Samuel Steinhardt  
Ivy Tech Community College  
SDEV-120: Programming Logic and Design  
October 17, 2025

Understanding Computer Files and Their Role in Programming

Computer files are how programs store large amounts of data on non-volatile storage. Non-volatile means that the data will still exist even after a program closes or the computer shuts down. Programs typically perform most of their operations in RAM (Random Access Memory), which is extremely fast but volatile. This means that anything saved in memory will disappear or be overwritten once the program finishes running.  
  
If a program needs to keep information after it closes, it must move that data from volatile memory to a permanent location — this is where non-volatile storage comes into play. Non-volatile storage can take many forms: hard drives, solid-state drives, optical disks such as DVDs, or portable media like flash drives and SD cards. The most common example is the hard drive. Almost any time a program needs to save data for later use, it writes that data to a location on a hard drive. Non-volatile storage plays a vital role in how data is permanently preserved for later access (Farrell, 2023).  
  
A great example of this is configuration files in video games. Imagine if every time you wanted to play Black Ops 27: Quantum PC Edition — Now Powered by Hot Topic Angst, you had to manually remap every key, adjust your controller sensitivity, and fine-tune your graphics settings before playing. Without non-volatile storage, those preferences would be lost when the game closed. Thanks to hard drives, the game can save these preferences in a configuration file and reload them automatically the next time you play.  
  
Files are simply collections of data stored in a specific format. There are two main categories of files: input files and output files. An input file provides data for a program to read, while an output file stores data that a program writes. Just like variables in programming, files can have different types and purposes. The two most common formats are text files and binary files.   
  
Text files store human-readable data — such as configuration settings, logs, or saved scores — using characters. These can often be edited directly if the user needs to make manual changes, like improving game performance by adjusting hidden settings in a config file. Binary files, on the other hand, store data in a machine-readable format, such as executable files (for example, CallOfDuty.exe) or compiled code. Programs must interpret binary files correctly in order to use the data they contain.  
  
File Operations and Their Importance  
  
To make use of files, programs perform a series of basic file operations, which usually include opening, reading, writing, appending, and closing files. These operations are essential because they allow data to move between a program and permanent storage. For example, when a player adjusts graphics settings or control sensitivity in a game, the program must open a configuration file, write the new values to it, and then close the file properly so the data is saved. The next time the game starts, it will open that same file again and read the saved values to restore those preferences.  
  
Every major programming language provides functions to handle these operations. In Python, for instance, the open(), read(), write(), and close() functions are used to manage files. Using a statement like with open("settings.cfg", "r") as file: ensures the file is safely closed even if an error occurs, which helps prevent corruption or data loss. In C++, similar functionality is provided through file stream objects like ifstream and ofstream. No matter the language, these basic operations form the bridge between a program’s temporary memory and its permanent storage.   
  
Another important use of files is saving progress or state. Imagine you’re halfway through a massive story campaign or just unlocked a legendary weapon — it would be frustrating if that progress disappeared every time you closed the game. Save files exist for this exact reason. They allow programs to write important data, such as player progress or inventory information, into a file that can be accessed later. When the player reloads the game, the program reads the file and restores the saved data, creating the illusion of continuity.  
  
In general software development, file handling is just as critical. Many business programs store customer data, transaction logs, or configuration settings in files. Without proper file operations, all data would be lost when the program stops running. By using files correctly, developers can create programs that retain data, exchange information between systems, and maintain stability even after shutdown. Files provide persistence — the ability for data to outlive the process that created it — and that makes them one of the most essential tools in programming.

References  
Farrell, J. (2023). Programming logic and design (10th ed.). Cengage Learning.  
W3Schools. (n.d.). Python file handling. Retrieved from https://www.w3schools.com/python/python\_file\_handling.asp