**Chapter 7: ERROR HANDLING**

**Topic – 1: Conventional Error Handling**

**Exiting**

* Works by importing ***stdlib.h*** library.

***exit (EXIT\_FAILURE);***

***exit (EXIT\_SUCCESS);***

**Creating Error Codes**

* Works by importing ***errno.h*** library.

***extern int errno;***

***errno = 404; // Custom error code***

***fprintf(stderror, "Error code: %d", errno); // Prints this string on screen***

* Keyword ***extern*** is used in **dynamic libraries**, so **GCC** will pop up warnings if used otherwise.
* And the error variable name must ***errno*** only, otherwise **GCC** will pop up error.
* We can use simple I/O functions too but writing code this way **increases readability**.

**Topic – 2: Handling Mathematical Errors**

* Works by importing ***errno.h*** & ***math.h*** library.

***sqrt(-2.0); // Trapped by EDOM***

***if (errno == EDOM) {/\*Statements\*/}***

***else if (errno == ERANGE) {/\*Statements\*/}***

* **EDOM** is used to detect mathematical errors due to **impossible calculation**.
* For example, the **example above**.
* **ERANGE** is used to detect mathematical errors due to **undefined return value**.
* For example, **log(0.0)**.
* Keep in mind that the error doubt statement is immediately before the checks.

**Topic – 3: File Related Error**

***FILE \*fptr;***

***fptr = fopen("filename.txt", "r");***

***if (ferror(fptr)) {printf("Error opening up the file");}***

* ***ferror()*** returns a **non-zero value** if some issue is faced with file.