PROGRAM 6 REPORT

1. TIME MEASUREMENT

The time measurement tool is very beneficial in measuring the execution time when determining application performance metrics. It is useful for both application developers and end users. The goal of using this tool is to determine areas of the code that are taking a while to run and seeing if another approach should be implemented. Time measurement is a vital aspect to efficiency and can cause your programs to complete in a much faster time. The tool outlined in this document will be of use for programmers trying to get the most efficiency out of their code.

1. PROGRAM IMPLEMENTATION

To start using the time measurement tools, you need to include the header file <time.h>. This will give you access to all of the functions of that library. After that the tools will then be at your disposal. All the tools included in this kit are all related to timing.

The tool that I used is included in the library. I used the clock time tool to time a portion of my code. This tool can be set to a variety of different kinds of clocks that are included in the library. It also allows you to set the kind of time you are measuring in. This can be useful when dealing with other time measurements such as nanoseconds or milliseconds.

The clock that I am using is the CLOCK\_MONOTONIC. I want to use this one because it cannot be set, and the time is monotonic. Meaning that it represents the absolute elapsed wall-clock time. This is useful for my program because I just want to derive the amount of seconds it takes to run a portion of my code. It also has the CLOCK\_MONOTONIC\_RAW option that prevents the clock from being altered by adjusting the system time. There are other clock options such as CLOCK\_REALTIME, which wakes up the processor if its in power saving mode and CLOCK\_SLEEPTIME, which is when the processor is not in a power saving mode. But I will not be needing those for my example.

To use this tool, I set two-time specification variables with the before and after time. Then I used the function clock\_gettime(CLOCK\_MONOTONIC, &before) to indicate the before and after time points. To do this I wrapped the before and after around a section of my program 3 code that I wanted to time. I chose the triple for loop for this example. This will get the time to execute each iteration of the triple for loop. Note that the time at which the clock starts, and ends will not matter for this example, it will simply mark the time and do a simple calculation to find the seconds elapsed. After wrapping your code with the two functions, in a separate resulting variable immediately after the after time, subtract the after from the before time. To subtract the time in seconds you will need to indicate that the timespec variable is in seconds. To do this you put after.tv\_sec. The result variable will now contain the time to execute the triple for loop in seconds. You can find my code and script in the rest of the file I have provided.

1. WARNINGS

It is important to take into account that when timing your code you need to make sure that the code is the only thing running. For example, if you are timing the same code twice and you have an anti-virus scan running in the background this can greatly impact the runtime for the two separate tests. It can cause a randomized runtime making it impossible to determine if it is accurate. Always, make sure that you are timing in an uninhibited setting throughout all your runs.

Another thing to look out for when timing your code is the impact from user input. If you find your runtimes differ greatly from each attempt, check to see if user input is being timed. If you find that it is, simply start the timing after the input is received. This will avoid the user interaction being timed and will allow you to get a good estimate of the runtime.

Lastly, another thing to notice when using this tool is that randomization can greatly affect your runtimes. When generating random numbers, the timing is inconsistent. If you need to use randomization then make sure that you are using a fixed seed. This will allow the randomization to be always using the same seed which will ensure that the time it takes to complete is consistent with each run.

1. BENEFITS

After using this tool, I discovered that it can be very powerful in optimizing code. It is a fast and easy way to determine which parts of your code are taking the most amount of time. This can lead to having a more optimized program.

I noticed that timing can also be used for debugging. To debug, you will have to capture an area of code that you are concerned about and set your timing markers. After running you can see if the timing is not what you expected, if so, then it might indicate there is something wrong with your code. This isn’t necessarily the best way to debug in certain instances, but for performance debugging it can prove to be beneficial.

Timing is a good way to drive the strategy and direction of the organization of the code. With proper timing points, you can ensure that the code is maintaining high efficiency when being developed. This strategy can help in making difficult decisions and steer you in the right direction in the end. This is useful in the workforce because it demonstrates a high level of performance that can set you apart from other coders. It is always better to have more efficient code.

Lastly, practicing with this tool can prevent roadblocks. Since it drives good strategy and planning, it is reasonable to assume that it will help with all aspects of your coding process. If you can improve technology that already exists to make it run faster, it could potentially open up new opportunities for not only you, but other coders. It is very beneficial for companies when an employee has this skill. It can provide a more efficient work environment, drive public relations and lead to financial success.

1. DRAWBACKS

The timing tool is very useful, but it does take a while to set up. When setting up the tool, it can take time to determine where to put it. It is good to practice with it to get used to the set up. Having a good understanding of how it works is key to making the most of it. Also, make sure you understand the differences in the clocks you are using and what they do. They have different functionalities and can be useful, but if not understood, can prove to be a nuisance.

Another drawback to using this tool is that it can be misused very easily. Timing with inconsistency is a very big concern. I think that with good practice this could become less of a concern, but without practice, it could cause inaccuracy in your code. It has the power to make your code more complicated. Furthermore, you must be very careful with what you are timing and have a solid understanding of what is going on when you put those gettime functions in.

1. MORE FUNCTIONALITY

The <time.h> library has another function called clock resolution. It is accessed by using int clock\_getres(clockid\_t id, struct timespec \*tsp);. Like the get time function the clockid is the type of clock you want to use. The struct timespec is the time specification variable. This function returns the clocks resolution. The resolution is the smallest possible increase of time the clock model allows. This allows more functionality to your timing.

1. FUTURE

I can see how this tool might be useful in my future coding career. I would like to start implementing it into my coding practice. I think it could be useful in enhancing the quality of code I am producing. The use of this code could have a night and day difference when demonstrating my practices to companies during interviews. It would be a nice little touch on top for my skills.

I really enjoyed working with the tool it was very easy for me to understand and supplied me with another method of timing. I would like to try debugging with it because I think it could be a very interesting method. I would like to see what the output is when the time is not expected.

I also would like to start working with the different clock modes and seeing the difference in the runtimes with power saving mode enabled and when it’s not. Or try setting the time and pausing it so I can get multiple reads on different areas in the code. In this project I only timed the triple for loop. I would like to take the information I learned from this project and try experimenting with it, just to fully understand how each little thing can affect the time.

Works Cited

1. Sterling, T. L., Anderson, M., Brodowicz, M., & Bell, G. (2018). *High performance computing: modern systems and practices*. Cambridge (Mass.): Elsevier.