International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 5 Issue: 7
ISSN: 2454-4248
01 – 02

Evaluation of C Library Function rand() and the Associated Compilers Available Off the Shelf for Windows 10 and Kubuntu 19.04

Abhinandan H. Patil
14 Inc,
Belgaum, Karnataka, India
Abhinandan_patil_1414@yahoo.com

Sangeeta A. Patil
14 Inc,
Belgaum, Karnataka, India
Sangeetapatil1981@gmail.com

***Abstract*—** This paper documents the observations made with respect to library function rand() on Windows 10 and Kubuntu 19.04 platform with various compilers such as TDM-GCC 4.9.2 64 bit for Windows 10, clang for Windows 10,Microsoft Visual Studio VC++ compiler for Windows 10 and gcc for Kubuntu 19.04 for a very simple C program.

The observations were with respect to uniqueness of the generated random numbers and execution speed of the whole program.

Keywords- *mingw; gcc; g++; MS-VC++; Clang; Windows 10; Kubuntu19.04*

*****

I. INTRODUCTION

Following program is essential component for evaluation of the criteria mentioned earlier. That is,

- How unique the numbers generated are and
- How fast the computer did the task at hand.

========== Start of the code snippet =========

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/timeb.h>

#define NO_OF_RANDOM_NOS 100000


int randCustom (int seed);
  /* Variables for the program */
int randomNumberArray[NO_OF_RANDOM_NOS], iter, iter1, dupValue = 0, dupValueBuiltIn = 0;
int builtInRandomNumberArray[NO_OF_RANDOM_NOS], dupFlag[NO_OF_RANDOM_NOS], dupBuiltInFlag[NO_OF_RANDOM_NOS];
long customStart, customEnd, builtInStart, builtInEnd;
//structCustomStart, structCustomEnd, structBuiltInStart, structBuiltInEnd;
int main()
{
    clock_t start, end;
    double cpu_time_used;

    start = clock();
    // srand (time (NULL));
  for (iter = 0; iter < NO_OF_RANDOM_NOS; iter++)
   {
      randomNumberArray[iter]=0;
      builtInRandomNumberArray[iter]=0;
      dupFlag[iter]=0;
      dupBuiltInFlag[iter]=0;
    }
    //customStart = times (&structCustomStart);
    for (iter = 0; iter < NO_OF_RANDOM_NOS; iter++)
     {
      randomNumberArray[iter] = randCustom (NO_OF_RANDOM_NOS*iter + 1);
       printf(": %d \n",randomNumberArray[iter]);
       /* randomNumberArray = randCustom (iter); */
     }
    //customEnd = times (&structCustomEnd);
    //printf ("\n Custom number generation time = %ld", customEnd - customStart);
    //builtInStart = times (&structBuiltInStart);

    for (iter = 0; iter < NO_OF_RANDOM_NOS; iter++)
     {
      builtInRandomNumberArray[iter] = rand ();
      printf(": %d \n",builtInRandomNumberArray[iter]);
     }
    //builtInEnd = times (&structBuiltInEnd);
    //printf ("\n built in number generation time = %ld", builtInEnd - builtInStart);
    for (iter = 0; iter < NO_OF_RANDOM_NOS; iter++)
     {
      for (iter1 = 0; iter1 < NO_OF_RANDOM_NOS; iter1++)
        {
         if (iter == iter1)
           continue;
         if (randomNumberArray[iter] == randomNumberArray[iter1])
           {
            dupFlag[iter] = 1;
           }
         if (builtInRandomNumberArray[iter] == builtInRandomNumberArray[iter1])
           {
            dupBuiltInFlag[iter] = 1;
           }
        }
     }
    dupValue = 0;
    dupValueBuiltIn = 0;
```

 ISSN: 2454-4248
01 – 02

```
    for (iter = 0; iter < NO_OF_RANDOM_NOS; iter++)
      {
       if (dupFlag[iter] == 1)
         dupValue++;
       if (dupBuiltInFlag[iter] == 1)
         dupValueBuiltIn++;
      }
    printf ("The number of duplicates in the custom array %d\n", dupValue);
    printf ("The number of duplicates in the built in array %d\n",
    dupValueBuiltIn);
    end = clock();
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf ("CPU TIME USED  %f", cpu_time_used);
    return 0;
   }

   int
   randCustom (int seed)
   {
    static long temp = 100001;
    if (seed == 0)
      return 1;
    temp = (temp * 125) % 2796203;
    return ((temp % seed) + 1);
   }
==================End of Code snippet ===========
```

II. Hardware used for the experiment

Intel Core i3 CPU 2.4 GHz with 4 GB RAM. Further details on request/demand.

III. Observations

Following observations were made.

	Windows 10, Dev C++ 5.11 with TDM-GCC 4.9.2 64 bit release		
Run 1	Non unique numbers 2	Non unique numbers 95238	91.78 Seconds
Run 2	Non unique numbers 2	Non unique numbers 95238	128.3 Seconds

	Windows 10, Visual Studio with VC++ Compiler		
Run 1	Non unique numbers 2	Non unique numbers 95238	59.97 Seconds
Run 2	Non unique numbers 2	Non unique numbers 95238	57.083 Seconds

	Windows 10, clang with bare metal		
Run 1	Non unique numbers 2	Non unique numbers 95238	101.47 Seconds
Run 2	Non unique numbers 2	Non unique numbers 95238	97.91 Seconds

	Kubuntu 19.04 with gcc 8.3.0		
Run 1	Non unique numbers 2	Non unique numbers 6	62.49 Seconds
Run 2	Non unique numbers 2	Non unique numbers 6	62.423 Seconds

	Kubuntu 19.04 with clang 8.0.0-3		
Run 1	Non unique numbers 2	Non unique numbers 6	64.77 Seconds
Run 2	Non unique numbers 2	Non unique numbers 6	64.76 Seconds

IV. Conclusion

- rand() is truly random on {Kubuntu 19.04, gcc/clang} platform
- Performance of {Kubuntu, gcc} was predictable and far superior than {gcc, windows 10}
- However, {Windows 10, VC++} was comparable with {Kubuntu 19.04,gcc}. Infact, {Windows 10, VC++} was “marginally” better than {Kubuntu, gcc} by 3 seconds.
- Custom random function was predictable on all platforms with respect to randomness as the number of non-unique instances were only 2 for 100,000 instances.

Acknowledgment

We would like to thank open source software and free software used in this study.

References

[1] https://sourceforge.net/p/dev-cpp/discussion/48211/thread/3b15b75b/
[2] https://www.geeksforgeeks.org/
[3] https://docs.microsoft.com/en-in/
[4] https://kubuntu.org