C Pointer to Pointer, Pointer to Functions, Array of Pointers Explained with Examples

by Himanshu Arora on January 27, 2012

In C programming language, the concept of pointers is the most powerful concept that makes C stand apart from other programming languages. In the part-I of this series we discussed the fundamental concepts around C pointers.

In this article, we will try to develop understanding of some of the relatively complex concepts. The following are explained in this article with examples:

1. Constant pointer and pointer to constant.
2. Pointer to pointer with an example
3. Array of pointers with an example
4. Pointer to functions with an example

1. C Constant Pointer and Pointer to Constant

As a developer, you should understand the difference between constant pointer and pointer to constant.

C Constant pointer

A pointer is said to be constant pointer when the address its pointing to cannot be changed.

Lets take an example :

```
char ch, c;
char *ptr = &ch
ptr = &c
```

In the above example we defined two characters ('ch' and 'c') and a character pointer 'ptr'. First, the pointer 'ptr' contained the address of 'ch' and in the next line it contained the address of 'c'. In other words, we can say that Initially 'ptr' pointed to 'ch' and then it pointed to 'c'.

But in case of a constant pointer, once a pointer holds an address, it cannot change it. This means a constant pointer, if already pointing to an address, cannot point to a new address.

If we see the example above, then if 'ptr' would have been a constant pointer, then the third line would have not been valid.

A constant pointer is declared as :

```
<type-of-pointer> *const <name-of-pointer>
```

For example :

```
#include<stdio.h>

int main(void)
{
    char ch = 'c';
    char c = 'a';

    char *const ptr = &ch; // A constant pointer
    ptr = &c; // Trying to assign new address to a constant pointer. WRONG!!!!

    return 0;
}
```

When the code above is compiled, compiler gives the following error :

```
$ gcc -Wall constptr.c -o constptr
constptr.c: In function ‘main’:
constptr.c:9: error: assignment of read-only variable ‘ptr’
```

So we see that, as expected, compiler throws an error since we tried to change the address held by constant pointer.

Now, we should be clear with this concept. Lets move on.

C Pointer to Constant

This concept is easy to understand as the name simplifies the concept. Yes, as the name itself suggests, this type of pointer cannot change the value at the address pointed by it.

Lets understand this through an example :

```
char ch = 'c';
char *ptr = &ch
*ptr = 'a';
```

In the above example, we used a character pointer ‘ptr’ that points to character ‘ch’. In the last line, we change the value at address pointer by ‘ptr’. But if this would have been a pointer to a constant, then the last line would have been invalid because a pointer to a constant cannot change the value at the address its pointing to.

A pointer to a constant is declared as :

```
const <type-of-pointer> *<name-of-pointer>;
```

For example :

```
#include<stdio.h>

int main(void)
{
    char ch = 'c';
    const char *ptr = &ch; // A constant pointer 'ptr' pointing to 'ch'
    *ptr = 'a';// WRONG!!! Cannot change the value at address pointed by 'ptr'.

    return 0;
}
```

When the above code was compiled, compiler gave the following error :

```
$ gcc -Wall ptr2const.c -o ptr2const
ptr2const.c: In function ‘main’:
ptr2const.c:7: error: assignment of read-only location ‘*ptr’
```

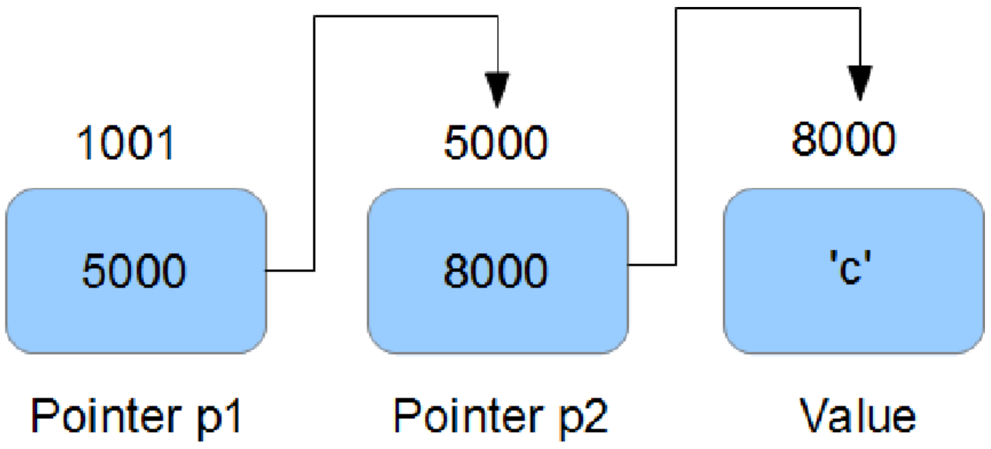
So now we know the reason behind the error above ie we cannot change the value pointed to by a constant pointer.

2. C Pointer to Pointer

Till now we have used or learned pointer to a data type like character, integer etc. But in this section we will learn about pointers pointing to pointers.

As the definition of pointer says that its a special variable that can store the address of an other variable. Then the other variable can very well be a pointer. This means that its perfectly legal for a pointer to be pointing to another pointer.

Lets suppose we have a pointer ‘p1′ that points to yet another pointer ‘p2′ that points to a character ‘ch’. In memory, the three variables can be visualized as :



So we can see that in memory, pointer p1 holds the address of pointer p2. Pointer p2 holds the address of character ‘ch’.

So ‘p2′ is pointer to character ‘ch’, while ‘p1′ is pointer to ‘p2′ or we can also say that ‘p2′ is a pointer to pointer to character ‘ch’.

Now, in code ‘p2′ can be declared as :

```
char *p2 = &ch;
```

But ‘p1′ is declared as :

```
char **p1 = &p2;
```

So we see that ‘p1′ is a double pointer (ie pointer to a pointer to a character) and hence the two *s in declaration.

Now,

- ‘p1′ is the address of ‘p2′ ie 5000
- ‘*p1′ is the value held by ‘p2′ ie 8000
- ‘**p1′ is the value at 8000 ie ‘c’

I think that should pretty much clear the concept, lets take a small example :

```
#include<stdio.h>

int main(void)
{
    char **ptr = NULL;

    char *p = NULL;

    char c = 'd';

    p = &c;
    ptr = &p;

    printf("\n c = [%c]\n",c);
    printf("\n *p = [%c]\n",*p);
    printf("\n **ptr = [%c]\n",**ptr);

    return 0;
}
```

Here is the output :

```
$ ./doubleptr
```

```
 c = [d]

 *p = [d]

 **ptr = [d]
```

3. C Array of Pointers

Just like array of integers or characters, there can be array of pointers too.

An array of pointers can be declared as :

```
<type> *<name>[<number-of-elements];
```

For example :

```
char *ptr[3];
```

The above line declares an array of three character pointers.

Lets take a working example :

```
#include<stdio.h>

int main(void)
{
    char *p1 = "Himanshu";
    char *p2 = "Arora";
    char *p3 = "India";

    char *arr[3];

    arr[0] = p1;
    arr[1] = p2;
    arr[2] = p3;

   printf("\n p1 = [%s] \n",p1);
   printf("\n p2 = [%s] \n",p2);
   printf("\n p3 = [%s] \n",p3);

   printf("\n arr[0] = [%s] \n",arr[0]);
   printf("\n arr[1] = [%s] \n",arr[1]);
   printf("\n arr[2] = [%s] \n",arr[2]);

   return 0;
}
```

In the above code, we took three pointers pointing to three strings. Then we declared an array that can contain three pointers. We assigned the pointers 'p1′, 'p2′ and 'p3′ to the 0,1 and 2 index of array. Let's see the output :

```
$ ./arrayofptr

 p1 = [Himanshu]

 p2 = [Arora]

 p3 = [India]

 arr[0] = [Himanshu]

 arr[1] = [Arora]

 arr[2] = [India]
```

So we see that array now holds the address of strings.

4. C Function Pointers

Just like pointer to characters, integers etc, we can have pointers to functions.

A function pointer can be declared as :

```
<return type of function> (*<name of pointer>) (type of function arguments)
```

For example :

```
int (*fptr)(int, int)
```

The above line declares a function pointer ‘fptr’ that can point to a function whose return type is ‘int’ and takes two integers as arguments.

Lets take a working example :

```
#include<stdio.h>

int func (int a, int b)
{
    printf("\n a = %d\n",a);
    printf("\n b = %d\n",b);

    return 0;
}

int main(void)
{
    int(*fptr)(int,int); // Function pointer

    fptr = func; // Assign address to function pointer

    func(2,3);
    fptr(2,3);

    return 0;
}
```

In the above example, we defined a function ‘func’ that takes two integers as inputs and returns an integer. In the main() function, we declare a function pointer ‘fptr’ and then assign value to it. Note that, name of the function can be treated as starting address of the function so we can assign the address of function to function pointer using function’s name. Lets see the output :

```
$ ./fptr

 a = 2

 b = 3

 a = 2

 b = 3
```

So from the output we see that calling the function through function pointer produces the same output as calling the function from its name.

To conclude, in this article we touched some of the advanced concepts related to pointers. There can be some interesting problems related to pointers, which we might cover in some future article.

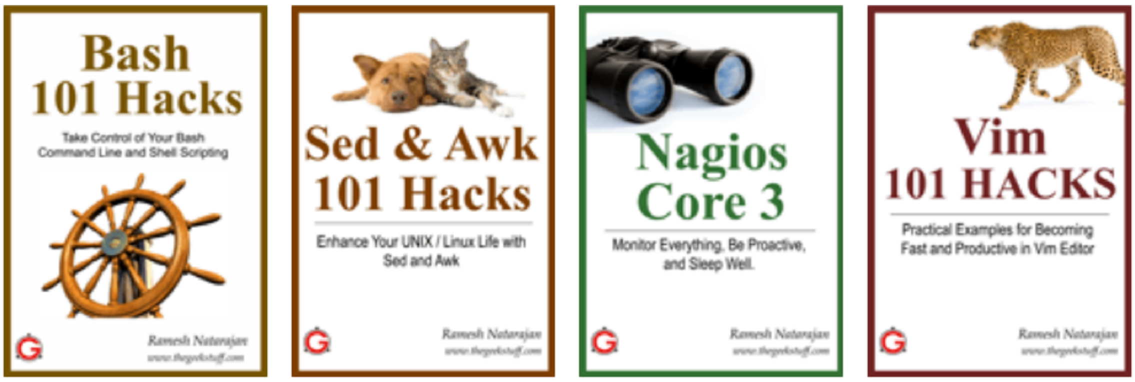
Linux provides several powerful administrative tools and utilities which will help you to

manage your systems effectively. If you don't know what these tools are and how to use them, you could be spending lot of time trying to perform even the basic administrative tasks. The focus of this course is to help you understand system administration tools, which will help you to become an effective Linux system administrator.
Get the Linux Sysadmin Course Now!

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


Bash 101 Hacks
Take Control of Your Bash Command Line and Shell Scripting
Sed & Awk 101 Hacks
Enhance Your UNIX / Linux Life with Sed and Awk
Nagios Core 3
Monitor Everything, Be Proactive, and Sleep Well.
Vim 101 HACKS
Practical Examples for Becoming Fast and Productive in Vim Editor
Ramesh Natarajan
www.thegeekstuff.com

{ 82 comments… read them below or add one }

1 cee January 27, 2012 at 3:43 am

Hi,
This article is so good, I will print it out for my apprentice (and myself).
It would be nice if this blog had a 'print this site with all pictures and the whole formatting' option.

Best Regards,
cee

2 Himanshu January 27, 2012 at 4:18 am

@cee
Thanks!!!

3 Makis January 27, 2012 at 6:40 am

Very nice examples. Good work! I believe it would also be nice to cover some dynamic memory fundamentals using pointers (ie dynamic single and two-dimensional arrays). Also consider the possibility of covering this dynamic memory tutorial not only for C but also for the C++ language.

4 apol January 27, 2012 at 12:12 pm

Yeah ! you've wrote it as you said before the end of january 😉 Thanks a lot for this article and your site in general.

5 Jay January 29, 2012 at 4:20 am

The article is very nice..and useful as well.

If the uses mentioned then it would be more and more helpful.

6 Balakrishnan M February 6, 2012 at 12:44 am

Clearly explained.. Useful one !!

7 divya March 28, 2012 at 8:35 am

explanation and article both are very nice ……..:)

8 priya April 15, 2012 at 2:45 am

thnk u so much..ifor giving such difficult concept in a nice manner…

9 raj May 19, 2012 at 12:27 pm

good explanation with examples……………..!!!thanks

10 Dev June 3, 2012 at 7:43 am

Awesome article . hope will get advance article also ..
Thanks a lot. Really appreciate ur effort 😃

11 Max steel June 8, 2012 at 12:48 pm

Awesome work man…!!! I be gained my comfort level wid ptr by ur article…thanks..

12 Anonymous June 14, 2012 at 4:53 pm

Great article, it contained great detail and simple explanations of pointers.
Thanks.

13 samadhan July 5, 2012 at 7:57 am

u have explain the concept in nicely in manner,
easy to understand

14 zah July 24, 2012 at 4:12 am

Thanks for this article mayn!! really useful! I think I got what i needed for my interview!!

Thanks,
Best regards!!

15 sony July 26, 2012 at 3:25 am

hiiiiiii
this article is really good it helps us lot in understnading pointer and its working
Thanks.

16 shari July 26, 2012 at 5:37 am

simple and clean!

17 BB July 30, 2012 at 12:36 am

Useful peace of info 😃

18 Nagarajan August 1, 2012 at 10:48 am

I have learnt so much from geekstuff especially this made pointer concepts very clear!!
Thank you 😃

19 Syed Shareef Ali August 10, 2012 at 1:15 am

Awesome!!!…Very good article with clear information.
Like to know few more advanced topics.!!Please point the similar links.

20 Krishnakumar gupta August 17, 2012 at 4:22 am

Hi
This Artical is very good and easy to understand
i want to know more advance concepts of pointer
if you have please send me on my emailid

21 Mayank August 22, 2012 at 11:36 pm

Can you please suggest some good book to go into more detail of understanding and working on pointers?

22 lala August 30, 2012 at 12:12 am

concept is describedn in very simple way i like it

23 Mihai bairac September 5, 2012 at 5:20 am

Great article! Thank you!

24 kalpana b September 14, 2012 at 6:06 am

This article is easy to understand and very helpful…
Thanks a lot!!

25 Ranjith Kumar September 21, 2012 at 7:33 am

```
#include
int main(void)
{
char **ptr=NULL;
char *p=NULL;
char c=”d”;
p=&c;
ptr=&p;
printf(“\n c=[%c] \n”,c);
printf(“\n *p=[%c] \n”,*p);
printf(“**ptr=[%c] \n”,**ptr);
return(0);
}
```

26 Ramesh September 22, 2012 at 3:01 am

Nice explanation…

27 raju September 22, 2012 at 5:30 am

```
char ch, c;
char *ptr = &ch
ptr = &c
```

28 Niranjan September 26, 2012 at 10:45 pm

Hi everybody i want 5-10 questions and solutions in-depth on pointers please help me

29 Anonymous October 1, 2012 at 2:28 pm

“”

So we can see that in memory, pointer p1 holds the address of pointer p2. Pointer p2 holds the address of character 'ch'.

So 'p2′ is pointer to character 'ch', while 'p1′ is pointer to 'p2′ or we can also say that 'p2′ is a pointer to pointer to character 'ch'.

""

You have a typo in this paragraph.
its p1 with is pointer to pointer to character ch not p2

30 pari October 3, 2012 at 2:41 pm

Brief
And useful ,thanks alot!

31 sibusiso October 10, 2012 at 5:42 am

Brief , to the point and very useful. thanks a Million.

32 SURBHI JAIN October 16, 2012 at 3:00 am

Easy to understand pointers through this article.

33 Pranoti October 21, 2012 at 2:43 am

Thank you very much for giving concepts of pointer that to in easy way. I like the way to put the example to under each concept of pointer,pointer to array,function pointers.
Thanks…:)

34 sarita October 21, 2012 at 9:22 am

thanks!

35 Deepika October 30, 2012 at 1:53 am

Example and explanation both are goood…

36 ashutosh November 1, 2012 at 9:28 am

if int variable holds 2 bytes than what should pointer int occupy?????????? i.e int a=5 means a occupy 2 bytes and int *a where a=&a; than what should *a occupy in memory???????????

37 Neeti November 6, 2012 at 9:03 am

Thanks

38 pavithra November 6, 2012 at 10:54 am

Thank you…a very neat explanation
what does this statement mean
int (*ptr)[3] = &a[0];

39 Harsha P M November 13, 2012 at 5:07 am

can u tell about pointer to an array i.e int (*p)[10];.and also performance between
array of pointers n pointer to an array

40 A. Das November 19, 2012 at 9:23 am

This article really helped. Thank You.

41 Kush Sahni November 23, 2012 at 8:09 am

Nice and easy way to explain pointers..!! good job

42 Alok November 28, 2012 at 11:19 am

Very nicely explained. This really helped.

43 Anonymous November 29, 2012 at 1:15 am

int (*p)[10]; can u tell about pointer to an array in the same site. ur doing a good job…hatsoff…

44 Sandip pawar December 1, 2012 at 9:17 pm

Ok,i can ustand

45 Anoop December 13, 2012 at 11:04 pm

Can you please implement the Inheritance concept in C using function pointer?

46 Ujjwal December 29, 2012 at 5:40 am

Above deceleration of every definition is very easy to understand……
nice job buddy……….

47 zubair January 28, 2013 at 12:21 pm

this is very gud can you tell me the other example more

48 Bharath January 31, 2013 at 11:54 am

above described ex are nice.but i advice you to provide much more examples.

49 sajan palakkad February 24, 2013 at 8:33 am

yes it's realy hlpful
thnks

50 jignesh March 5, 2013 at 1:26 am

realy it's very nice…………

51 Manikandan March 21, 2013 at 11:07 pm

Good to begin with

52 raj March 31, 2013 at 8:21 am

simple and very easy to understand

53 vijay chauhan April 6, 2013 at 11:32 am

superb!!! Easy to understand and very helpful as well..keep it up!

54 Rohit April 29, 2013 at 6:00 am

Thank you so much. Finally got a hang of the difference between the *ptr and the **ptr and the ***ptr.

Thank you very much

55 karan patel May 4, 2013 at 6:52 am

```
#include

int func (int a, int b)
{
```

```
printf("\n a = %d\n",a);
printf("\n b = %d\n",b);

return 0;
}

int main(void)
{
int(*fptr)(int,int); // Function pointer

fptr = func; // Assign address to function pointer

func(2,3);
fptr(2,3);

return 0;
}
```

you explain this concept . i am not understand what is the need of pointer to this function when func(2,3) does whatever we want to do ……….
can you give such example which explain in what setutation we should use pointer to function?????

56 prasad May 27, 2013 at 12:14 am

it is good

57 sara June 13, 2013 at 2:53 pm

aWsm…gr8…i got the idea abt pointer in just 15 mins…:)

58 amit June 19, 2013 at 8:35 pm

nice explanation…of key points in pointers……

59 satyajit July 12, 2013 at 1:07 am

Thanks for such awesome explanation…

60 gaurav bansal July 16, 2013 at 11:09 pm

very nice example pointer programs .

61 sandeep July 30, 2013 at 4:21 am

nicee…very easy to understand

62 yogesh August 11, 2013 at 4:52 am

when i run function pointer example on my system it taks atleast 15s to execute,

why so and please give me link of your further articles on pointer

63 surendra August 21, 2013 at 10:10 pm

please make me understand about the output of program.

```
#include
void main()
{
static int a[3]={0,1,2};
int * p[]={a,a+1,a+2};
printf("%u\n%u\n%d",p,*p,*(*p));
getch();
```

}

64 payal September 4, 2013 at 8:22 am

vry good explanation about pointer to pointet

65 Bidyut. September 6, 2013 at 8:44 am

Good explaination .

66 tarun sharma October 1, 2013 at 10:56 am

nice explanation,………..you reali made it look easy…..
Thank you 😃

67 Anonymous October 12, 2013 at 2:54 am

very good explanation about pointers

68 Shoaib October 22, 2013 at 12:30 am

Examples are good….
need more example which returns pointers…..
Please

69 allex mclen November 5, 2013 at 6:20 am

Thanks a lot man!!!
these were some of the concepts which never get cleared…by my boring course book

70 muhammad arif November 11, 2013 at 6:56 am

i need more help on pointer . can you please advise me any good books on pointer
thanks a lot….

71 Varun November 27, 2013 at 6:08 pm

```
int rows = 2, col = 45;
ptr = (char **)malloc(sizeof (char) * rows);
int i;
for (i = 0; i < rows; i++)
{
ptr[i] = (char *)malloc(col* sizeof (char));
}
for (i = 0; i < rows; i++)
{
printf("Address of row-%d is %p\n", i, ptr+i);
}

for (i = 0; i < rows; i++)
{
printf("Enter Names\n");
scanf("%s", ptr[i]);
}

for (i = 0; i < rows; i++)
{
printf("Entered Names: %s\n", ptr[i]);
printf("Address(ptr[i]): %p\n", ptr[i]);
printf("Address(ptr+i): %p\n", ptr+i);

}
```

```
printf("ptr[1] – ptr[0] = %p\n", ptr[1] – ptr[0]);
```

```
Output:
Address of row-0 is 0×9132008
Address of row-1 is 0x913200c
Enter Names
sad
Enter Names
c
Entered Names: sad
Address(ptr[i]): 0×9132018
Address(ptr+i): 0×9132008
Entered Names: c
Address(ptr[i]): 0×9132050
Address(ptr+i): 0x913200c
ptr[1] – ptr[0] = 0×38 //Here expecting 0x2d (Col = 45) but why 0×38??
```

Not getting please let me know.

Thanks in advance

72 KIRAN SOLANKI December 20, 2013 at 3:07 am

Your answer is good. I need example of pointer how is work and how point to one or more….

73 TiTo December 26, 2013 at 9:52 pm

You forgot pointer to array.. Rest are very good explained…

74 John Walker January 4, 2014 at 5:50 am

Great. After struggling with the details arrays of functions I sorted
it out in a couple of hours,
All the best
Thankyou
John Walker

75 B.MATHANGI January 17, 2014 at 9:35 am

exclusive and crispy informations about c it’s fabulus!!!

76 Darren B January 30, 2014 at 9:31 am

Excellent article. One of the best and clearest explanations of pointers I have seen.
However, I may be mistaken but I think there is a typo in section “2. C Pointer to Pinter”.

I think the comment

“So ‘p2′ is pointer to character ‘ch’, while ‘p1′ is pointer to ‘p2′ or we can also say that ‘p2′ is a pointer to pointer to character ‘ch’.”

should be

“So ‘p2′ is pointer to character ‘ch’, while ‘p1′ is pointer to ‘p2′ or we can also say that ‘p1′ is a pointer to pointer to character ‘ch’.” ???

Thanks

77 tp February 20, 2014 at 7:25 am

Eminent article.
It clears my doubts about pointer to pointer

but will u describe what's the difference between
const int *p and int const *p

Thanks

78 Sony March 11, 2014 at 9:52 am

It is a helpful article to clear the basic concept on pointer..thanks

79 DefinitionOfScum May 2, 2014 at 2:30 pm

This article and the one before it on pointer was amazing. Thank you very much, cleared up everything I ever wanted to know about pointers.

80 peddaraju May 5, 2014 at 7:33 pm

its good explanation…. can you provide me where this pointer to functions are used?

81 Aditi July 26, 2014 at 12:31 am

awesome explanation of pointers
thankew!!!!

82 Azam khan September 7, 2014 at 6:20 am

Hi
Explanation with ex is good

Leave a Comment

Name

E-mail

Website

☐ Notify me of followup comments via e-mail

Submit

Previous post: Linux Time Command Examples

Next post: 5 Useful Perl 5.10 Features – Say, State, ~~, Defined OR

- RSS | Email | Twitter | Facebook | Google+

  Search

-

- **COURSE**
  - Linux Sysadmin CentOS 6 Course - Master the Tools, Configure it Right, and be Lazy

- **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

  Like

  7,757 people like The Geek Stuff.

  Facebook social plugin

- **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

Follow

- **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.

- **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

- **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+