Unit: Unit III

Unit III: Files in C
1. Brief Introduction
Files are a crucial aspect of programming, allowing for the persistent storage and retrieval of data. In C, files are managed through a series of high-level functions which provide a robust interface for reading and writing data. This unit provides an in-depth understanding of handling files, discussing their types, modes of access, and methods for error handling, all essential for effective file management in C programming.
2. Key Concepts
2.1 Streams in C
- **Streams**: A stream is an abstract representation of a source or destination of data. In C, all file operations are performed using streams, either for input (reading) or output (writing).
2.2 Types of Files
- **Text Files**: These files store data in a human-readable format. Data is arranged in lines of text, often separated by newline characters.
- **Binary Files**: These files store data in a binary format, meaning the data is not meant to be human-readable. Useful for storing complex data structures.
2.3 File Pointer Declaration
- Files are handled via file pointers, declared using the `FILE` structure.
```c
FILE *filePointer;
```
2.4 Opening and Closing Files
- Opening a file: `fopen()`
- Closing a file: `fclose()`
- Modes for opening files:
- `"r"`: Read
- `"w"`: Write (creates a new file)
- `"a"`: Append
- `"rb"`, `"wb"`, `"ab"`: Binary modes for reading, writing, and appending
2.5 Reading Data from Files
Functions:
- `fscanf()`: Reads formatted input from a file.
- `getw()`: Reads a word (integer) from a file.
- `fgets()`: Reads a string from a file until a newline or EOF.
- `fgetc()`: Reads a single character from a file.
- `fread()`: Reads multiple bytes of data from a file.
2.6 Writing Data to Files
Functions:
- `fprintf()`: Writes formatted output to a file.
- `putw()`: Writes a word (integer) to a file.
- `fputs()`: Writes a string to a file.
- `fputc()`: Writes a single character to a file.
- `fwrite()`: Writes bytes of data to a file.
2.7 Detecting End of File
- **EOF**: An indication that there are no more data to read from a file.
- Function: `feof()` checks if the end of a file has been reached.
2.8 File Access Types
- **Sequential Access**: Data is read or written in a linear order.
- Example: Using `fgetc()` or `fgets()`.
- **Random Access**: Data can be read or written at any given position within the file.
- Functions: `fseek()`, `ftell()`, `rewind()`, `fgetpos()`, `fsetpos()`.
2.9 Error Handling
- Functions for error handling: `clearerr()`, `perror()` helps manage operations that may fail.
2.10 Working with Files of Records
Reading and writing records (structured data) using `fread()` and `fwrite()` for data structures and arrays.
2.11 Other File Operations
- Renaming a file: `rename()`
- Removing a file: `remove()`
- Use of command line arguments for file processing.
3. Examples
Example 1: Opening and Closing a File