Execution Of C++ Programs In Shortest Possible Time (Nanoseconds Or Milliseconds)

Zaher Saif Al-Hashami

Email: ac.res@hotmail.com

ABSTRACT: Many programmers have multiple ways for programming in programming languages. Can reach the same result for a particular function works in a short time compared with the same function in another program to ensure access the result in a longer time. Difference between good and excellent programmers is reaching to the last result in short time and short way algorithm. In C++ programs could measure execution of some programs with Nanoseconds and Milliseconds. In this article loop and condition statements in C++ programming could measure them execution in little Nanoseconds or in Milliseconds. Comparison and approach between some programs algorithms are the method used in this paper. Time Execution taking of some C++ programs is depending on the algorithm of the program and statement time take. Preprocessor has role in speedup some Nanoseconds of execution of some C++ programs as compile time (convert to source code). These preprocessor statements like #define and #include preprocessor directives. There are many circumstances control the speed of some programs execution. Architecture of programming progress as well as hardware has role in speed of execution. The algorithm used in some C++ programs and execution of some functions is the topic of this article.

Keywords*:* Milliseconds, Nanoseconds, Comparison, preprocessor, #define, #include

1 INTRODUCTION

As known, the computer operates two different time scales. The first level is execute instructions at a rate of one or more per clock cycle. While each clock cycle requires only around one nanosecond. This state time nanoseconds executed based on the algorithm of the program or the way of a programming as well as the hardware has role to state if the program speed. The second is one is on macroscopic scale, which the processor must respond to external events. There are commonly multiprogramming systems inside the computer, and according to Ugur Halıcı, 2007, processes are performed in a pseudo parallelism as if each process has its own processor. Also based on this author there is only one processor but it switches back and forth. So when he said execution of a process, he meant that the processor's operations on the process like changing its variables, which all these done inside the computer in the processor. Input and output data (I/O) work meant the interaction of the process with the I/O operations like reading from x file or writing to x file. Based on these definitions, he classify programs as two types, the first is processor bound program which is a program having long processor bursts. The second is bound program a program having short processor bursts [1], (Halıcı2007). According to Roberto Cipolla, 2004, a program go through three phases: the first is editing (writing the program) the second phase is compiling like translating the program to executable code and detecting errors and the fourth is running the program and checking for logical errors which can call it debugging in the programming. First the editing phase, consists of editing a file by typing in the C++ program with a text editor and making corrections if necessary. On the disk the program stored as text file. The extension file.cc indicate that it is C++ program. Next phase is the compiling; the compiler translates the C++ program into machine language code which it stored on the disk as a file. Then linker then links the object code with can call it standard library routines which the program may use and creates an executable image and saved on the disk. The last phase is execution, it is loaded from the disk to memory and the computer's processing unit executes the program one instruction at a time [2], (Cipolla, 2004). As what familiar that one second is equal 1000 milliseconds and also equal 1000,000,000 nanoseconds. There are many Algorithms in the C++ programs can programming it by them in short or long ways. Could be compared some programs executed in nanoseconds with other longer that doing the same functions.

1.1 COMPILING IN C++

During testing a C++ program, it must compile the program before running it. The compilation process is convert the program written in human as readable language like C, C++, java and vb.net into a machine code. It directly understood by the Central Processing Unit inside the computer. There are many stages involved during creating a executable file from the source file. These stages include Preprocessing, Compiling and Linking in C++. This means that even if the program gets compiled. Hence most IDE (Integrated Development Environment) like Eclipse, Geany and others consider the term build for transforming source code file to an executable file. There are two phases define compilation in C++ programming: the first can call it the preprocessing phase and the second is compilation phase. In the preprocessing phase, the preprocessor changes the Preprocessing according to the directives mentioned starts with hash # sign. The C++ preprocessor takes the program and deals with the # include directives. For example in C++ program #include<iostream> will tell the preprocessor to read all the contents of the iostream header file and include the contents into the program and generate the separate C++ program file. C++ language supports many preprocessor directives like #include, #define, #if and #else. After preprocessing come the compilation which translate C++ program into assembly low level. The compiler takes the preprocessed file without any preprocessor directives. After that can generates an object file containing assembly level code then therefore, the object file created is in the binary form. In the created low level object file. In each line describes one low level machine level instruction.

This assembly phase converts object files in assembly code into machine level instructions. The created file is a re-locatable object code. Hence, the compilation phase generates the re-locatable object program in different places without have to compile again [3], (Prashant, 2013).

2 COMPARING 1

2.1 PROGRAM 1

```
#include <iostream.h>
void main()
{int d1, d2, d3, d4, sum = 0 ;
float average ;
cout << "Enter Four Degree : " ;
cin >> d1 >> d2 >> d3 >> d4 ;
sum =(d1+d2+d3+d4) ;
average = sum/4 ;
cout << endl ;
cout << "The Average of Student is " << average << endl ;}
```

The above C++ program is about calculating the average of four values with usual stages, starting with entering the four numbers d1, d2, d3, d4, then sum these numbers by sum function. Then, come the last function to get average calculating function and which take average = sum/4 statement. In the last statement in this average example of C++ program count the average after identify it. In this program there are some nanoseconds spent for execution and there are extra statements added that could spend. The program below spends approximately 10,000,000 nanoseconds (0.01 seconds) for execution, which measures computer device clock start and end. It an example for some measured execution of some C++ programs:

```
#include <time.h>
#include <iostream>
using namespace std;
int main()
{clock_t start, end;
start = clock();
//perform calculations for which performance needs to be checked
end = clock();
cout << "Time required for execution: "
<< (double)(end-start)/CLOCKS_PER_SEC
<< " seconds." << "\n\n";
return 0;
```

2.2 Program 2

```
#include <iostream.h>
void main()
{int d1, d2, d3, d4 ;
cout << "Enter Four Degree : " ;
cin >> d1 >> d2 >> d3 >> d4 ;
cout << endl ;
  cout << "The Average of Student is " << (d1+d2+d3+d4)/4 ;
```

This program is the same of above which calculate average of d1, d2, d3, and d4 numbers with short statements. Example sum = (d1+d2+d3+d4) this statement is not mentioned in program2 while it exited in program1, also can be seen that this statement average = sum/4 not mentioned in program 2. These two statements in program1 are longer and not concise compared in the last statement of program 2. Nanoseconds execution approximate time of program2 is less that the execution time of program1. These two programs are typical example of how to concise a short time execution of C++ programming by short concise statements.

In the below C++ program about stopclock, the execution time spends approximately 0.832 seconds (832 000 000 nanoseconds or 832 milliseconds)[4]:

```
#include <boost/chrono.hpp>
#include <cmath>
int main()
{boost::chrono::system_clock::time_point start =
boost::chrono::system_clock::now();
for ( long i = 0; i < 10000000; ++i )
std::sqrt( 123.456L ); // burn some time
boost::chrono::duration<double> sec =
boost::chrono::system_clock::now() - start;
std::cout << "took " << sec.count() << " seconds\n";
return 0;}
```

In this program the loop statement (for) took some time before execution during calculating the duration in seconds 0.832 seconds. As define or undefined statement (for), it will take more time calculations for execution.

3 COMPARING 2

3.1 PROGRAM 1

```
#include <iostream.h>
int main ()
{int n ;
cout << "Enter the starting number : " ;
cin >> n ;
while (n>0)
{cout << n << ", " ;
--n;}
cout << "FIRE!" ;
return 0 ;}
```

In program 1 above in comparing 2 there is execution of while condition statement till a stated case (n>0). This program will execute --n (n-1) until face this condition, then if the condition not attain it will send this statement cout << "FIRE!".

3.2 PROGRAM 2

```
#include <iostream.h>
int main ()
{int a, b;
int result;
a = 5;
b = 2;
result = a - b;
cout << result;
return 0;}
```

In program 2 of comparing 3 a subtraction calculation is written. The program started with identifying a, b and result. This identification is as int and it gives a, and b numbers. Then identify result=a-b. The last statement is: cout << result. In program 2 of comparing 3 can be seen that executing subtracting calculation directly without any loop or without any condition that must attain and that will fast execution than program 1. In program 1there is loop and while condition statement which may continue the execution till stated case (a condition). Program2 an example of short time execution of some statements C++ language. In program2 milliseconds or even a few nanoseconds that may time spend to execution comparing with program1. By give it condition (n<1000) ++n and it approximately take 0.1 seconds (1000, 000,000 nanoseconds) or 1000 milliseconds. In program 2 the result of execution time as approximate take 0 seconds.

4 COMPILE TIME EXECUTION

According to sources, that the identification of compilation it is like just in time compilation (JIT), it a method to measure and improve the runtime performance of computer programs based on byte code. When byte code is interpreted, it executes slower than compiled machine code, that could be performed before the execution and which during the program loading and also during the execution make it slow. Compile time execution in some programs, execute before rune time execution. Some programs do not need and others need as instance for replacement done in compile time execution in #define which is the preprocessor directive substitute's statement in C++ programming, like in the program below:

4.1 PROGRAM 1

```
#INCLUDE <IOSTREAM.H>
#DEFINE X1 B + C
#DEFINE X2 X1 + X1
#DEFINE X3 X2 * C + X1 – D
#DEFINE X4 2 * X1 + 3 * X2 + 4 * X3
MAIN ()
{INT B = 2; // DECLARES AND INITIALIZES FOUR VARIABLES.
INT C = 3;
INT D = 4;
INT E = X4;
// PRINTS THE VALUES.
COUT << E << “, “ << X1 << “, “ << X2;
COUT << “, “ << X3 << “, “ << X4 << “\N”;
RETURN 0;
```

In the above program there are four replacements for preprocessor directives. It is executed before C++ compiling. Sometimes execution of preprocessor directive (#define) will cause somewhat slowing before execute C++ script in extra nanoseconds because it is compile to source code.

4.2 Script 2

```
int factorial (int n) {if (n == 0)
return 1;
return n * factorial(n - 1);}
// computed at compile time
const int y = factorial(0); // == 1
const int x = factorial(4); // == 24
```

In script2 of factorial function there is a comment before if (n == 0)
return 1; return n * factorial (n - 1);}.

Also this statement compiled at compile time execution that could delay execution of whole program some millisecond or some nanoseconds. Also this statement compiled at compile time execution that could delay execution of whole program some millisecond or some nanoseconds.

5 DISCUSSION

David B. Stewart, 2006, in his overview of measurement techniques of execution time explains that there are many different methods exist to measure execution time, but there is no single best technique. He mentioned that each technique is a compromise between multiple attributes, like resolution, accuracy and granularity [5]. The resolution first which is a representation of the some limitations of the timing that concern hardware. The example as Stewart a stop watch measures with a 0.01 sec resolution, while a logic analyzer may be measured with a resolution of 50 nanoseconds. That it which conduct us for different nanoseconds of some programs. Off course thereupon the algorithm (short or long way) that could execute the programs and its statements. Then it come the accuracy which is give method of measuring, as compared to the actual time if a perfect measurement that obtained through execution of the program. If a particular measurement is repeated several times, there is usually some amount of error in the measurements that is explains also as findings of this research that for instance loop statement for or while condition. The last technique is the granularity, it is the part of the compiling code that can be measured and it is specified in which David B Stewart called it a subjective manner.

6 CONCLUSION

C++ program execution different from program to another based on the statements, commands and compiling time taking and the statements that execute at compile time as well as running time. According to Professor Roberto Cipolla, that there are three phases of program go through development are editing, compiling and running the program and checking for logical errors (could call it debugging). There are short ways and the algorithm of the program may reveal through programming and development to get short time results with the same precision could be in the same programs aim. Millisecond or nanoseconds different in the speed could be between two programs with non-same statements. Loop (for) statements or big condition while statement as example are could also take more milliseconds than usual a + b +c or a*b/c direct, which take a little nanoseconds of execution speed. Compile time in development which is compiling to source code take a little time before compiling of C++ program. Some commands in C++ programming like (#define) and (#include) preprocessor in the beginning of C++ programming body, is must compile before C++ program compile or rune time.

7 ACKNOWLEDGMENT

Thank to everyone help me and was helpful to finish this simple article. I hope that this article is very good beginning for me to go through scientific work. I hope to do more researches in IT and CS scope.

ISSN 2347-4289

8 REFERENCES

[1]. Ugur Halıcı, operating Systems, processing scheduling, 2007. Website: eee.metu.edu.tr/~halici/courses/442/Ch2%20Process%20Scheduling.pdf

[2]. Roberto Cipolla, Editing, Compiling and Executing a Simple Program, 2004. Website: eng.cam.ac.uk/help/languages/C++/C++_tutorial/editing.html

[3]. Prashant, Compiling and Linking in C++, 2013. cplusplus.com/articles/2v07M4Gy/

[4]. Howard Hinnant, Beman Dawes, Vicente J. Botet Escriba, Boost C++ libraries,2008-2009. boost.org/doc/libs/1_47_0/doc/html/chrono/users_guide.html

[5]. David B. Stewart, Measuring Execution Time and Real-time Performance, 2006. drdobbs.com/embedded-systems/measuring-execution-time-and-real-time-p/193502123