C - File I/O

Just Before We Get Started

House-Keeping Rules

Let's Get Started

HouseKeeping Rules

- Stay attentive, we do not want YOU missing **anything.**
- Share your questions on the **Slido link**
- **Links and instructions** to participate in activities **will be shared in the Youtube chat**

Agenda

Activity	Duration(GMT)
Check in and recap	1:00 PM
C - File I/O	1:10 PM
Q & A Session	1:40 PM
Announcements & Next steps	2:00 PM

Pay close attention, we are now starting our:

C – File I/O

open

- **Purpose**: Opens a file or creates a new file.
- **pathname**: The path to the file.
- **flags**: Flags specifying the mode of the file opening (e.g., O_RDONLY, O_WRONLY, O_CREAT).
- **mode**: Permissions for the file when O_CREAT is used.
- Return Value:
  - On success, returns a non-negative file descriptor.
  - On failure, returns -1 and sets errno to indicate the error.

int open(const char *pathname, int flags, mode_t mode);

WHAT IS FILE DISCRIPTOR

A file descriptor is a non-negative integer that uniquely identifies an open file in a computer's operating system. In Unix-like operating systems, including Linux, macOS, and many others, file descriptors are a fundamental concept for interacting with files, sockets, pipes, and other input/output resources.

In C programming, functions like open, read, write, and close operate on file descriptors. When you open a file, the open function returns a file descriptor that you can subsequently use to perform read and write operations on the file.

Flags

The flags used in the open function for specifying the mode of file opening are defined in the **<fcntl.h>** header file. Here are some commonly used flags:

- **O_RDONLY**: Open the file for reading only.
- **O_WRONLY**: Open the file for writing only.
- **O_RDWR**: Open the file for both reading and writing.
- **O_CREAT**: Create the file if it does not exist.
- **O_TRUNC**: Truncate the file to zero length.
- **O_APPEND**: Set the file offset to the end of the file before each write.
- **O_EXCL**: Ensure that this call creates the file; if the file already exists, the call will fail.

close

- Purpose: Closes a file descriptor.
- fd: File descriptor to be closed.
- Return Value:
  - On success, returns 0.
  - On failure, returns -1 and sets errno to indicate the error.

int close(int fd);

read

*ssize_t read(int fd, void *buf, size_t count);*

- *Purpose: Reads data from a file descriptor.*
- *fd: File descriptor.*
- *buf: Buffer to store the read data.*
- *count: Number of bytes to read.*
- *Return Value:*
  - *On success, returns the number of bytes read.*
  - *On end-of-file, returns 0.*
  - *On failure, returns -1 and sets errno to indicate the error.*

write

- Purpose: Writes data to a file descriptor.
- fd: File descriptor.
- buf: Buffer containing the data to be written.
- count: Number of bytes to write.
- Return Value:
  - On success, returns the number of bytes written.
  - On failure, returns -1 and sets errno to indicate the error.

*ssize_t write(int fd, const void *buf, size_t count);*

dprintf

- Purpose: Writes formatted output to a file descriptor.
- fd: File descriptor.
- format: A format string, similar to printf.
- Return Value:
  - On success, returns the number of characters written.
  - On failure, returns a negative value and sets errno to indicate the error.

*int dprintf(int fd, const char *format, ...);*

Q&A Session

Share your questions on the slido link pinned on the chat or use the QR code

Announcements

Do not forget to join the next live learning session

See you at the next session!