# EEM103
# Computer Programming

## Week12

- File operations in C

1

## File operations

- Storage of data in variables and arrays is temporary—such data is lost when a program terminates.
- Files are used for *permanent* retention of data.
- Computers store files on secondary storage devices, such as hard drives, CDs, DVDs and flash drives.
- In this lecture, we will learn
  - how data files are created, updated and processed by C programs.

2

# Files and Streams

- C views each file simply as a sequential stream of bytes
- Each file ends either with an end-of-file marker or at a specific byte number recorded in a system-maintained, administrative data structure.
- When a file is opened, a stream is associated with it.
- Three files and their associated streams are automatically opened when program execution begins—the standard input, the standard output and the standard error.
- For example, the standard input stream enables a program to read data from the keyboard, and the standard output stream enables a program to print data on the screen.
- Opening a file returns a pointer to a FILE structure (defined in <stdio.h>) that contains information used to process the file.
- The standard input, standard output and standard error are manipulated using file pointers `stdin`, `stdout` and `stderr`.

3

- The standard library provides many functions for reading data from files and for writing data to files.
  - Function `fgetc`, like `getchar`, reads one character from a file.
  - Function `fputc`, like `putchar`, writes one character to a file.
  - The `fgets` and `fputs` functions, for example, can be used to *read a line from a file* and *write a line to a file*, respectively.
- The file-processing equivalents of functions
  - `scanf` and `printf`
  - `fscanf` and `fprintf`.

4

# Opening a file

- To read an existing file, open it for reading (`"r"`).
- To add records to the end of an existing file, open the file for appending (`"a"`).
- To open a file so that it may be written to and read from, open the file for updating in one of the three update modes—`"r+"`, `"w+"` or `"a+"`.
- Mode `"r+"` opens an existing file for reading and writing.
- Mode `"w+"` creates a file for reading and writing.
- If the file already exists, it's opened and its current contents are discarded.
- Mode `"a+"` opens a file for reading and writing—all writing is done at the end of the file.
- If the file does not exist, it's created.

5

# File modes

| Mode | Description |
|---|---|
| `r` | Open an existing file for reading. |
| `w` | Create a file for writing. If the file already exists, *discard* the current contents. |
| `a` | Open or create a file for writing at the end of the file—i.e., write operations *append* data to the file. |
| `r+` | Open an existing file for update (reading and writing). |
| `w+` | Create a file for reading and writing. If the file already exists, *discard* the current contents. |
| `a+` | Open or create a file for reading and updating; all writing is done at the end of the file—i.e., write operations *append* data to the file. |

6

# End of file markers (eof)

| Operating system | Key combination |
|---|---|
| Linux/Mac OS X/UNIX | *<Ctrl> d* |
| Windows | *<Ctrl> z* then press *Enter* |

7

# fseek ()

- The C library function **fseek** sets the file position of the stream to the given offset.

  int **fseek** (FILE *stream, long int offset, int whence)

- Parameters
  - stream − This is the pointer to a FILE object that identifies the stream.
  - offset − This is the number of bytes to offset from whence.
  - whence − This is the position from where offset is added. It is specified by one of the following constants
    - SEEK_SET Beginning of file
    - SEEK_CUR Current position of the file pointer
    - SEEK_END End of file

8