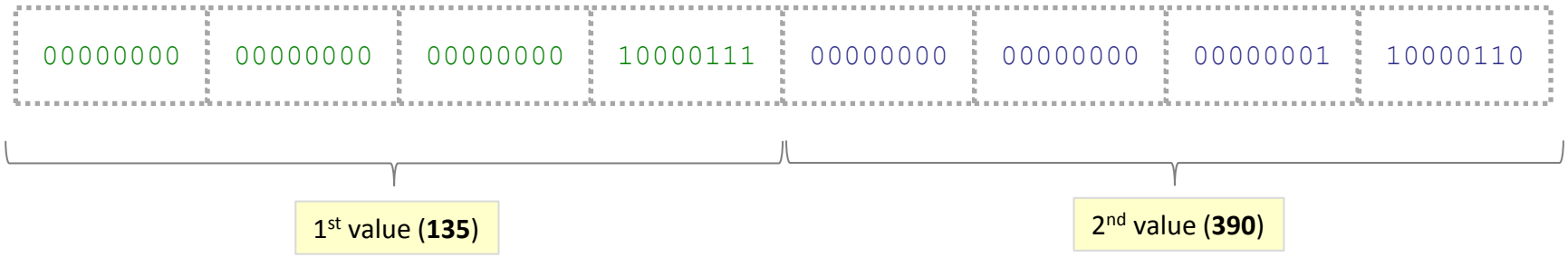
Binary files

Store a sequence of bits corresponding to the binary representation of data values

E.g.: an integer value is represented with 4 bytes

135 >> 00000000 00000000 00000000 10000111

390 >> 00000000 00000000 00000001 10000110



Text files

Store a sequence of bits corresponding to the textual representation of a data value

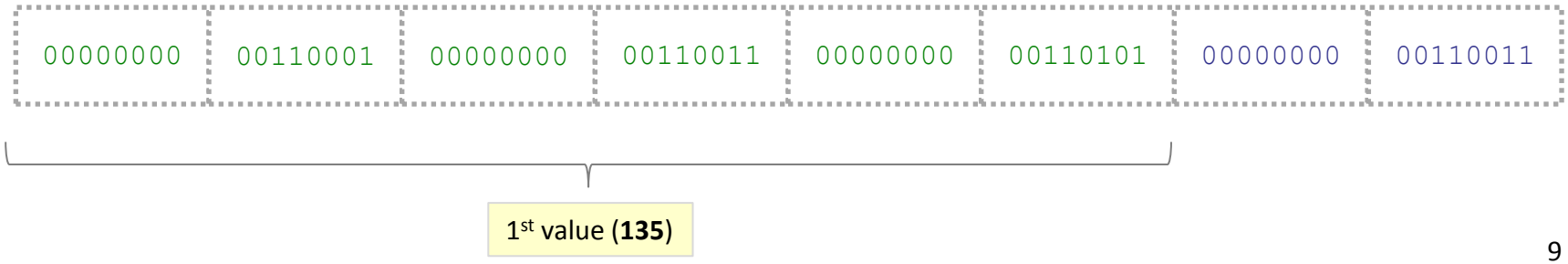
E.g.: an integer value is represented with as many characters as necessary (each character is 2 bytes)

135 >> '1' '3' '5' >> ASCII 49, ASCII 51, ASCII 53

00000000 00110001 00000000 00110011 00000000 00110101

390 >> '3' '9' '0' >> ASCII 51, ASCII 57, ASCII 48

00000000 00110011 00000000 00111001 00000000 00110000



Eventually, **all files contain just bytes**

The difference appears when a file is read, since it is necessary to correctly interpret the bytes

When it is a text file, each two bytes correspond to a character

When it is a binary file, it is necessary to know which kind of values were stored and in which order

By default, typical editors assume that files are text files

Text files can be created and edited with text editors. If a text file is open with a text editor (e.g., Notepad), its contents can be read by humans

Binary files can be created from a C program. If a binary file is open with a text editor (e.g., Notepad), its contents cannot be read by humans

Binary files

Are shorter than text files (with the same data)

It is not necessary to *translate* data from a text-based representation to the internal byte-based representation (this is a time-consuming task, in particular with float and double values)

Text files

Are easy to interpret

Can be edited with external tools

Procedure

> Declare file variable

`FILE *`

`<stdio.h>`

> Open file

`fopen`

Specify mode: binary/text, read/write

> Read data

text: `fscanf`, `fgetc`, `fgets`

binary: `fread`

> Print data

text: `fprintf`, `fputc`, `fputs`

binary: `fwrite`

> Close file

`fclose`

It is also possible to read and write data in a specific position of the file (not necessarily in sequential order)

`fseek`

ReadWrite.c

```c
#include <stdio.h>

#define N 10

int main(void) {

    FILE *fi, *fo;      // text files
    FILE *fib, *fob;    // binary files
    int i;
    int t, bin[1];      // values read from files

    /* Open files     */
    fi = fopen("input.txt",  "r");
    fo = fopen("output.txt", "w");
    fib= fopen("input.dat",  "rb");
    fob= fopen("output.dat", "wb");

    /* Process values (read and write from files) */
    for(i=0; i<10; i++) {
        // text file
        fscanf(fi, "%i", &t);
        t = t * i;
        fprintf(fo, "%i ", t);

        // binary file
        fread(bin, sizeof(int), 1, fib);
        bin[0] = bin[0] * i;
        fwrite(bin, sizeof(int), 1, fob);
    }

    /* Close files */
    fclose(fi);
    fclose(fib);
    fclose(fo);
    fclose(fob);
}
```

Outline

1. **External data storage**
   a. Files
   b. **Databases**
2. Dynamic memory management
3. Programs used in Engineering

From an abstract perspective, a **database is a data storage organized according to a logical structure**

Relational databases: data is stored in **tables**

- Record store database
  - **Album** table: album id, title, year, group id, song set id
  - **Group** table: group id, name, city, number of members
  - **Song** set table: song set id, song id
  - **Songs** table: song id, title, length minutes, length seconds

Database data is stored as files in the hard disk

- These files are not directly accessed
- A special program is used: the RDBMS (Relational Data Base Management System)
  - Oracle DBMS, MySQL, etc.

Non-relational databases: documents (MongoDB), graph-based (Neo4j), key-value (Cassandra), etc.

1. External data storage

Relational Databases

Album table

Album ID	Title	Year	Group ID	Song Set ID
1	The Suburbs	2011	1	1
2	Neon Bible	2004	1	2
3	Hombre Lobo	2011	2	3
4	Funeral	2004	1	4
5	High Violet	2011	3	5

Group table

Group ID	Name	City	# memb.
1	Arcade Fire	Montreal	8
2	Eels	California	1
3	The National	Cincinatti	5

Data is split in tables to avoid redundancies. Pieces of information are linked through **keys**

The RDBMS provides a query language to retrieve data from tables

SQL language (Structure Query Language)

> Retrieve the title and year of all albums by “Arcade Fire”

```
SELECT
Title, Year FROM Album, Group
WHERE
Album.Group_ID=Group.Group_ID AND
Name = "Arcade Fire"
```

Outline

1. External data storage
   a. Files
   b. Databases
2. **Dynamic memory management**
3. Programs used in Engineering

Dynamic memory management is a set of techniques (supported by C functions) to optimize the amount of memory required by a program

Program to store the record store catalogue

May include 30 or 30.000 albums

Easy solution

Declare an array of 30.000 albums

Use only the first 'N' elements of the array

30.000 – 'N' are empty

>> Memory is wasted!

Good solution

Dynamic memory allocation and management

The size of the array is not assigned when it is declared

The size is assigned only when required, and with only the needed elements

>> Less memory is used (and only when needed)

2. Dynamic memory management

C functions for memory allocation

```
<stdlib.h>

void * malloc(int n_bytes)
void * calloc(int n_elements, int size_element)
```

`malloc` and `calloc`

(1) allocate the amount of memory specified as parameter

(2) return a pointer to the first position of this memory chunk

The pointer can be used as the first position of an array (remember that pointers and arrays are closely related!)

2. Dynamic memory management

C functions

`void * malloc(int n_bytes)`

n_bytes: memory size (in bytes) to be allocated

The values of the allocated memory cells are unknown (garbage)

`void * calloc(int n_elements, int size_element)`

n_elements: number of 'elements' of the memory chunk

size_element: size (in bytes) of each 'element' of the memory chunk

The values of the allocated memory cells are set to 0

Allocated memory **must be released** when it is no longer necessary

It is a responsibility of the programmer to pair up conveniently memory allocation and release

> Very powerful feature but also error-prone

```
<stdlib.h>

void free(void * p)
```

`p` is a pointer to the address of a memory chunk allocated with `calloc` or `malloc`

2. Dynamic memory management

Example

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   int *a;
   int n, i;

   printf("Array size? ");
   scanf("%i", &n);

   // allocate memory for new array a
   a = (int *) malloc(n * sizeof(int));

   // assign values to array elements
   for(i=0; i<n; i++) {
      a[i] = i;
      printf("%i ", a[i]);
   }
   printf("\n");

   // release memory
   free(a);
   a = NULL;     // <-- make sure that a is no longer used

   // end program
   system("pause");
   return 0;
}
```

Outline

1. External data storage
   a. Files
   b. Databases
2. Dynamic memory management
3. **Programs used in Engineering**

3. Programs used in Engineering

Description

Engineering tasks are supported by computer software

Engineering software

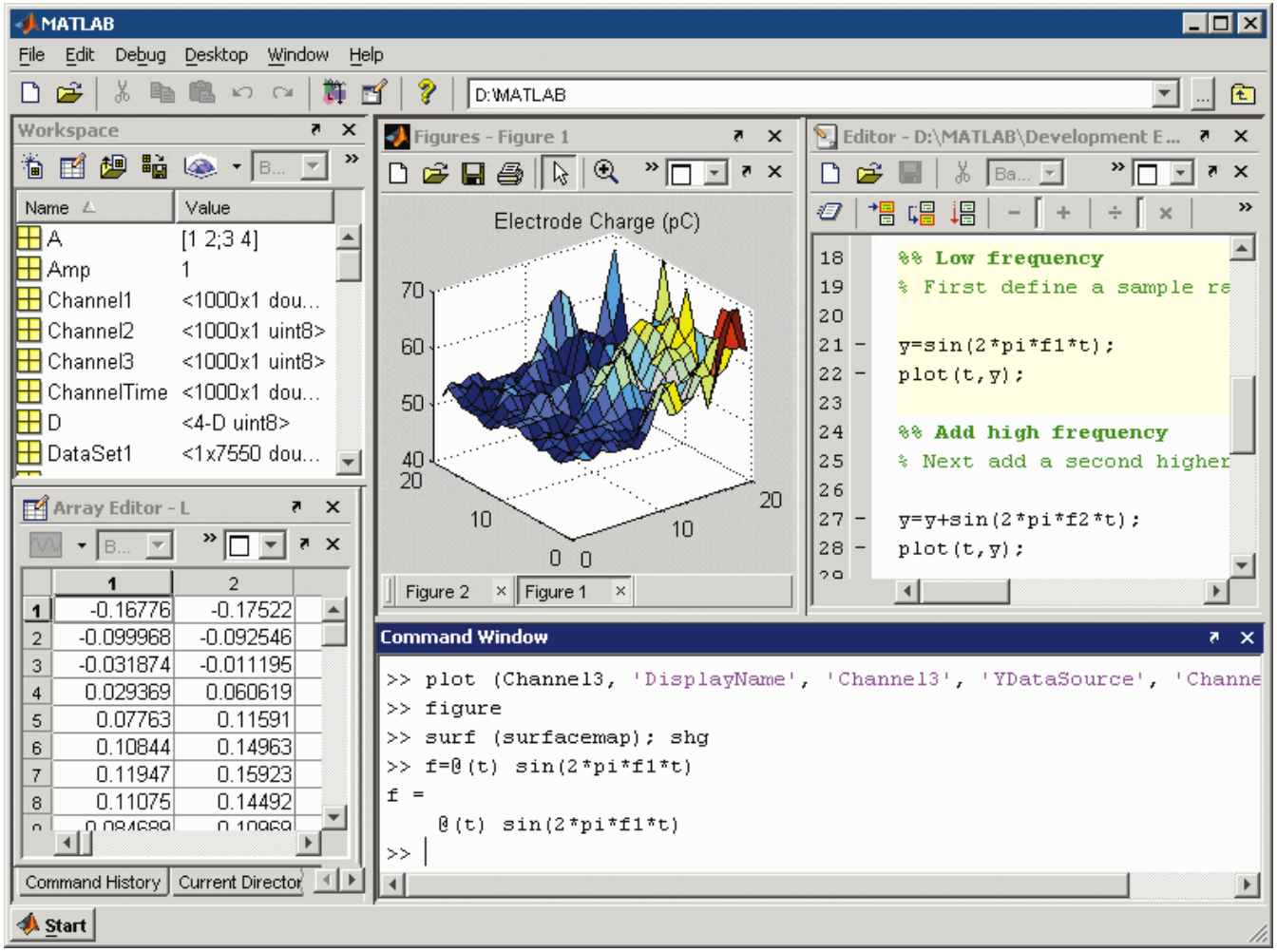
- Matlab
- Computer Aided Design (CAD)
  Computer Aided Manufacturing (CAM)
- Design, calculus, and simulation software

Office tools

As in any other professional activity

3. Programs used in Engineering

Matlab



3. Programs used in Engineering

Matlab

MATLAB® is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation

High-level language for technical computing

Development environment for managing code, files, and data

Interactive tools for iterative exploration, design, and problem solving

Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration

2-D and 3-D graphics functions for visualizing data

Tools for building custom graphical user interfaces

Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel

http://www.mathworks.com/products/matlab/

Computer Aided Design

Tools to support design and documentation process

The result of the CAD process is a design ready to be printed or manufactured

Graphical information, materials, process, dimensions, tolerances, etc.

Advantages

Optimization of the design process

Advanced functionalities not available in traditional procedures: rotations, simulation, etc.

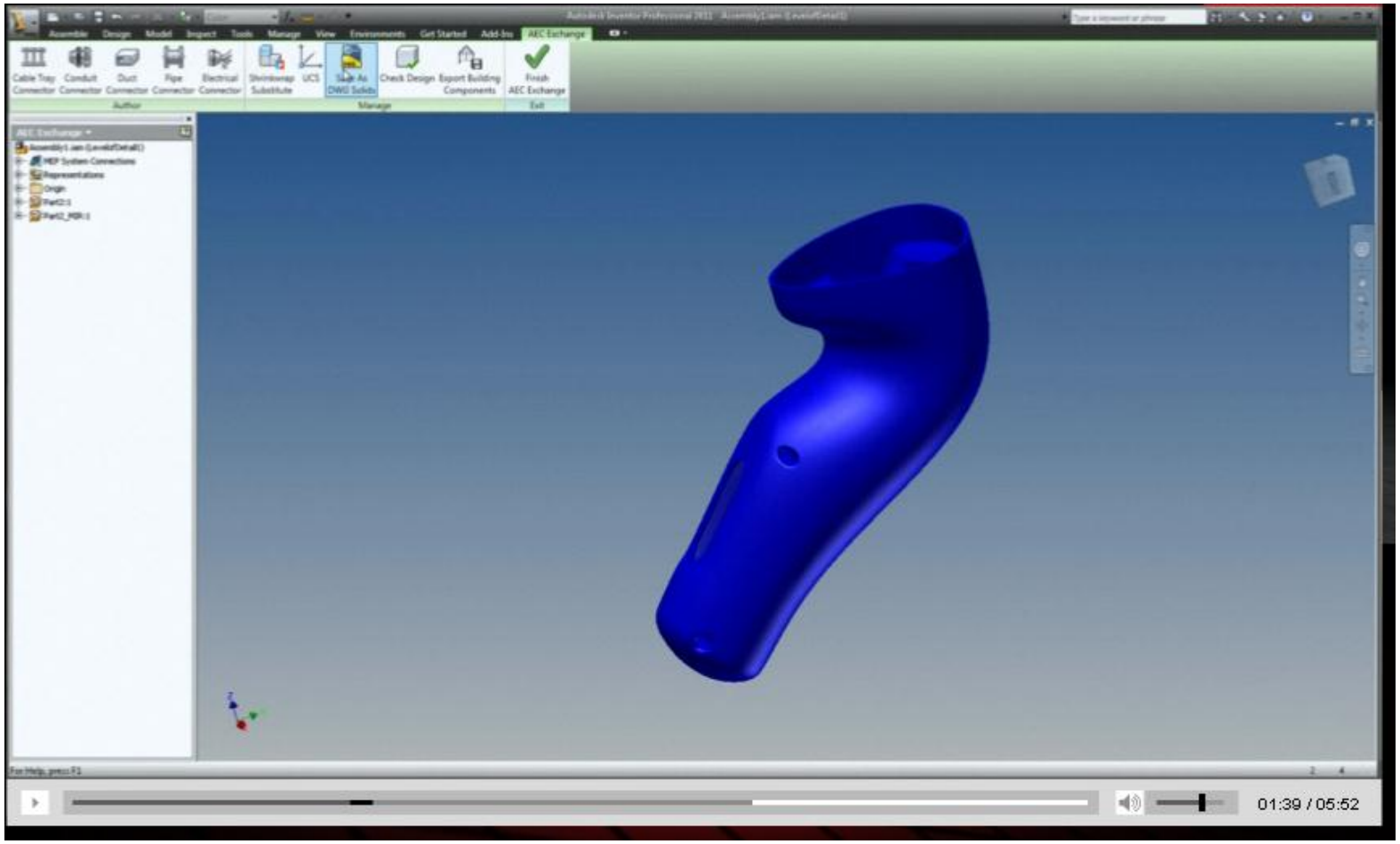
2D and 3D tools

2-dimension graphics: vector graphics (points, lines, arcs, polygons, etc.)

3-dimension graphics: solids, surfaces, etc.

3. Programs used in Engineering

AutoCAD®



http://adskmedia.com/conceptual-design/

Specific software programs to support different Engineering tasks

- Industrial Organization
  - Project management
  - Manufacturing management
  - Simulation and optimization of logistics and production process
    - E.g.: *Witness* environment used in *Quantitative Organization Methods*
  - Logistics management
  - Production and logistics systems
- Materials
  - Simulation of polymeric fluids behavior in injection molding
    - E.g.: *Moldflow* used in *Polymer Technologies*

3. Programs used in Engineering

Design, calculus, and simulation

- Electrical installations
  - Analysis and simulation of electric systems
    - E.g.: *PSE/E Power System Simulator for Engineering* used in *Electrical Grids and Circuits*
  - Planning of electricity distribution networks
  - Production and logistics systems
- Energy Technologies
  - Fluid dynamics calculus and simulation
    - E.g.: *FLUENT 6.2* used in *Computer-aided simulation of industrial flows*
- Robotics and automation
  - Robot programming (E.g.: *RAPID*)
  - Integration of instrumentation systems (E.g.: *LabView*)
  - 3D modeling, simulation and animation of physical systems (E.g.: *Roboworks*)

Bibliography

Recommended lectures

Basic

- Files
  - Ivor Horton. *Beginning C: From Novice to Professional*. Apress, 2006 (4th Edition) – Chapter 12
  - Stephen Prata. *C Primer Plus.* Sams, 2004 (5th Edition) – Chapter 13
- Dynamic memory management
  - Ivor Horton. *Beginning C: From Novice to Professional*. Apress, 2006 (4th Edition) – Chapter 7.4 (*Using memory as you go*)
  - Stephen Prata. *C Primer Plus.* Sams, 2004 (5th Edition) – Chapter 12.6 (Allocated Memory)

Additional information

- Stephen G. Kochan. *Programming in C.* Sams, 2004 (3rd Edition), Programming in C – Chapter 16 (*Input and Output Operations in C*), 17 (*Miscellaneous and Advanced Features)*

Outline

1. External data storage
   a. Files
   b. Databases
2. Dynamic memory management
3. Programs used in Engineering