

CS551 - Assignment 1

1. Design of Program :

1. To complete this assignment, we add all necessary header file, which are required for the run the code on different operating system.
2. From starting of the program, we have defined the mutex, which will help us to lock and unlock purpose, through that we are easily able to write and read data from the files.
3. Here, we used the open system call, it will help us to create the .txt files, and we are checking the values of file descriptor for validation purpose, if the value of the file descriptor is equal to -1, then it would be failure of cases.
4. To write the data into file, we have used the same file descriptor, which takes the values of the creating file, then we passed the write_buff, which we have defined through the for loop, now through that we pass the file descriptor, write_buff and size of the write_buff.
5. Now, after the writing the data into file, we have closed the files for further operation.
6. To read the data from the file, we accessed the same file using the file descriptor, and we passed the read file buffer to store the data into read file buffer.
7. To calculate the time taken for each client to read and write the data, we have used chrono steady time operation which tells the exact time taken after each client operation.

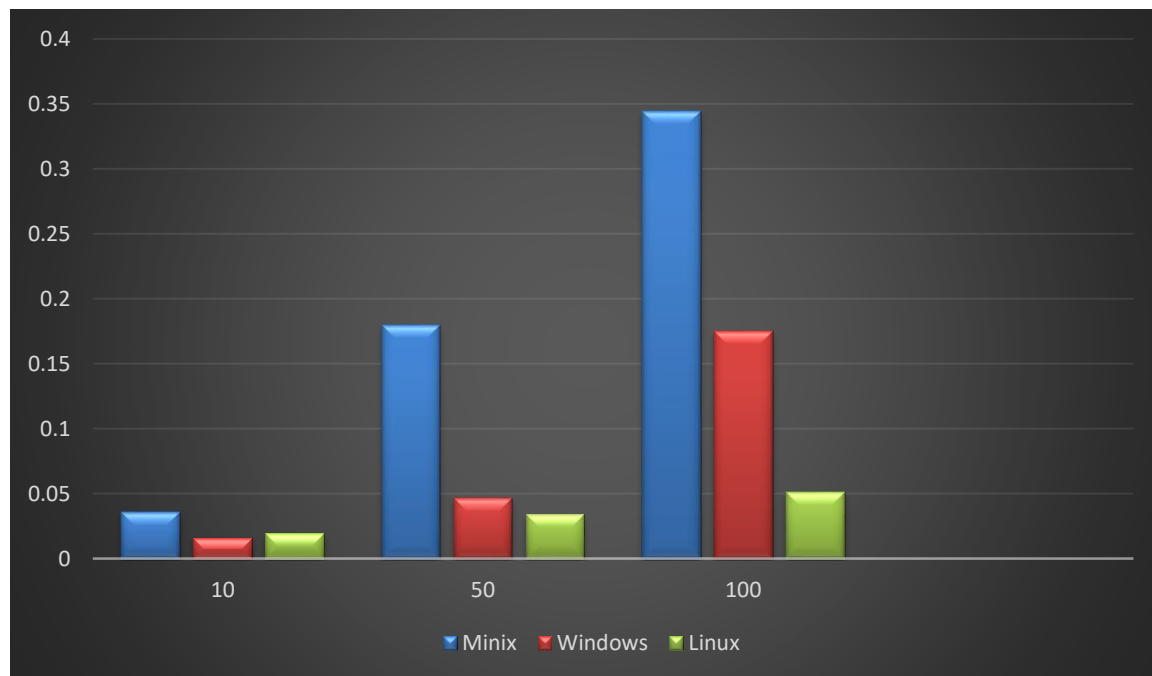
2. System Specification :

1. Linux : 2 GB RAM , 20 GB Memory
2. Minux : 2 GB RAM , 20 GB Memory
3. Window : 16GB RAM, 1TB Memory
 - We have installed this specification on virtual box on windows system.

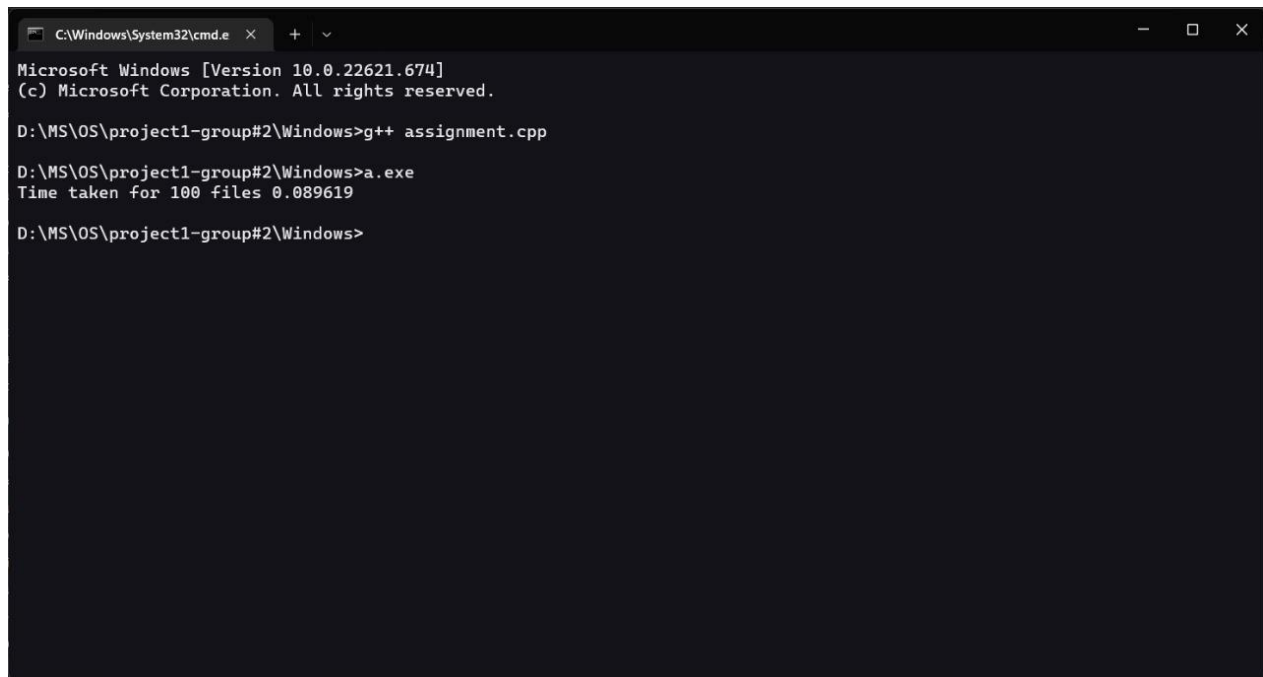
3.Data :

2. Here, the attached bar charts visualize the data for three different clients like 10, 50 and 100, on three different operating systems such as windows, Linux, and minx.
3. The X- axis represents the operating system and Y-axis represents the number of seconds required to complete the given operation.
4. From the code output, we can see that the Linux operating system takes less time for executing read/write operations compared to the other two operating systems.

Time taken for the various clients on different operating system



Snapshots of output on Window:



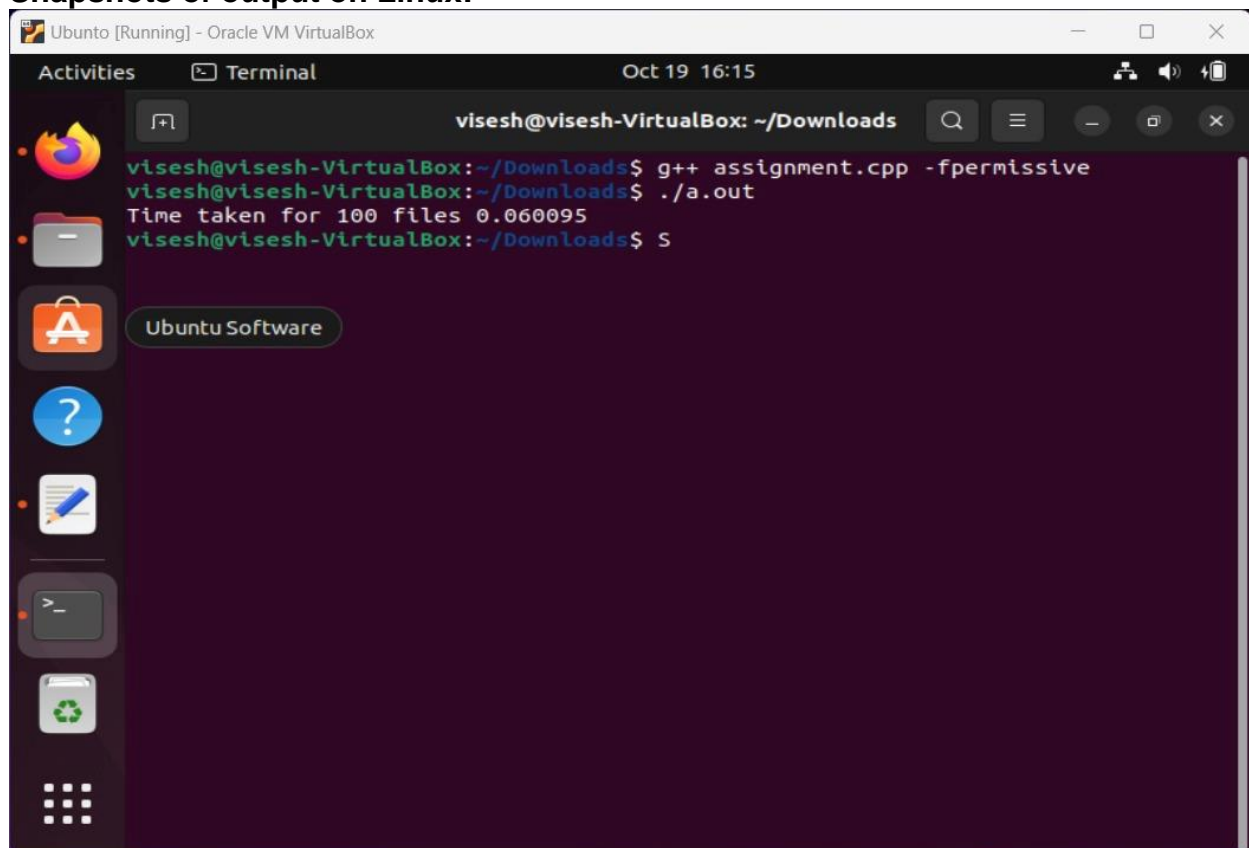
```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.674]
(c) Microsoft Corporation. All rights reserved.

D:\MS\OS\project1-group#2\Windows>g++ assignment.cpp

D:\MS\OS\project1-group#2\Windows>a.exe
Time taken for 100 files 0.089619

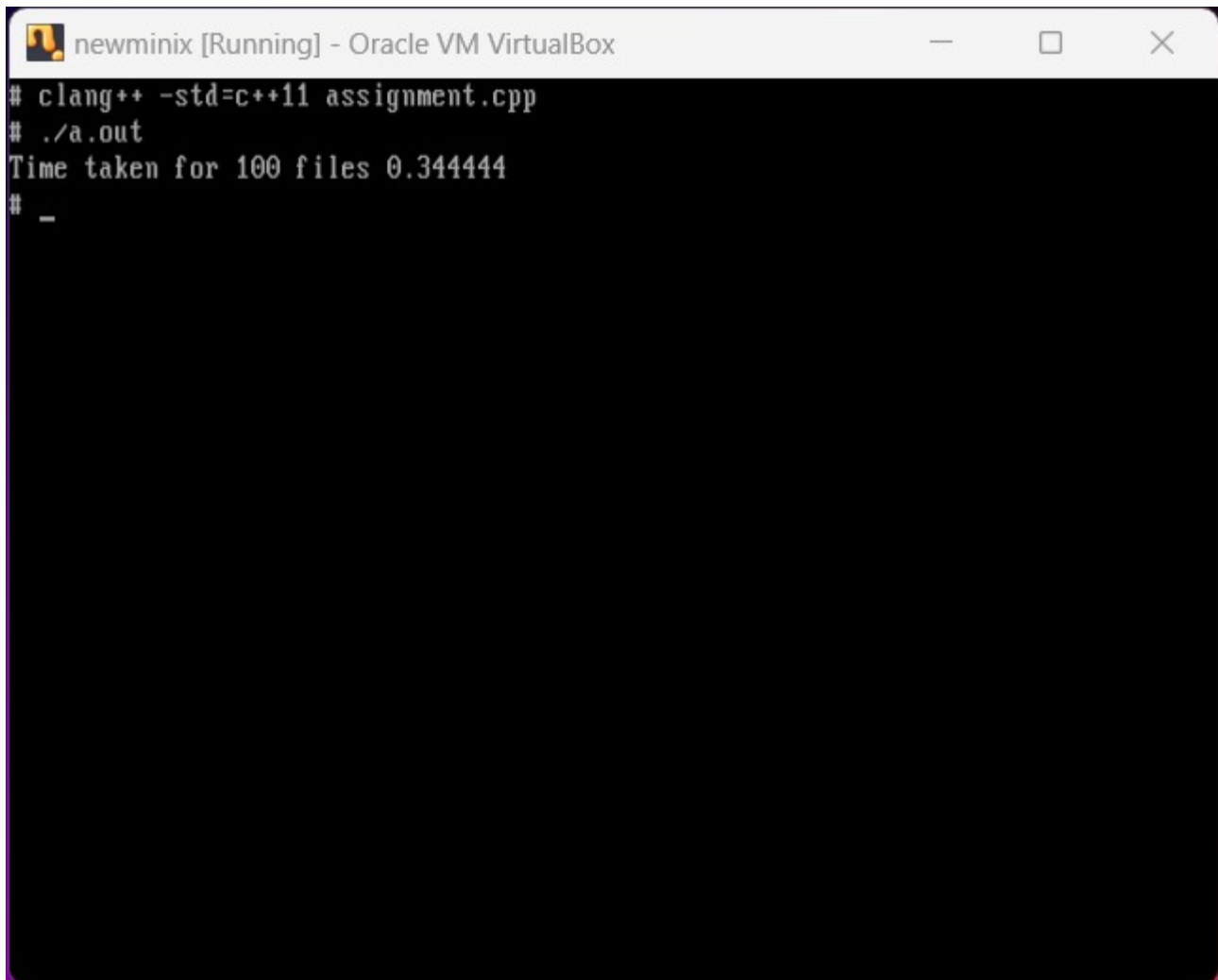
D:\MS\OS\project1-group#2\Windows>
```

Snapshots of output on Linux:



```
Ubuntu [Running] - Oracle VM VirtualBox
Activities Terminal Oct 19 16:15
visesh@visesh-VirtualBox: ~/Downloads
visesh@visesh-VirtualBox:~/Downloads$ g++ assignment.cpp -fpermissive
visesh@visesh-VirtualBox:~/Downloads$ ./a.out
Time taken for 100 files 0.060095
visesh@visesh-VirtualBox:~/Downloads$ s
```

Snapshots of output on Minix:

A screenshot of a terminal window titled "newminix [Running] - Oracle VM VirtualBox". The terminal has a black background with white text. The text shows the execution of a C++ program using clang++. The output indicates that the program processed 100 files in 0.344444 seconds. The prompt character is a hash symbol (#).

```
# clang++ -std=c++11 assignment.cpp
# ./a.out
Time taken for 100 files 0.344444
# _
```

Questions:

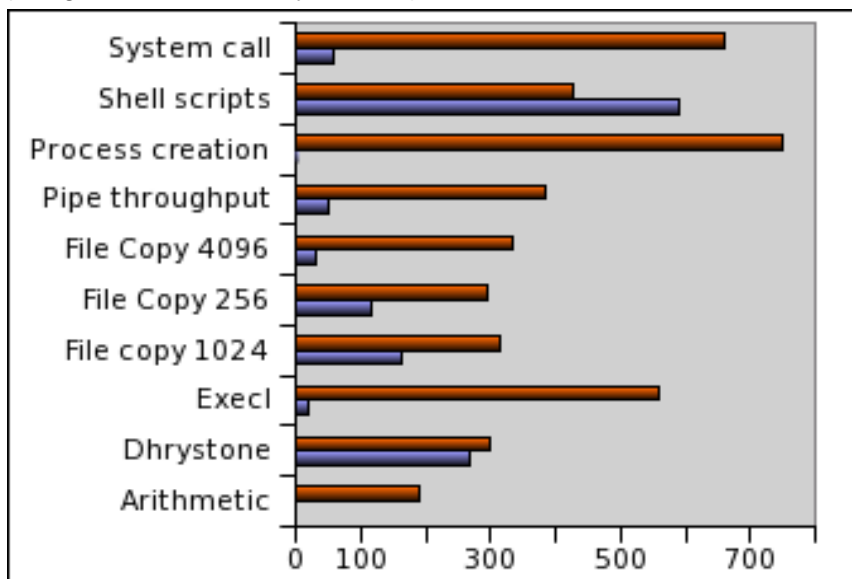
1. Performance comparison between Linux and Windows OS?

Linux powered PCs are faster than Windows. The main reason is that Linux is a lightweight system and Windows is with lots of unnecessary software. A lot of systems running in the background makes your windows PCs slow. Another reason for that is, file systems are pretty organized in Linux. Files are located in chunks and are closer to each other, which makes read-write operation way faster in Linux .That's the reason, most cloud systems run on Linux. Even Microsoft runs Linux to run Azure cloud. We can run Linux from Super Computers to Game Consoles, Smart TVs, Smartwatch, Car infotainment systems, Flight entertainment systems, Self-driving cars, Nuclear Submarines and many more. NASA relies on Linux for data transmission from satellites and telescopes.

2. Performance comparison between Linux and Minix?

Minix is used to build Linux. Linux kernel creator is Linus Torvalds, and Minix creator Andrew Tanenbaum. Torvalds' decision to design Linux as a monolithic kernel, not a micro-kernel, which Minix was. Linux has become the most used piece of operating systems in history, and Minix has found use in very specific applications, such as network routers and switches. Micro-kernel are not being used in general purpose systems. Linus Torvalds actually used GNU/Minix to create GNU/Linux, because Minix was the only Unix like. The measured system call overhead for Minix is a full ten times higher than the value in Linux. Linux is almost 10 times faster than Minix when the file copy tests were performed. Pipe throughput differed by a factor of seven. Minix is 140 times slower at process creation.

(Image below. Courtesy lwn.net)



3. Why with a smaller number of clients the programs run faster?

The more the clients are the more processing overhead the operating system takes, which makes it run slower once the number of clients goes up. This can be compared to the number of people who access a specific system, which makes the system slow. So if we reduce the number of clients from 100 to 50, then to 40 then to 30, in all those cases the processing time reduces.

4. How does threading work in the OS?

A thread shares its memory with the parent process and other threads within the process. Inter-process communication is slower due to isolated memory in processes. Inter-thread communication is faster due to shared memory with processes. They are a popular modern programming abstraction. They provide multiple threads of execution within the same program in a shared memory address space. They can also share open files and other resources among them. Threads allow for concurrent programming and, on multiple processor systems, so true parallelism is attained.

5. What resources are shared between threads?

Threads share all segments except the stack frame. Threads have independent call stacks, however the memory in other thread stacks is still accessible and in theory we can hold a pointer to memory in some other thread's local stack frame area.

6. How shared data and resources are managed among threads?

Thread synchronization is the mechanism which ensures that two or more concurrent processes or threads do not simultaneously execute some particular program segment (*critical section*). Processes' access to critical sections is controlled by using synchronization techniques, like mutex and semaphore for example. When one thread starts executing the critical section of the program(a serialized segment of the program) the other thread should wait until the first thread finishes the task. If proper synchronization techniques are not employed, it may cause a *race condition* where the values of variables may go unpredictable and vary depending on the timing of context switching of the processes or threads.