**Mohammad Tehabsim**

**23110880**

**Systems Programming**

Contents

[What is system programming? 1](#_Toc166930160)

[Multiprogramming (multi-process) 3](#_Toc166930161)

[Multitasking (multi-threading) 4](#_Toc166930162)

[Thread results and parallelization techniques and how to handle it 5](#_Toc166930163)

[Describe -with aid of a diagram(s)- all the communication abilities of different processes in your simulator. 6](#_Toc166930164)

# What is system programming?

System programming deals with interlacement of software that can be used as a foundation to execute and control hardware, to interact with the OS directly, and manage system resources. It is narrowed down to creating and maintaining system software like operating systems, device drivers, embedded systems, and low-level libraries. Programming at the system level is usually in the C language or Assembly which have fine-grained control over resources and hardware.

However, on the contrary to application programming, it is low-level programming that makes the task easier. systems programming allows computers to communicate with each other where application programming allows humans and computers to interact and communicate. :systems programming allows computers to communicate with each other whereas application programming allows humans and computers to interact and communicate. :

* System Programming: It is about software which is used to ensure that the computer hardware works properly and has the ability to run application software. This is accomplished by monitoring memory, processes, devices, as well as system resources. It is a system which is based on firm knowledge of how hardware and the OS operate.
* Traditional Programming: Targets at solving typical user needs or tasks by means of applications that are running on an OS. This kind of programming is usually abstraction and is more higher-level and often depends on libraries and frameworks which hide the complexities of low-level hardware manipulation as well as resource management.

The usage of OS-Controlled Resources as OS manages hardware resources ensuring that each process has the necessary resources and maintain system stability and security. But in system programming our processes can have or use these resources for different purposes which are:

1. Memory Management: The OS allocates memory spaces to different processes and ensures that the memory usage of one process does not interfere with another. For example, in my code I create an array using malloc, but when we fork ( create a new process) now this array differ between each process which the OS manages, and its now my role to either synchronize it by using file descriptors or pipe for the processes to communicate over. Also, when sorting algorithms are executed, they operate in separate processes where each has its memory space, thus isolating them from each other and preventing accidental data corruption

.

1. CPU Scheduling and Process Management: for example, from my code the OS schedules CPU time for various processes like hashing a file, filling an array with random numbers, sorting, and searching within the sorted array. Each of these operations can be performed in separate processes using fork(). The OS ensures these processes are executed efficiently, possibly in parallel on multi-core systems, to optimize performance.
2. File System Interaction: for example, as my code involves direct file operations which are controlled by the OS and I manage files where sorted arrays are written or from where arrays are read for searching suing functions like open(), read(), and write() which are system calls that interact with the OS to manage disk files securely and efficiently. The OS handles disk scheduling and caching strategies to optimize file access.
3. Resource Cleanup: after the completion of processes, it’s crucial to release resources properly. As I do in my code by removing temporary files at the end of operations, which is managed through system calls like remove(), indicating to the OS to free up the used disk space.

A computer screen with text

Description automatically generated

# Multiprogramming (multi-process)

The multi-process concept in programming involves creating multiple processes, which are instances of an application, to execute different parts of a program in parallel. This is beneficial for optimizing performance and efficiency, especially on multi-core systems where processes can run simultaneously on separate cores. Multi-process architecture is widely used in applications that require concurrent execution of tasks, large computational tasks, or tasks that need to be isolated from each other for security or stability reasons.

**How did I implement Multi-Process Concept in my simulator?**

In my simulator I used the multi-process concept with the fork() system call in Unix-like operating systems. The fork() function creates a new process by duplicating the calling process and the new process is referred to as the child process while the original is the parent. This child process runs concurrently with the parent process and has its own separate memory space. I used it in order to achieve Multiple sorting and search algorithms each run in its own process together in order to sort it and search it as fast as possible as I used the following forking Processes:

* Hashing Process: Created a process dedicated to hashing a file. This allows the hashing operation to proceed independently of other operations.
* Random Array Generating: a process is forked to fill an array with random numbers.
* Sorting Processes: Multiple sorting algorithms are applied concurrently as each sorting algorithm (quicksort, insertion sort, merge sort, bubble sort) runs in its own process. This allows multiple sorting operations to proceed without waiting for each other.
* I managed to manage the communication between processes by using system calls to creating a file for the array not sorted, which the process that generates it write on the file, and then the sorting algorithms can read from it, each have its own file descriptor, and then each sorting algo finishes sorting it send it over pipe to one of the search algorithms and search for the target.
* Search Operations as I made search processes (binary search, Fibonacci search, jump search, ternary search) each in their own process. This allows simultaneous searching of the array using different algorithms.
* Synchronization and Process Management I used wait() or waitpid() to synchronize processes. This ensures that a process like file deletion or a subsequent process operation does not start until the necessary previous processes (like sorting) have been completed.
* Resource Cleanup At the end of the main function, there are commands to remove files, which are typically executed after all child processes have finished. This ensures that all temporary data used by the processes are cleaned up properly.

**Benefits:**

* Parallel Execution: Multi-process design provides the ability for your software to run various tasks concurrently capitalizing on the CPU and making it operate more efficiently.
* Isolation: Each process in its own memory space makes up the operating system. It facilitates the isolation of processes from each other which provides security and stability.

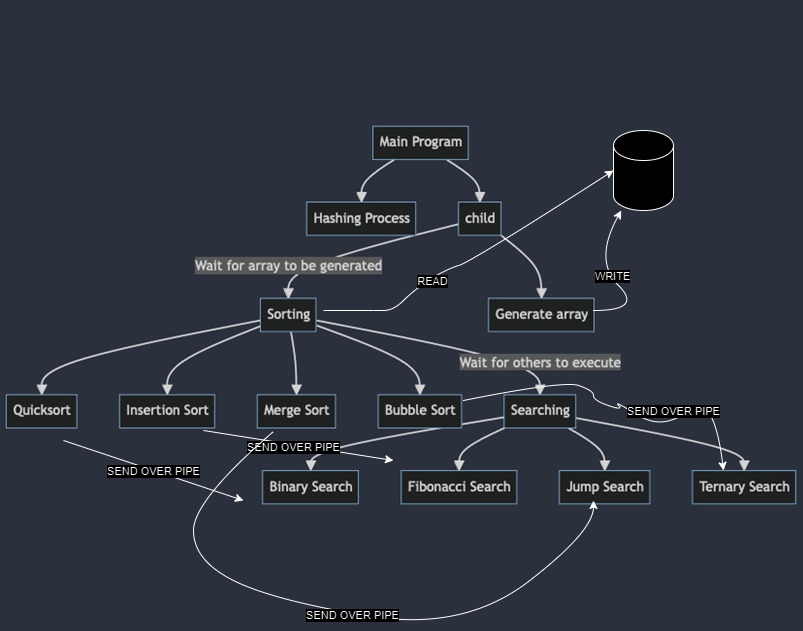
**Considerations:**

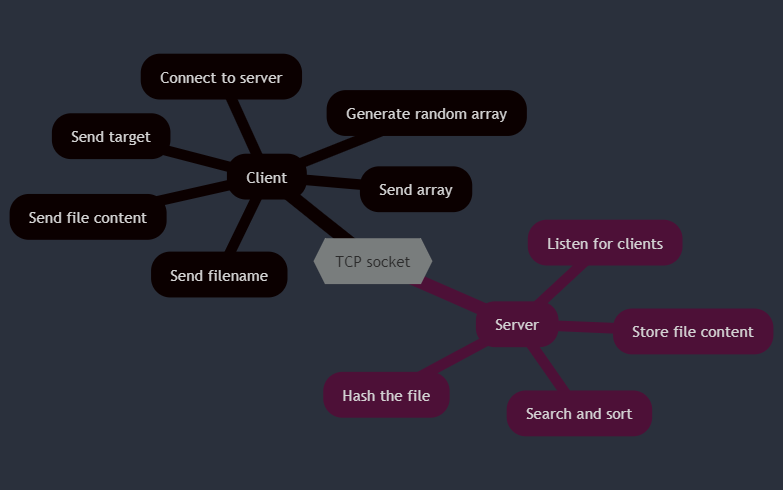
* Resource Management: Each process has its own memory area, which is a memory-consuming way.
* Complexity in Synchronization: Synchronizing among processes using wait() and inter-process communication leads to multiple lines of code that will make the code complex and harder to understand.

My code uses of multi processes for large sorting and searching problems simultaneously, indicating the way to increase the speed of processing with parallel processing but also made the code way more complex than if we did not use many processes.

# Describe -with aid of a diagram(s)- all the communication abilities of different processes in your simulator.

In my simulator we can see how the processes act together in order to perform hashing, sorting in four different algorithms and searching in four different algorithms, as for the data communication I uses the system calls to create, open, and read or write into a file, where they can communicate the not sorted array and then use pipes so the sorting processes can interact with the searching processes.





# Multitasking (multi-threading)

Multitasking is the ability of an operating system to work on more than one task at a time. Achieving this can be done by resorting to multiprocessing (by using many processes) or multithreading (by using many threads within the same process).

Multithreading means a program, or an operating system is able to manage its use by more than one user at a time or to manage and conduct multiple tasks at one time. In contrast to the multiprocessing paradigm where each process has its own memory and process state, multithreading enables multiple threads to share the same memory space and resources within the same process which can drastically enhance the execution speed of an application by reducing the overhead of memory usage and resource allocation.

Differences between Multithreading and Multiprocessing:

1. Shared Memory Space: Threads running within the same process are allowed to use the same memory space, which is a major advantage of inter-thread communication because it is faster and more efficient than inter-process communication.
2. Resource Efficiency: Threads are lighter weight in that they do not require more resources compared to processes because they share the resources of the processes and are thus less demanding.
3. Performance: Multithreading may be the most powerful method against large overheads for faster execution of tasks with a lot of waiting or I/O operations.
4. Responsiveness: In user-interface applications, multithreading can be used to keep a responsive user interface by detaching the user-interface from processing tasks.

How can I implement it in my code, I see the most perfect place for the threads is when sort and search for the array as where multiple algorithms run consecutively but independently, meaning there will be no issue because of the shared memory between the threads.

# Thread results and parallelization techniques and how to handle it

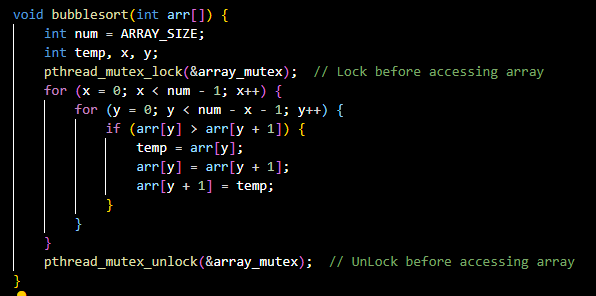
I implemented threading for sorting and searching algorithms to use better concurrency. However, this brought several challenges and considerations which needed to be handled, particularly around shared data access and the efficiency of using threads versus processes.

**Challenges Faced**

Concurrent Access to Shared Array:

Problem: All sorting algorithms are trying to access and modify the same array simultaneously leading to race conditions where the output of the sorting process becomes unpredictable and potentially incorrect.

Solution: Using locks was employed to ensure that only one thread could modify the array at any given time. Preventing data corruption but it is not concurrent anymore and the simulator would take longer periods to sort the array.



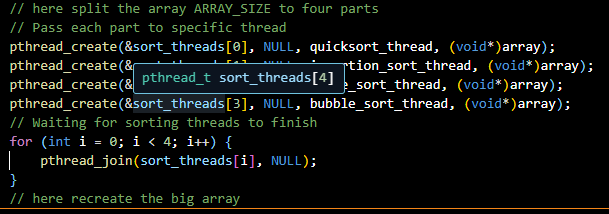
**Performance**

Slower Than Multi-Processing as locks which are used to ensure consistency, effectively serializing parts of the algorithms which are dealing with the array modification. Despite the fact that threading avoids the heavyweight overhead of process creation and inter-process communication, the locks can make the threads inactive, waiting for the release of the locks. This is a more severe situation when it comes to big arrays because the time of locking and unlocking becomes a noticeable factor.

**Alternative Parallelization Techniques**

Give each thread its own data and that either by making each thread its own copy or dividing array into four parts: Instead of sorting the entire array in each thread, divide the array into four parts, with each sorting algorithm working on its section. This can be an effective way to minimize locking overhead and utilize CPU resources better.

Implementation: Each thread sorts a quarter of the array independently. Once all parts are sorted, a final merging step could be required if a fully sorted array is necessary.



# Client server communication

I implemented the communication between the client and server using a TCP WebSocket, which enable a communication tunnel between the client and the server and they can send over the data, now if it is on my machine it will be two processes each represent either the server or the client, starting from the client processes it generate a random array and send it over the socket, the first number of the array is considered the target number to search for, it also opens up the file the clients wants to hash and send its content over the network, in this case since both of them are on the same device it is not necessary but when they are not on the same machine, this means that the files are not shared between the devices so we need to send the file content in order to hash it.   
After using the system calls to open the file and send it, it waits for the response from the server process, which in its turn receive the array, the file name, and its content, which results in using system call to store the received content in a new file under the same name. From here it uses system programming to initiate a process to hash the file content and new one to sort and search, which they communicate over a pipe so once the hash is finished it sends it to the parent processes, in the meantime that process start a new process for sorting and another one for searching, once the sorting is done the sorted array is sent over pipe to the searching process and when it finishes it sends the result to its parent, which now have both the hashed value and the sorted array and the index of the item to search for so it sends it back to the client as a response. Now if they are on different machines, it would be the same process, but the client processes would be on a machine different than the one on the server machine which because of this we need to send the file content over the socket.