10 Practical Linux nm Command Examples

by Himanshu Arora on March 2, 2012

The nm commands provides information on the symbols being used in an object file or executable file.

The default information that the 'nm' command provides is :

- Virtual address of the symbol
- A character which depicts the symbol type. If the character is in lower case then the symbol is local but if the character is in upper case then the symbol is external
- Name of the symbol

The characters that identify symbol type describe :

- **A** : Global absolute symbol.
- **a** : Local absolute symbol.
- **B** : Global bss symbol.
- **b** : Local bss symbol.
- **D** : Global data symbol.
- **d** : Local data symbol.
- **f** : Source file name symbol.
- **L** : Global thread-local symbol (TLS).
- **l** : Static thread-local symbol (TLS).
- **T** : Global text symbol.
- **t** : Local text symbol.
- **U** : Undefined symbol.

Note that this list is not exhaustive but contains some important symbol types. For complete information please refer to the man page of this utility.

The default way to use 'nm' utility is :

```
$ nm <object file or executable name>
```

if no executable name is given then nm assumes the name to be ‘a.out’.

With the basic idea about this utility, one may question that why this information would be required?

Well, suppose that you have an executable that is made of many different object files . Now assume that while compiling the code, the linker gives error about an unresolved symbol ‘temp’. Now it will become a nightmare to find where the symbol ‘temp’ is in the code if the code is too large and includes a lot of headers. It is here where this utility comes to rescue. With some extra options, this utility also gives the file in which the symbol is found.

Since now we have a basic idea about the nm utility. Lets understand the usage of this utility through some practical commands.

1. Display Object Files that Refer to a Symbol

The following command displays all the object files that refer to the symbol ‘func’ in my current directory

```
$ nm  -A ./*.o | grep func

./hello2.o:0000000000000000 T func_1
./hello3.o:0000000000000000 T func_2
./hello4.o:0000000000000000 T func_3
./main.o:                   U func
./reloc.o:                  U func
./reloc.o:0000000000000000  T func1
./test1.o:0000000000000000  T func
./test.o:                   U func
```

Note that the -A flag is used to display the file name along with other information. So we see that in the output we get all the object files where the symbol ‘func’ is being used. This could be extremely useful in cases we want know how which object files are using a particular symbol.

2. Display all Undefined Symbols in an Executable

The following command lists all the undefined symbols in an executable file ‘1’

```
$ nm -u 1
w _Jv_RegisterClasses
w __gmon_start__
U __libc_start_main@@GLIBC_2.2.5
U free@@GLIBC_2.2.5
U malloc@@GLIBC_2.2.5
U printf@@GLIBC_2.2.5
```

Note that the flag ‘-u’ is used in this case for listing only the undefined symbols. This could be extremely useful in cases where one may want to know about the undefined symbols being used in the code that could either really be unresolved or could be resolved on run time through shared libraries.

On a related topic, you should also understand how GCC linking process works.

3. Display all Symbols in an Executable

The following command lists all the symbols in the executable ‘namepid’ but in sorted order of their addresses

```
$ nm -n namepid
```

```
w _Jv_RegisterClasses
w __gmon_start__
U __libc_start_main@@GLIBC_2.2.5
U exit@@GLIBC_2.2.5
U fclose@@GLIBC_2.2.5
U fgets@@GLIBC_2.2.5
U fopen@@GLIBC_2.2.5
U fork@@GLIBC_2.2.5
U memset@@GLIBC_2.2.5
U printf@@GLIBC_2.2.5
U puts@@GLIBC_2.2.5
U signal@@GLIBC_2.2.5
U sleep@@GLIBC_2.2.5
U strchr@@GLIBC_2.2.5
U strlen@@GLIBC_2.2.5
U strncat@@GLIBC_2.2.5
U strncpy@@GLIBC_2.2.5
U system@@GLIBC_2.2.5
0000000000400778 T _init
00000000004008a0 T _start
00000000004008cc t call_gmon_start
00000000004008f0 t __do_global_dtors_aux
...
...
...
```

We see that by using the flag ‘-n’, the output comes out to be in sorted with the undefined symbols first and then according to the addresses. Sorting could make life of a developer easy who is debugging a problem.

4. Search for a Symbols and Display its Size

The following command searches for a symbol ‘abc’ and also displays its size

```
$ nm  -S 1 | grep abc
0000000000601040 0000000000000004 B abc
```

So we see that the flag -S displays an extra information about the size of the symbol ‘abc’

5. Display Dynamic Symbols in an Executable

The following command displays on dynamic symbols in the executable ‘1’.

```
$ nm  -D 1
w __gmon_start__
U __libc_start_main
U free
U malloc
U printf
```

This could be extremely useful in cases where one is interested to know about the symbols that can only be resolved by shared libraries at the run time.

6. Extract Symbols of Various Types

Another powerful feature of nm command is to be able to extract out symbol from various types of object file format. Normally on Linux we have either ‘a.out’ or ELF format object or executable code but if an object or executable code is of some other format then also nm provides a flag ‘–target’ for it.

7. Change the Format of the nm Output

By default the format of output displayed by nm is the bsd type. We can change the format using the flag -f. The following command displays the output of nm command in posix style.

```
$ nm -u -f posix 1
_Jv_RegisterClasses w
__gmon_start__ w
__libc_start_main@@GLIBC_2.2.5 U
free@@GLIBC_2.2.5 U
malloc@@GLIBC_2.2.5 U
printf@@GLIBC_2.2.5 U
```

Similarly we can use '-f sysv' if we want the output to be in systemV style.

8. Display Only the External Symbols of an Executable

The following command lists only the external symbols in the executable

```
$ nm -g 1
0000000000400728 R _IO_stdin_used
w _Jv_RegisterClasses
0000000000600e30 D __DTOR_END__
0000000000601030 A __bss_start
0000000000601020 D __data_start
0000000000601028 D __dso_handle
w __gmon_start__
0000000000400640 T __libc_csu_fini
0000000000400650 T __libc_csu_init
...
```

Please note that the use of flag -g enables the output of only external symbols. This could come in handy while specially debugging external symbols.

9. Sort the nm Output by the Symbol Size

The following command sorts the output by the size of symbols

```
$ nm -g --size-sort 1
0000000000000002 T __libc_csu_fini
0000000000000004 R _IO_stdin_used
0000000000000004 B abc
0000000000000084 T main
0000000000000089 T __libc_csu_init
```

Note that the flag –size-sort sorts the output with respect to size. As already explained -g is used to display only external symbols.

10. Specify nm Options in a File

Another valuable feature of nm is that it can take its command line input from a file. You can specify all the options in a file and specify the file name to nm command and it will do the rest for you. For example, in the following command the nm utility reads the command line input from the file 'nm_file' and produces the output

Please note that the symbol '@' is required if you provide the file name.

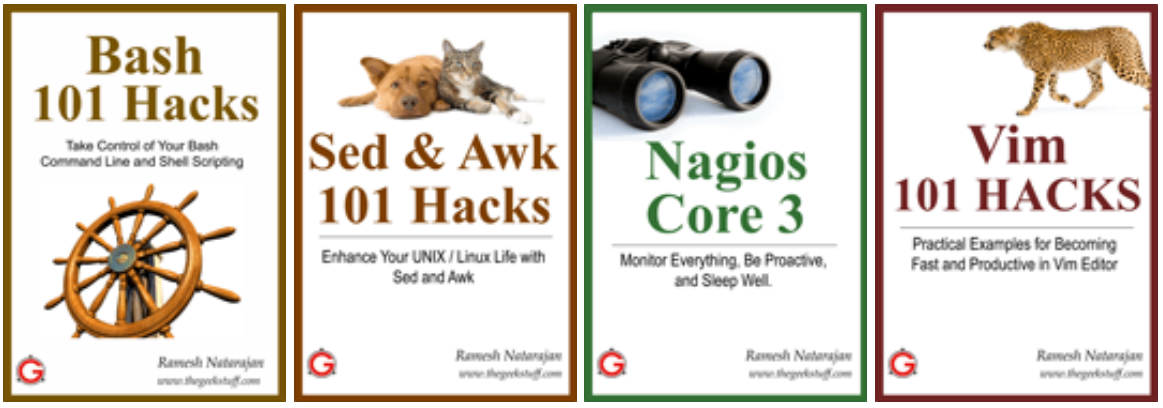
```
$ nm @nm_file
```

```
0000000000000002 T __libc_csu_fini
0000000000000004 R _IO_stdin_used
0000000000000004 B abc
0000000000000084 T main
0000000000000089 T __libc_csu_init
```


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



{ 8 comments… add one }

- rakesh March 2, 2012, 3:55 am

  Good article, the executable name should have been good if its not “1”, it might be mistaken as a parameter :-(.

  Good work 🙂

  Link
- Lakshmanan Ganapathy March 2, 2012, 5:41 am

  Really good work…

  I enjoyed reading it. One thing is nm won’t work on object file which are striped using “strip” command.

  Link
- hdaz March 2, 2012, 11:39 am

in what situations might this be useful? are there any good examples???

Link

- Yuvaraj March 5, 2012, 10:53 pm

  Hi Friends,

  Really Good article .I want add one more point here.
  nm can be applied on un striped library(static and dynamic) also.

  Ex1. nm /usr/lib/libfplib.so.1.2.3
  Ex2. nm nm /usr/lib/libc.a
  The above helps to solve undefined references in linking .

  Link

- Matt March 7, 2012, 3:50 pm

  The NM command is not available on my version. What package do I need to download? I have been searching but can't seem to find a reference to it.

  Thanks.

  Link

- Himanshu March 10, 2012, 5:05 am

  @Matt : Which Linux distro and version are you using??

  Link

- shouume April 17, 2012, 9:38 am

  I hope you don't mind questions from wannabe geeks. What is an object file? When I try using this command, I get this error message, "File format not recognized."

  Mommy, I found it(?), at least I think I did:
  object file \ob"ject file\, object program \ob"ject program\, n.
  (Computers)
  A computer program which has been translated into machine
  language by a compiler and assembler, but not yet linked into
  an executable program; sometimes called an obj file,
  because its file name typically has the extension "obj" .
  [PJC][The Collaborative International Dictionary of English v.0.48 :]
  But now, I am wondering what a symbol is.

  Link

- moa October 16, 2013, 10:24 am

  @Matt: binutils in debian/ubuntu

  Link

Leave a Comment

Name

Email

Website

Comment

Submit

Notify me of followup comments via e-mail

Next post: Linux Signals Fundamentals – Part I

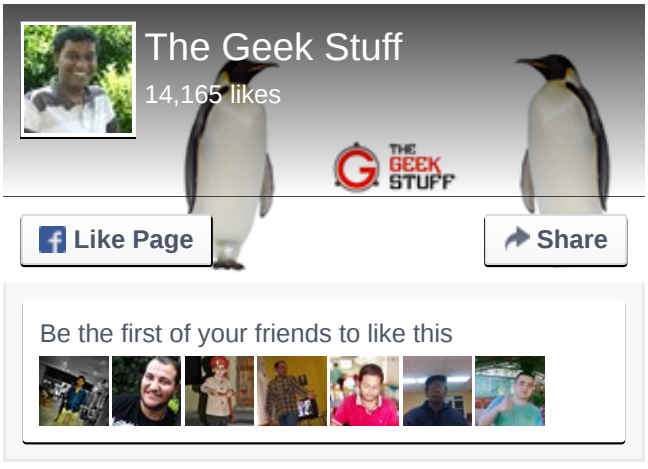
Previous post: 10 Linux DIG Command Examples for DNS Lookup

RSS | Email | Twitter | Facebook | Google+

Google™ Custom Search 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



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.

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