



Linux | DB | Open Source | Web



≡ Menu

- [Home](#)
- [Free eBook](#)
- [Start Here](#)
- [Contact](#)
- [About](#)

10 Tips for C and C++ Performance Improvement Code Optimization

by Koscica Dusko on January 15, 2015



When you start writing your code in C, C++ or any other programming language, your first objective might be to write a program that works.

After you accomplished that, the following are few things you should consider to enhance your program.

1. Security of the program
2. Memory consumption
3. Speed of the program (Performance Improvement)

This article will give some high-level ideas on how to improve the speed of your program.

Few general points to keep in mind:

- You could optimize your code for performance using all possible techniques, but this might generate a bigger file with bigger memory footprint.
- You might have two different optimization goals, that might sometimes conflict with each other. For example, to optimize the code for performance might conflict with optimize the code for less memory footprint and size. You might have to find a balance.
- Performance optimization is a never-ending process. Your code might never be fully optimized. There is always more room for improvement to make your code run faster.
- Sometime we can use certain programming tricks to make a code run faster at the expense of not following best practices such as coding standards, etc. Try to avoid implementing cheap tricks to make your code run faster.

1. Optimize your Code using Appropriate Algorithm

For any code you write, you should always take some time to think through and pick the right algorithm to use for your specific scenario.

The problem we are going to analyze for this example is to find a maximum value of the function in a two dimensional segment.

We'll consider only whole numbers.



First we'll write the program without consider performance. Then, we'll discuss few ways to boost the performance of this program.

Our Scenario: We have interval for x $[-100...100]$ and interval for y $[-100...100]$. Now in these two intervals we are looking for a maximum of the function $(x*x + y*y)/(y*y + b)$.

This is a function of two variables: x and y . There is one more constant which could be different and user will enter it. This constant b is always greater than 0 and also lesser than 1000.

In our program ,we will not use function pow() that is implemented in math.h library. It would be interesting exercise to figure out which approach would create faster code.

Example code:

```
#include <iostream>

#define LEFT_MARGINE_FOR_X -100.0
#define RIGHT_MARGINE_FOR_X 100.0
#define LEFT_MARGINE_FOR_Y -100.0
#define RIGHT_MARGINE_FOR_Y 100.0

using namespace std;

int
main(void)
{
    //Get the constant value
    cout<<"Enter the constant value b>0"<<endl;
    cout<<"b->"; double dB; cin>>dB;

    if(dB<=0)    return EXIT_FAILURE;
    if(dB>1000) return EXIT_FAILURE;

    //This is the potential maximum value of the function
    //and all other values could be bigger or smaller
    double dMaximumValue = (LEFT_MARGINE_FOR_X*LEFT_MARGINE_FOR_X+LEFT_MARGINE_FOR_Y*LEFT_MARGINE_FOR_Y)/ (LEFT_MARGINE_FOR_Y*LEFT_MARGINE_FOR_Y+dB);

    double dMaximumX = LEFT_MARGINE_FOR_X;
    double dMaximumY = LEFT_MARGINE_FOR_Y;

    for(double dX=LEFT_MARGINE_FOR_X; dX<=RIGHT_MARGINE_FOR_X; dX+=1.0)
        for(double dY=LEFT_MARGINE_FOR_Y; dY<=RIGHT_MARGINE_FOR_Y; dY+=1.0)
            if( dMaximumValue<((dX*dX+dY*dY)/(dY*dY+dB)))
            {
                dMaximumValue=((dX*dX+dY*dY)/(dY*dY+dB));
                dMaximumX=dX;
                dMaximumY=dY;
            }

    cout<<"Maximum value of the function is="<< dMaximumValue<<endl;
    cout<<endl<<endl;
    cout<<"Value for x="<<dMaximumX<<endl
        <<"Value for y="<<dMaximumY<<endl;

    return EXIT_SUCCESS;
}
```

Now, if we analyze the code more carefully, we notice that the part for $dX \cdot dX$ is calculated more times than it should, in this case it is calculated 200 times and this is a waste of CPU time. What we could do?

One of the tricks is to create one variable $dX_Square = dX \cdot dX$, and calculate it after first for repetition, then we could use that in all calculations afterwards. You just need to add one more bracket.

There are few more optimizations you can do in the above code, just try to spot them.

The next point we could consider is how general our algorithm is, versus how optimal is it from speed point of view.

In that case we could apply few algorithms depending on the size of input set. What do we mean by that?

For example, in one of our earlier C++ articles, we discussed about [binary numbers](#) that have only two ones in many zeros.

We could use MVA algorithm that could outperform the original algorithm from speed point of view on smaller numbers, the ones that are fit for unsigned long long int, but if you use my algorithm combined with vectors it could be used in some problems where you try to pick two objects that are in set.

So, in order to create the best possible solution you could merge two algorithms and apply one according to the size of the problem. So, if the number used is smaller than unsigned long long int, you could use first algorithm and if number will not fit already mentioned type of data you could use vectors or some other data structures.

Similar to this would be addition of numbers, where it is simple to consider case of long long int, but in case we need to add to big numbers that are in size way bigger than unsigned long long int you could use vectors to store them and apply operation of addition with your algorithm. If you prefer classes you could use them to, but if you don't need OOP approach you could just use double linked list or arrays or some other more appropriate data structure.

2. Optimize Your Code for Memory

Now we will look how you could optimize your code from point of memory consumption.

Let us take a simple example. Let us try to swap two values in the memory, which is done in many sorting algorithms.

Some people like to think of this as two people sitting on two chairs and adding one more chair as temporary holder for one of them during the swap.

```
int nFirstOne =1, nSecondOne=2;
int nTemp = nFirstOne;
nFirstOne = nSecondOne;
nSecondOne = nTemp;
```

This is nice. But usage of nTemp which in memory reserves some place that will be used for copy of one variable.

This could be done without nTemp like this:

```
int nFirstOne = 3, nSecondOne = 7;  
nFirstOne += nSecondOne;  
nSecondOne = nFirstOne ? nSecondOne;  
nFirstOne -= nSecondOne;
```

In some cases you could have large objects in memory that need to swap their places. So, what could you do? Instead of coping to many memory locations you could use their addresses and instead of replacing all of the memory locations you could just change their address.

How do you know if your code is faster and how do you calculate it?

Well, when you finish your code it will translate in some assembler and then into something that is called machine code. Each operation is performed in processor or in some other part of computer like mathematical coprocessor or in graphic card or something similar.

One operation could be done in one clock circle or few, this is the reason why it could be faster to multiply than to divide, but it could be also important did you pick some optimization done by your compiler.

Sometimes, the task of optimization could be left to compiler. For all available C++ compiler, check this [GCC Optimization Options](#).

To understand how fast program is, you should know the architecture of a device you are working with. Sometimes things become faster because your program is in the cache memory or you use mathematical coprocessor or because branch predictor got it right most of the times.

Now let's consider these numbers $O(n)$, $O(\log(n) * n)$, $n*n$, $n!$. To estimate the algorithm according to the size of the input set you use this numbers.

If you have an algorithm of size n and you input 10 elements you get time t , and if you input 100 elements you will end up with time 10 times longer than t . If you deal with a program that has equivalent to $n*n$ and you increase the size of set from 10 to 100, the program will not be 10 times slower but rather approximately $10*10$ times. You should be aware of these types of limits a number can have on your algorithm.

Some people think that they could time the code and have good idea how fast the algorithm is. Ok, let us think. Most of the programs you write are not in kernel mode, which means that they could be stopped by operating system and processor could be given to another task and so on. This means that your program will be stopped and started many times. It could be even tougher to figure out what could happen to program if you have few cores or even processors.

Idea of measuring the speed of algorithm is pretty iffy. Well, the results are just useful as a snowflake on a North Pole or like hand of sand in desert.

The only good results are if you find a way to prevent your program from losing the core he is in, or perhaps to stop the counter of time and then continue, but you need to eliminate interrupt time that will be added each time you stop your program, as well as the starting initializations.

There are also differences you will notice due to the fact that same code will not be transformed into machine code if you apply different optimization, and as you should know that already one product could translate code in different way than some other version, by the way it is important what architecture is it executed as well and also due to installed amount of memory, cache memory, prediction methods, etc.

3. printf and scanf Vs cout and cin

Sometimes, if you use different functions for same task you will get faster code.

Those first two functions are mostly used in C style of programming, but you could use it sometimes with file manipulation and small difference in speed could add up a lot saved time.

For example, let us assume that you have numbers in a file to read.

From point of security, cout and cin would be considered as better option for files, as you would have adequate instructions in fstream header.

If you use C or printf in C++ you should consider some other functions that could even more increase the speed of your program.

For strings you could use puts, gets or their equivalents for file operations. Well they are not formatted and to write data in one way takes some time.

4. Using Operators

Most basic operations like +=, -=, and *=, when applied on basic data types could slow down the program as well. To be sure you will need to know how it gets transformed into assembler on your computer.

One interesting idea is to replace postfix increment and decrement with their prefix versions.

Sometimes you could try to use operators >> or << instead of multiplication or division, but be very careful, you could end up with bad mistake this way, and then to fix it you could add some range estimations and that will be way slower than original code you have started with.

Bit operators, and tricks that go with them could increase the speed of program, but you should be very careful because you could end up with machine dependant code and that is something to avoid. To be sure, you could still code with add move from assembler in C++.

It is important to understand that this is hybrid language and it will support assembler coding, problem orientated solutions, the object orientated solutions, and if you add some additional libraries you could use some more advanced tricks that are not commonly used.

5. if Condition Optimization

If you use if in your code, when possible, it is a good idea to replace if with switch. In “if”, you usually have tests and that could produce code that is bit slower.

One good fact to know about if command is to know that it has some of the optimizations built in. Well, if you have few conditions that are connected with && or || it could be evaluated that this is true or false without calculating complete expression.

Let us illustrate this with two condition that are connected with && operator. If you have expression p and q, as soon as you have p equal to false you know that there is no way to get true as a result, this is used in C/C++ and sometimes it could be reason why people do get wrong code.

If you have situations in which you could say that something could occur more often put it before, because there is a better chance to say that expression is false or true. If you have many conditions to calculate, and if they could be sorted, consider splitting that range into few sub ranges first.

One bad thing that could happen is that you create the branch that will never be used or even few lines of the code that could be added and you will never use those cases.

Sometimes you will have a very long expression composed of many conditions, one could use function that will return true or false, but functions are expensive, they use stack and few copies could be created, if possible you could use a macro or a macro with a variable to increase the speed and create code that will be more easy to maintain.

Also, don't forget that negation is an operation too.

6. Problems with Functions

While using functions, if you are not careful, you might end-up creating a bad code.

For example, if you have a code like this, it might be a bad thing.

```
for(int i=1; i<=10; ++i)
    DoSomething(i);
```

Why? As soon as you code something like this you will have to call DoSomething 10 times, and we have mentioned that function calls could be expensive.

To implement this better, you could do it like this, and implement that for repetition in your function.

```
DoSomething(n);
```

Next thing to consider is inline functions. There is a chance that they will be used like macros if they are small. This way you benefit from point of speed, from point of better organization, and as well as reusability.

When passing a big object to a function, you could use pointers or references. Prefer to use references because they would create the code that is way easier to read.

If you are not worried about the changing the value that is passed to the function, use references. If you use object that is constant, it could be useful to use const, which will save some time.

When you use C that will support C99 standard you have option to use restrict on pointers to.

In certain situations casting in function might increase the speed of the code. You should consider this depending on your specific situation.

Creating temporary objects in the function could slow the program. I have already shown how you could avoid using temp variable in some situations.

Also, while recursion is extremely helpful in certain specific scenarios, in general, it will generate a slow performing code. If possible, try to avoid recursion, when you don't need to use it to solve your problem.

7. Optimizing Loops

If you like to check if a number is lower than 10 or greater than zero, pick the second option.

It is faster to test if something is equal to zero than to compare two different numbers.

In other words, the following is slower when compared to the alternative option shown below:

```
for( i =0; i<10; i++)
```

The following is faster when compared to the above for loop. But, this might be harder to read for beginners.

```
for(i=10; i--; )
```

Similar to this is case if you are in a situation where you could pick form !=0 and <=n, use the first one it will be faster. For example, when you try to calculate factorial in the separate function.

It is better to avoid loop in situations where you have few functions called with different arguments that are ranging from 1 to 5, it is better to use linear calls with five calls.

If you are in the situation to use: one loop and few tasks or few loops with one task in each loop. Pick the first option. It is a trick that could generate faster code. I am not sure, but compiler probably could not optimize this still.

8. Data Structure Optimization

Is the data structure that we use affect the performance of the code?

The answer this this question is not simple, that you could expect from simple math. It is rather vague and hard to formulate.

To illustrate my statement, we will analyze one example. If your task is to create permutations that are like the following, then you could use array or linked list.

```
1, 2, 3, 4,  
2, 3, 4, 1,  
3, 4, 1, 2,  
4, 1, 2, 3,
```

If you use array, you could copy first element and move all others toward first element and then move the first element at the last place. This would create so many not needed operations that your program or a function would be very slow.

If you keep your data in the list, you could very easy create the program that will outperform one with array we have mentioned.

Sometimes, if you save your data in some form of tree you could create program that will perform faster than the one without adequate data structure.

Be careful when using data structure. Sometimes a problem could be solved without keeping all elements of array or using any data structure at all.

To elaborate on this subject, refer to the discussion we had on the [Fibonacci algorithm](#). If you look at the Fibonacci elements you could be tricked in applying the vector in combination with recursion, but instead you could use some trick from applied math to create very fast code.

9. Binary Search or Sequential Search

Should we use binary search or sequential search to solve a problem?

One of the common tasks we need to do when we program is to search for some value in some data structure. Yes, it is basis for a hash tables, multi-level hash tables, etc.

If you are trying to find one number in an array of numbers you could have two strategies.

The first strategy is very simple. You have your array and value you are looking for. From beginning of the array you start to look for the value and if you find it you stop the search, and if you don't find the value you will be at the end of the array. There are many improvements to this strategy.

The second strategy requires the array to be sorted. If array is not sorted you will not get the results that you wish for. If the array is sorted you split it in two half. In the first half, the elements of array are smaller than the middle one in another half the elements are bigger than the middle one. If you get yourself in situation that two markers are not situated the way they should you know that you don't have the value you have been looking for.

What is the dilemma here? If you sort elements of array you will lose some time, but if you invest in that you could benefit from faster binary search.

This is one of situations where you would need to understand the problem well and act according to best possible situation based on your specific scenario.

10. Optimizing Arrays

The array is one of the most basic data structures that occupy some space in memory for its elements.

To understand how these optimizations work, you should be aware of the arrays structure. Ok, what do I mean by this. The name of array is a constant pointer that points at first element of an array. This means that you could use pointers and pointer arithmetic.

If you access members of array like this:

```
for(int i=0; i<n; i++) nArray[i]=nSomeValue;
```

Instead of the above code, the following is better:

```
for(int* ptrInt = nArray; ptrInt< nArray+n; ptrInt++) *ptrInt=nSomeValue;
```

The reason for this is in the operations with pointers. In the above example, we have pointer to int data type that takes address from the name of the array. In this case, it is nArray, and we increase that address for one element, and the pointer is moved toward the end of the array for size of int data type.

If you have used double, your compiler would know how far it should move the address.

It is way harder to read code this way, but it will increase the speed of program. In other words, when you don't use better algorithm, but your program still runs faster, the increased speed might be due to better syntax that will generate faster code.



If you use matrix, and you have chance to approach the elements of matrix row by row or in some other manner you should always pick to go row after the row in your matrix. The matrix is an array of arrays it will be stored in memory row after the row, so the most natural way to do approach the array members is to go row by row.

Avoid initialization of large portions of memory with some element. If you could not avoid this type of situation, consider memset and similar commands.

When you use array of chars, these are sometimes called strings in style of language C, you could create faster code if you use pointers trick as well. If you use string as instance from C++ class, you might feel more comfortable, but you could create slower code and sometime even the a bigger file size.

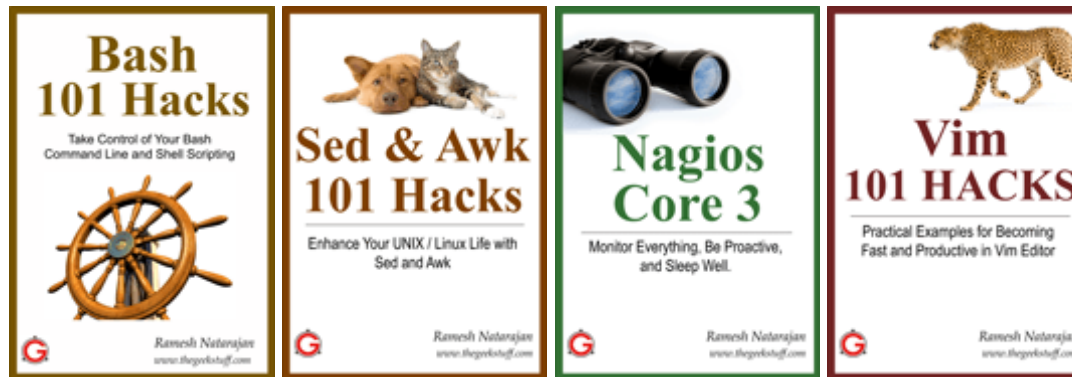
If you use new to create your array, your program could perform badly because you use a lot of growth in memory at one moment, which is the reason why you should use vectors. They will add some space in memory and grow in better way.

If you try to move large set of data in memory, you could use array of pointers. This way, you will not move the real data in memory, but replace the addresses instead.

 38  38  23 [Add your comment](#)

If you enjoyed this article, you might also like..

1. [50 Linux Sysadmin Tutorials](#)
 2. [50 Most Frequently Used Linux Commands \(With Examples\)](#)
 3. [Top 25 Best Linux Performance Monitoring and Debugging Tools](#)
 4. [Mommy, I found it! – 15 Practical Linux Find Command Examples](#)
 5. [Linux 101 Hacks 2nd Edition eBook](#) **Free**
- [Awk Introduction – 7 Awk Print Examples](#)
 - [Advanced Sed Substitution Examples](#)
 - [8 Essential Vim Editor Navigation Fundamentals](#)
 - [25 Most Frequently Used Linux IPTables Rules Examples](#)
 - [Turbocharge PuTTY with 12 Powerful Add-Ons](#)



{ 15 comments... [add one](#) }

- santosh ydv January 16, 2015, 2:08 am

using inline functions cautiously can improve performance.

[Link](#)

- santosh ydv January 16, 2015, 2:10 am

declare constants value holding variables as constants, so that compiler can optimize accessing those values,

[Link](#)

- duskoKoscica January 16, 2015, 1:31 pm

[This](#) one is promising.

[Link](#)

- Erlo Haugen January 16, 2015, 2:44 pm

In my opinion, optimisation is best left to the compiler. Picking the optimal algorithm and implementing it wisely is of course up to the programmer, but 'microoptimization', e.g Your for-loop example and array access using pointers is best left to the compiler. In the case of array access, the (good) compiler will use pointers anyway.

One useful 'hand optimization' however is to use the appropriate data type for the job at hand. Don't use float or dpuble if an integer type will do, use unsigned insted of integer if possible.

Just my two cents.

[Link](#)

- duskoKoscica January 17, 2015, 4:38 am

@santosh ydv

is this some improvement for encryption thing, I mean ydv, or U just hate: a,e, i, o, u ?

Nice comment it is useful in some situations. THX!

@Erlo Haugen it would be best to create computer that wold make programs. So, there would not be disscusions about this. Some compilers would be able to use one thing and some not, it might be problem for people that don't understand the pointer syntax. It is worth investing into it.

Appropriate data type for pro is bid different than this data types used like, but it is not an issue to use them, however I would like to have them changed even in C++ standard, ...

Stingy from you

I have not covered all tricks, I hope people would contribute with some of they own, that would be point of C++ and open source, wouldn't it?

[Link](#)

- duskoKoscica January 17, 2015, 6:04 am

‘And nice picture to go with it>

http://osborne-friends-hannover.de/Stiere/toro_06.jpg-for-web-LARGE.jpg

[Link](#)

- duskoKoscica January 21, 2015, 9:46 am

Ok!

This one is for swap of addresses>

```
int* ptrTemp= i;  
i = j;  
j = ptrTemp;
```

This way you only exchange the addresses. It is basis for exchange of pointers that are in an array.

[Link](#)

- duskoKoscicaca January 23, 2015, 11:07 am

There are few more C++ optimization tutorials: 1. [here](#) 2. [here](#) 3. [here](#) 4. [here](#) 5. [here](#)

Yeah, like after party!

[Link](#)

- Kiran February 2, 2015, 8:22 am

```
>> nSecondOne = nFirstOne ? nSecondOne;
```

This line should be

```
nSecondOne = nFirstOne - nSecondOne;
```

[Link](#)

- duskoKoscica February 3, 2015, 3:58 am

One thing that could be influenced by operating system.

If you have connection to the Internet you would need more security.

If you are not connected to the internet you would not need complete safety thing.

One idea to consider if it is not implemented already.

[Link](#)

- dusKO*2scica February 9, 2015, 12:15 pm

Yeah!

One thing I am sure about do. I will never write the article that will try to persue the programmer to trust their compiler any way...

It is just not my kind of compiler, we have open source we could trust to!

Obvious difference, for real!

[Link](#)

- duskoKoscica February 11, 2015, 2:24 am

HI!

One thing I know, I will never write the article that has purpose to convince people to use optimizations by their compiler.

If you don't believe you could check it and you could double check with source code as well.

Have nice day!!!

[Link](#)

- duskoKoscica March 8, 2015, 6:39 am

Yes this is ok, as well

http://en.wikipedia.org/wiki/International_Women%27s_Day

to all woman that celebrate that!

[Link](#)

- Avinish June 15, 2015, 5:41 am

how can i optimize this code snippet

```
result = 1
for (int32_t x = 1, pxcount = 0; primes[x] <= n; ++x, pxcount = 0) {
    for (int32_t y = primes[x]; y <= n; y *= primes[x])
        pxcount += n / y;
    result = (1ll * result * (pxcount + 1)) % m;
}
```

Here,

I am trying to find the total no of divisors of n! but a sqrt(n) approach is possible i think which i am not able to find out

[Link](#)

- DuskoKoscica July 5, 2015, 6:15 am

@Kiran nice job!

@Avinish, yes sqrt(n) works, however in your case I would recommend you to check some results from number theory, it would generate way faster code.

For this one, I am not 100% sure.

Have nice time!

[Link](#)

Leave a Comment

Name

Email

Website

Comment

Submit

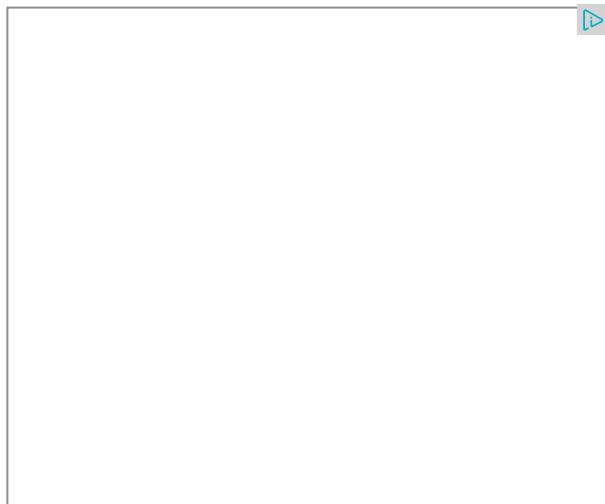
☐ Notify me of followup comments via e-mail

Next post: [Registration Now Open: 2-Day Linux Sysadmin Workshop in Los Angeles](#)

Previous post: [How to Fix DELL PowerEdge W1228 Raid Controller Battery Capacity <24hr Error Message](#)

[RSS](#) | [Email](#) | [Twitter](#) | [Facebook](#) | [Google+](#)

Search



EBOOKS

- **Free** [Linux 101 Hacks 2nd Edition eBook](#) - Practical Examples to Build a Strong Foundation in Linux
- [Bash 101 Hacks eBook](#) - Take Control of Your Bash Command Line and Shell Scripting

- [Sed and Awk 101 Hacks eBook](#) - Enhance Your UNIX / Linux Life with Sed and Awk
- [Vim 101 Hacks eBook](#) - Practical Examples for Becoming Fast and Productive in Vim Editor
- [Nagios Core 3 eBook](#) - Monitor Everything, Be Proactive, and Sleep Well



The Geek Stuff

12 758 Me gusta

Me gusta esta página

Compartir

Sé el primero de tus amigos en indicar que le gusta esto.



POPULAR POSTS

- [12 Amazing and Essential Linux Books To Enrich Your Brain and Library](#)
- [50 UNIX / Linux Sysadmin Tutorials](#)
- [50 Most Frequently Used UNIX / Linux Commands \(With Examples\)](#)
- [How To Be Productive and Get Things Done Using GTD](#)
- [30 Things To Do When you are Bored and have a Computer](#)
- [Linux Directory Structure \(File System Structure\) Explained with Examples](#)
- [Linux Crontab: 15 Awesome Cron Job Examples](#)
- [Get a Grip on the Grep! – 15 Practical Grep Command Examples](#)
- [Unix LS Command: 15 Practical Examples](#)
- [15 Examples To Master Linux Command Line History](#)
- [Top 10 Open Source Bug Tracking System](#)
- [Vi and Vim Macro Tutorial: How To Record and Play](#)
- [Mommy, I found it! -- 15 Practical Linux Find Command Examples](#)
- [15 Awesome Gmail Tips and Tricks](#)
- [15 Awesome Google Search Tips and Tricks](#)
- [RAID 0, RAID 1, RAID 5, RAID 10 Explained with Diagrams](#)
- [Can You Top This? 15 Practical Linux Top Command Examples](#)
- [Top 5 Best System Monitoring Tools](#)

- [Top 5 Best Linux OS Distributions](#)
- [How To Monitor Remote Linux Host using Nagios 3.0](#)
- [Awk Introduction Tutorial – 7 Awk Print Examples](#)
- [How to Backup Linux? 15 rsync Command Examples](#)
- [The Ultimate Wget Download Guide With 15 Awesome Examples](#)
- [Top 5 Best Linux Text Editors](#)
- [Packet Analyzer: 15 TCPDUMP Command Examples](#)
- [The Ultimate Bash Array Tutorial with 15 Examples](#)
- [3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id](#)
- [Unix Sed Tutorial: Advanced Sed Substitution Examples](#)
- [UNIX / Linux: 10 Netstat Command Examples](#)
- [The Ultimate Guide for Creating Strong Passwords](#)
- [6 Steps to Secure Your Home Wireless Network](#)
- [Turbocharge PuTTY with 12 Powerful Add-Ons](#)

CATEGORIES

- [Linux Tutorials](#)
- [Vim Editor](#)
- [Sed Scripting](#)
- [Awk Scripting](#)
- [Bash Shell Scripting](#)
- [Nagios Monitoring](#)
- [OpenSSH](#)
- [IPTables Firewall](#)
- [Apache Web Server](#)
- [MySQL Database](#)
- [Perl Programming](#)
- [Google Tutorials](#)
- [Ubuntu Tutorials](#)
- [PostgreSQL DB](#)
- [Hello World Examples](#)
- [C Programming](#)
- [C++ Programming](#)
- [DELL Server Tutorials](#)
- [Oracle Database](#)
- [VMware Tutorials](#)

Ramesh Natarajan



About The Geek Stuff



My name is **Ramesh Natarajan**. I will be posting instruction guides, how-to, troubleshooting tips and tricks on Linux, database, hardware, security and web. My focus is to write articles that will either teach you or help you resolve a problem. Read more about [Ramesh Natarajan](#) and the blog.

Contact Us

Email Me : Use this [Contact Form](#) to get in touch me with your comments, questions or suggestions about this site. You can also simply drop me a line to say hello!.

[Follow us on Google+](#)

[Follow us on Twitter](#)

[Become a fan on Facebook](#)

Support Us

Support this blog by purchasing one of my ebooks.

[Bash 101 Hacks eBook](#)

[Sed and Awk 101 Hacks eBook](#)

[Vim 101 Hacks eBook](#)

[Nagios Core 3 eBook](#)

Copyright © 2008–2015 Ramesh Natarajan. All rights reserved | [Terms of Service](#)