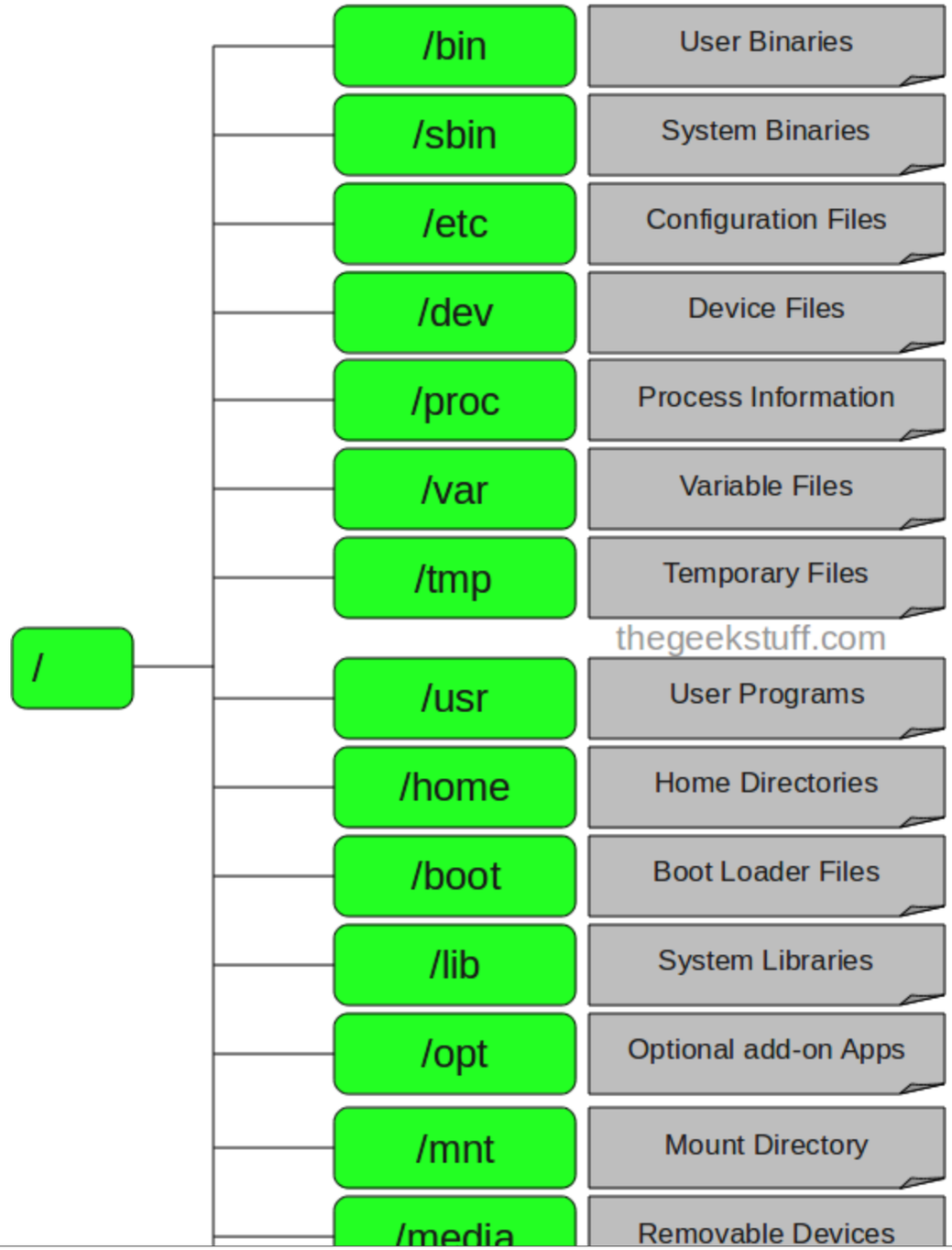
Linux Directory Structure (File System Structure) Explained with Examples

by RAMESH NATARAJAN *on* SEPTEMBER 8, 2010

Have you wondered why certain programs are located under /bin, or /sbin, or /usr/bin, or /usr/sbin?

For example, less command is located under /usr/bin directory. Why not /bin, or /sbin, or /usr/sbin? What is the different between all these directories?

In this article, let us review the Linux filesystem structures and understand the meaning of individual high-level directories.



POPULAR POSTS

15 Essential Accessories for Your Nikon or Canon DSLR Camera

12 Amazing and Essential Linux Books To Enrich Your Brain and Library

50 UNIX / Linux Sysadmin Tutorials

1. / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

3. /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.

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

- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

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

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*

- For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

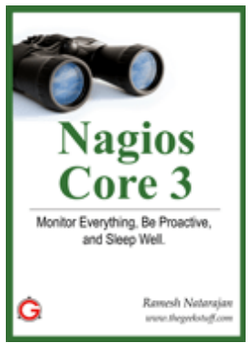
- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

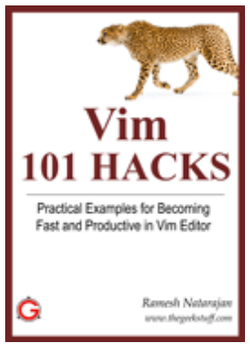
16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

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





Comments on this entry are closed.

vagvaf

September 8, 2010, 2:51 am

perfect!!! it was about time for someone to do that!

∞

Felix Frank

September 8, 2010, 3:12 am

It’s a common misconception, but /usr has in fact nothing to do with the word “user”, but instead is an acronym for “UNIX special routines”. No, it doesn’t matter 😉

∞

Jaydeep

September 8, 2010, 3:41 am

nice explanation…!

but should have been little more elaborative…!

∞

Feronera

September 8, 2010, 4:46 am

Great!!!.
Almost user has never known.

∞

Anonymous

September 8, 2010, 8:34 am

hmmm… still why ‘/usr/bin/’, or ‘/usr/local/bin/’ is missing.

∞

SG

September 8, 2010, 8:55 am

Thanks for putting this together – I have at times wondered what some of the directories like /srv & /opt meant since im not in them too often.

∞

BalaC

September 8, 2010, 9:18 am

Some system has /src folder which holds the kernel source code

∞

prakash.p

September 8, 2010, 10:39 am

Great! Simple and Clear.

∞

Srikanth

September 8, 2010, 11:38 am

Thanks.

∞

Ronald

September 8, 2010, 1:45 pm

@Felix Frank:
http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/usr.html
And I quote:
“In the original Unix implementations, /usr was where the home directories of the users were placed (that is to say, /usr/someone was then the directory now known as /home/someone). In current Unices, /usr is where user-land programs and data (as opposed to ‘system land’ programs and data) are. *snap* As such, some people may now refer to this directory as meaning ‘User System Resources’ and not ‘user’ as was originally intended.

∞

Shanil

September 8, 2010, 2:54 pm

Thanks mate, Just in time lol..Thanks again

∞

Øsse

September 8, 2010, 4:15 pm

I’m still unsure on what /usr/sbin contains. System binaries for user-land applications? Does that make sense? From the description above it seems a bit willy nilly if a binary is located in /sbin or /usr/sbin.

∞

hulk

September 8, 2010, 5:08 pm

Thanks!!!!!! This is great!
Why is my /srv dir empty?

∞

Ronald

September 9, 2010, 9:38 am

@Ronald
I see – thanks for clearing that up.

∞

dj

September 9, 2010, 7:40 pm

Another source:
Linux Filesystem Hierarchy Standard (fhs): http://www.pathname.com/fhs/

∞

ABCD

September 10, 2010, 11:51 pm

@ Øsse:

On some systems, /usr is on a separate partition (possibly on the network) from /. As such, anything that was absolutely required in single-user mode would have to be in /bin, /lib, /etc, or /sbin. /usr/sbin is for system administration tools that don't need to be used at that point. (Also, programs in /bin and /sbin need to have their shared libraries in /lib so that they are available before /usr is mounted — and / can be very small in such a setup).

∞

Yaro Kasear

September 11, 2010, 12:20 pm

I love the Filesystem Hierarchy Standard. Love it.

A lot of people new to Linux think it's directory structure is a horrible gaggle of directories and very disorganized.

They are incorrect and its because they don't understand it. The FHS spec file has a much more detailed explanation of how a POSIX filesystem is meant to look like. But it's extremely exhaustive and can even be confusing.

A good way to explain POSIX's FHS is to say, "everything is in its place, and there is a place for everything."

The FHS makes it also easy for me to figure out where something is likely to be, be it a configuration file or an asset for a program I use that I want to fiddle with. Prediction really is important for a filesystem hierarchy. And even a lot of Linux software counts on the FHS being obeyed, though if you're building from a source from a system not very compliant you can use the configure script's arguments to make the changes to how the program builds so it'll look in the right places.

I do have some criticisms, there are some flaws in the FHS. I find /opt to be a horrible thing, a directory to encourage a completely disorganized mess for some developers to just dump everything they've made instead of trying to make the FHS work. Navigating /opt for assets or configuration is a horrible nightmare, and if I were to be making a

tutorial on making a program, I would tell people to avoid /opt completely and instead scatter their files into the better places where they'll be better cared for: /usr, /etc, whatever else. /opt is evil.

I also find /mnt to be completely unused in favor of /media today. Even for local filesystems not in removable media: HAL and udev completely ignore /mnt. I think the FHS should deprecate it.

/usr/local I also find to be unneeded, especially since it doesn't serve any real purpose that /usr itself can't. It's a relic from the thin-cient days where /usr itself isn't on the system, but /usr/local is. I don't have to tell you, even in server-client systems ths isn't even needed anymore. /usr/local should also be deprecated since the base /usr structure already covers all the bases.

Now, I wish to point out that /dev itself is also treated like a pseudo filesystem. When Linux is not running, there is actually only about 3-4 static files in there, and that at runtine, udev populates /dev as the system boots, usually in the space between when the initramfs is mounted and when the permanent initscripts hit the modprobe phase. This is because harware, like processes, can change in a system and it needs to be prepared to change with the hardware. It's also because in the REALLY old UNIX days, there WAS no real hardware autodetection method as we know them today: No udev or HAL. You added some hardware, you had to make sure a few things:

1. You have the driver. Chances are in those days the kernel you were working with was 100% monolithic, none of those newfangled loadable kernel modules. Thus you ran a gamble of two outcomes with your UNIX implementation: Your kernel had the driver for your software built right in, not unlike if you build a driver right into the Linux kernel

instead of as a module. OR. You had to provide a userspace driver specifically written for your kernel… somehow. This is part of why in the early days UNIX-toting IT departments in a company had their own programmers and designers. This was made hard in those days because most UNIX was still proprietary.

2. You understand how it all works: The driver, the hardware, and the kernel. At least enough to know how to set it up so that the driver can interface with the kernel and, in turn, interface with the hardware.

3. You know exactly WHAT device file to create. Take a look in /dev sometime. Notice the file nomenclature. You had to be sure the filename is correct so that the driver and the programs using the hardware actually can talk to each other by means of raw data.

Now, even in those days a lot of hackers had shell scripts to help them, but they were pretty much on their own. They didn't have thiongs liek udev or HAL probing the entire system for hardware, and they certainly didn't have kernel modules to make sure their hardware, and only their hardware, was being fussed over by the kernel.

∞

ABCD

September 11, 2010, 2:17 pm

@Yaro:
The main reason nowadays for the split between /usr and /usr/local is that / and /usr are generally maintained by your package manager, where /usr/local is for the local sysadmin. If you start changing things in /usr without going through the package manager, you are

on your own (things may suddenly break on updates); if you just install into /usr/local, then you maintain independence from the PM without the breakage that would be caused by overwriting files that the package manager thinks it owns.

∞

Yaro Kasear

September 11, 2010, 2:36 pm

@ABCD – I stand corrected. I suppose that makes sense, though in my experience /usr/local isn't as reliable as it could be in PATH (I've even had to explicitly tell some configure scripts to do /usr instead to work.)

I like how Arch does it. They strongly recommend making a PKGBUILD so your package manager can keep an eye on it. It's probably the best way to do source builds of something. That and you can submit that PKGBUILD to the AUR and share the fruits of your labor: Figuring out how to build and install something.

Still, my opinion on /opt stands, it seems unneeded and a royal pain to navigate if you want something from there.

∞

Øsse

September 12, 2010, 2:28 pm

Thanks for the explanation on /usr/sbin, ABCD 🙂

∞

Domen Kožar

September 16, 2010, 10:12 am

/srv/ info is not correct.

It should contain data, *served* by this Linux box. There is no standard yet although, how should the data be structured.

∞

Gabriel Menini

September 21, 2010, 7:34 am

usr stands for Unix System Resource

∞

Satya

September 24, 2010, 12:12 pm

Thanks for this great article. It’s brief and easy to read.
Btw, there is a typo in /medica/cdrom (an extra c in media)

∞

Ron

September 27, 2010, 2:43 pm

I really enjoy your blog. Keep up the good stuff!

∞

SurrealWombat

October 11, 2010, 5:06 pm

IIRC sbin are actually static binaries, i.e. they statically link their library dependencies and so do not depend on any library files (i.e. /lib, /usr/lib, /usr/loca/lib, et al).

Originally there was a small boot partition on the disc (smaller the partition the less likely to have a disk error). This boot partition had /etc, /sbin and /boot. This is sufficient to be able to boot the OS and load the remaining partitions.

HTH.

∞

yaro@marupa.net

October 11, 2010, 7:08 pm

@SurrealWombat – No, things in sbin can be dynamically linked to their libraries. The “s” in sbin doesn’t mean “static” it means “system.”

sbin is different from bin in that it’s usually just utilities system administrators would need to use, and typically will need root permissions to work (Think fdisk or dd or fsck.), whereas bin is general purpose “everyday use” binaries (sh or ls or cp).

Same applies for the same subdirectories in /usr and /usr/local.

Also, what is found in the “root” sbin/bin is primarily stuff you absolutely WILL need in the event you can only mount / and no other filesystems. The FHS is pretty clear on that one. stuff in /usr and /usr/lib cannot be planned on in single user mode or in emergencies, since they can and often are on separate filesystems.

Now, that’s not to say there are not system binaries that are statically linked. But typically you’ll find static binaries are only used in early userspace (As in, when you are running the initramfs, before permanent / is available to the system.)

∞

Ashish

November 2, 2010, 4:49 am

Champ , that was amazingingly AWESOME article……..
Such article makes you FALL IN LOVE with LINUX all again……

∞

failurehappening

December 1, 2010, 3:31 am

Nice work intenets, a very informative comments section… 9.5/10

∞

sxazer

December 5, 2010, 4:43 pm

i still don’t get what is the differece between all these bin directories for single user like me,why not just one dir to avoid the hassle?

∞

Yaro Kasear

December 5, 2010, 8:55 pm

@sxaxer – Because the different bin and sbin directories actually serve a different purpose overall. Yes, they hole executable binaries, but /bin and /usr/bin hold very different priority executables. It’s not a matter of how many users you have, it’s a matter of keeping things organized. Chucking everything in one huge binary directory is going to make looking for the binaries that much harder.

/bin and /sbin are for binaries needed just in case you need to go into single user mode, which is used purely for recovery/maintenance purposes, and only really should contain software needed to make sure the system WORKS without additional services running or filesystems mounted. /bin being the absolute most common command every user is likely to use no matter what (ls, cp, ps, etc.) whereas /sbin is all about system administration like fsck, init, or fdisk.

/usr/bin and /usr/sbin are basically the non-critical programs on your system. The system doesn't technically need them to boot at least to a barebones recovery mode, though stuff in both those being removed is likely to keep the FULL system from working, as things like X or your actual normal applications are to end up in /usr/bin or /usr/sbin.

/opt is special. It's basically there for the sake of programs that make no attempts to follow the FHS at all, typically proprietary applications. Though sometimes some big extras can be placed there, like a 32-bit chroot environment or multilib. But most your normal apps will be /usr/bin, and anything more than basic system commands for maintenance that an administrator might use in NORMAL system runtime situations (As in, not trying to fix something broken.) will be in /usr/sbin.

/etc will frequently have some scripts within, but mostly there to support the system during runlevel changes, such as running daemons or for bootup, reboot, or shutdown. These are called the bootscripts, the reason they are in /etc and not /sbin is that init counts on them being part of and treated as part of system configuration (As modifying these scripts is changing how your system behaves at low levels, ergo, the scripts are still configuration, just an unusual kind.)

The FHS is very well organized. If you know how it works you'll be able to quickly and easily find just about any file in Linux or UNIX that you might want to find.

∞

Yaro Kasear

December 5, 2010, 8:59 pm

@sxaxer- Also, the FHS is designed so that there can be pretty much one universal PATH environment variable that ALWAYS works. It's so you won't really HAVE to hunt down the executables, you can just USE them. The FHS approach is oodles better than Windows' approach, for sure.

∞

sxazer

December 6, 2010, 9:37 am

@Yaro Kasear
Thank you very much.
But one last question: when i want to compile something that i want to go to /usr/bin and /usr/lib ,what should i type after ./configure?

∞

Yaro Kasear

December 6, 2010, 1:28 pm

That would be ./configure –prefix=/usr

NOTE: Not all packages go by GNU's autotools standards. If the source package uses cmake you're likely to find you need to do it differently.

∞

Aleve Sicofance

December 16, 2010, 7:35 am

Maybe this makes sense for server administrators, but the FHS is a complete mess for a desktop system. OS X and Gobolinux have all the arguments about it, so I won't repeat them here.

No ordinary human being can find any logic in this "organization". Only people who have spent years getting to know this thing can say that it is "organized".

∞

Yaroukh

December 29, 2010, 3:31 am

@Aleve Sicofance: +1

∞

Les Garwood

January 9, 2011, 10:01 pm

Damn, this is one helpful site. THANKS!

∞

Ramesh Kutumbaka

May 12, 2011, 5:50 am

Very Nice Explanation.

∞

robinatw

June 23, 2011, 3:39 am

well, thanks a million.. continue keep up upate…

∞

ramesh

July 14, 2011, 6:19 am

Nice article with clear and precise information! Thanks a lot!

∞

Baldev Singh

August 1, 2011, 5:29 am

how to install the plugins in the ubuntu…….
my MEdiaPlayer is not gives the output in linux but working in Windows….
how to listen the music in the Ubuntu??????

∞

CHiLi.HH

August 8, 2011, 9:14 am

One major mistake: /usr is NOT “User Programs” – these are located in “/opt”.

“/usr” stand for “UNIX System Resources” and contains ONLY system related files.

∞

Yaro Kasear

August 8, 2011, 10:07 pm

@CHiLi.HH – Reread the FHS, and try again. According to the FHS, which is the sole true authority on how things like /usr is used: “/usr is the second major section of the filesystem. /usr is shareable, read-only data. That means that /usr should be shareable

between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.”

Nowhere does it say /usr should be used only for system files. In fact, it even goes on to outline EXACTLY HOW user software should be installed inside: http://www.pathname.com/fhs/pub/fhs-2.3.html#THEUSRHIERARCHY

In fact, the FHS explicitly states that NOTHING THE SYSTEM NEEDS SHOULD BE IN /usr. That pretty much EXCLUDES essential system files, in several places it even points out where it wants the actual essentials, and they’re nowhere near /usr:

“/bin : Essential user command binaries (for use by all users)”
“/etc : Host-specific system configuration”
“/lib : Essential shared libraries and kernel modules”
“/sbin : System binaries”

That pretty much covers your “system files.” Note how none of these are in /usr.

Now, /opt can and is used for “user software.” But so is /usr, and in fact, /opt is generally only used in Linux or UNIX these days to handle software that won’t conform to the FHS or won’t easily be configured to apply to it.

Almost all this, I should note, is explicitly stated by the FHS, except for how /opt is used in real life. All the standard says is “/opt is for add-on software” but it also goes on to outline usage of /usr you incorrectly claim it’s not supposed to be used for.

∞

Yaro Kasear

August 8, 2011, 10:12 pm

I forgot to mention: /usr is frequently placed in a seperate partition because it’s treated as an “extension” of the normal hierarchy. And the FHS also says that anything needed in maintaining the system should be on the root counterparts of directories in /usr: /bin, /sbin, /include, /lib, and /etc.

/boot actually can and often is recommended to be used on a seperate fileystem. As it turns out, you can completely LEAVE OUT the /boot partition from being mounted by the OS in question. Once the system is up and running in an initramfs, /boot really is completely unneeded by the system. Still, it’s easier to configure bootloader settings if /boot is readily available and mounted and most distributions will end up mounting it anyway for “completeness.”

Let me sum up: No, /usr is not for system files. /bin, /sbin, /etc, /include and /lib are.

∞

M.Sch

September 21, 2011, 4:17 am

Exactly 11 years ago I started with Linux. Then so would be a brilliant representation of the root directory have been more than great. We then have the list yet drawn with ASCII

characters;-)
11 years later I love Linux still is – but now on a much higher level

∞

Perumal

October 5, 2011, 4:50 am

Thanks!!!!!! This is great!
Why is my /srv dir empty?

∞

Yaro Kasear

October 5, 2011, 11:37 am

@Perumal – Are you running any servers? Are they configured to treat /srv as their home directory? A lot of servers actually use /home.

∞

Rizwan

December 15, 2011, 1:23 pm

simple and pretty straight forward explanation.

∞

mahi

December 18, 2011, 11:31 am

thanks it help a lot !!!

∞

pravin

January 22, 2012, 12:46 pm

Nice……. It is helpful for me…

∞

adrian

January 24, 2012, 10:20 am

thanks a lot. this website is a great find for me. big help! 🙂

∞

Jens

January 25, 2012, 2:14 pm

Found this one and have to say thx! I made a printout and pluged it on the wall, cause i am more on beginner level. 😉

Cheers, Jens

∞

Ranar Gomes

February 13, 2012, 10:29 am

thnx a million….

∞

André

March 17, 2012, 9:20 am

/sys directory is missing. Since kernel 2.6 it provides device and driver info and an API. It also makes things in /proc depricated, e.g., ACPI, more info: http://en.wikipedia.org/wiki/Sysfs

∞

Steven Webb

March 17, 2012, 6:17 pm

Actually /sbin is for statically-linked binaries that dont require any dynamic libraries in case /lib goes away for some reason, I believe.

∞

cmthornton

March 18, 2012, 2:45 am

This reference is missing /sys and /run.

∞

PM

March 18, 2012, 1:26 pm

When you say “/usr/local contains users programs that you install from source”, what do you mean by source? Do you mean pcakages from CDROMs and DVDs?

∞

Roger Gilmartin

April 25, 2012, 4:14 am

Good, clear explanation. Thanks.

∞

benhank

May 9, 2012, 12:47 pm

This is absolutely AWESOME! do you have any stuff for centos?

∞

Maddy

May 12, 2012, 3:00 am

ya its good explanation, but i want to know about Kernel, Bash Shell, IP address etc..
how can i learn more about it????

∞

Ashok

May 14, 2012, 11:27 am

please give me a SAN-NAS details thanks..

∞

Rakesh

June 6, 2012, 3:25 am

Is this really required to know about the directories, to get good command in linux

∞

Ivaylo

June 14, 2012, 11:50 am

WOW,

This is really really useful for a Microsoft guy like me.

Thank You

∞

rama

July 18, 2012, 10:30 pm

thanks i realy like this

∞

Igor

September 28, 2012, 10:55 am

Fantastic article, very clear and concise.

∞

Joep

January 14, 2013, 3:53 am

Ramesh,
I've been after this kind of info for a long time. Surprised it's not more commonly available. Thanks and keep it up.

∞

Pooja Narang

February 16, 2013, 7:24 am

Thank you.. Such a ellaborated file structure…

∞

souban

April 20, 2013, 1:09 pm

Thanks for this, useful for my exam revision!

∞

Ashwin

April 23, 2013, 11:48 pm

A very helpful tutorial, thanks.

∞

Shrikant Panchal

May 14, 2013, 9:32 pm

very help

∞

hosein

May 15, 2013, 1:55 am

Great job.

Thanks, clear and simple explanation. Linux file system was horrible for me, but not now !

∞

Rupan Roy

October 3, 2013, 2:10 am

Thanks dear,
Is this really required to know about the directories, to get good command in linux

∞

Visitor

November 11, 2013, 7:35 pm

This is a great summary and the illustration is great!

Thanks of sharing 🙂

∞

wante

December 9, 2013, 6:45 am

Hi .. Ramesh
clear explanation, but may be you forget to tell about /lost+found directory and it’s purpose.
after linux crash or power failure, fsck will go through the system and try to recover any corrupt files that it finds. The result will be placed in this directory.

∞

Yaro

December 9, 2013, 5:34 pm

Wante, lost+found is actually a filesystem feature of ext2/3/4. Linux installers don't create it, the formatting utilities of those filesystems do. If you don't believe me, install your favorite distro on a filesystem like reiserfs and notice that lost+found doesn't exist. Or put /home on a different partition as ext4 and notice another instance of lost+found there.

L+F is not part of the filesystem hierarchy standard and is not actually a standard directory for Linux setups, just a directory the filesystem creates for recovery.

Further, not all filesystem recovery will result in files being chucked into lost+found. If there's no data that needs recovery (Very likely in ext3/4) then it'll almost always be lucky.

Lost+found is viewed by many as a "legacy" feature in ext4, a feature that isn't necessary anymore.

Now, I know this article is old (More than three years.) but if there's a directory now that needs to be addressed, it would be /run.

∞

wante

December 9, 2013, 9:29 pm

@yaro
you are right.
i use jfs and xfs on my slackware machine 🙂 , so the lost+found doesn't exist.

∞

himani

December 29, 2013, 9:44 pm

Super cool, tk

∞

amaldev ks

January 16, 2014, 4:07 am

great !!!!!!!!!!!!!!!!1

∞

himani

January 29, 2014, 3:30 am

Perfect, explained well

∞

viplove

February 3, 2014, 12:10 pm

can u please provide info abt apache2 web server,in which directory it is stored,how to really make use of it,its daemon etc.

∞

dini

May 13, 2014, 11:09 pm

well explanation thanks

∞

A

July 31, 2014, 9:34 am

How can I identify which programmes are in which directories.

For example, I have Libre Office.

Where is writer, calc and imopress stored. How would I know (asume I don’t know)?

Wheer is Document Viewer Installed?

Where is firefox installed?

How can I investigate any porgramme and determine its folders

Thanks

∞

KEM

August 26, 2014, 6:28 am

A

If the file is in your path, then use which
“which ls”

If not, use locate and use regex or grep to help cut down the output

∞

A

September 8, 2014, 7:57 pm

Do you have any tips for locating useful commands. For example, I can use ‘which is’, but only if I know it exists.

∞

Gowtham

January 29, 2015, 2:51 pm

Very well put. Thanks!

∞

M N Srinivas

February 5, 2015, 2:12 pm

Thank you very much, for this grate resource……

∞

factgasm

February 19, 2015, 3:30 am

Thanks for the post but what intrigues me even more is why no-one (to my knowledge) has produced some sort of overlay so that as the user uses Linux the abrvtns (abbreviations) are replaced with their full names – making everything more human readable and understandable. If I remember rightly Microsoft did this with Windows twenty years ago.

Or, in Linux speak:

Tnks fr th ps ut wt intrg me evmr is wy non (kneldge) hs prdd ss ovly usr use Lx the abrv (abrvtns) re repced wh th fl nam – mak eving mr hmn read and undstdle If I rmber rigly Great Satan of Seattle do wh Windows 20 yago.

The reason that abbreviations were so ubiquitous throughout the computing industry is because memory was expensive but nowadays you have more computing power in your smart-phone than was used to send Man to the moon so for Linux still to be using abbreviations everywhere is completely criminal.

∞

Omprakash

April 4, 2015, 10:24 am

sir could you please explain the difference between BASE_DIRECTORY And HOME_DIRECTORY.
I'm getting confused in the switches (-b,-d and -m)of "useradd" command .

∞

Harigopal A

June 26, 2015, 12:49 am

Hi Ramesh,

I am very happy to contact with you, Have read your "Linux 101 hacks" book. Its pretty much interesting and very useful to learn the Linux in a easy way, I would like to thank you to provide us a such kind of great book. To learn Linux, have invested more than 10k in so many institutes, they didnt teach me as expected. Mostly in linux have learned through your book only. Once again thank you very much ramesh. will keep read your

blogs and articals. I would like to learn Oracle DBA. Please suggest me some useful tips to learn in a easy way.

∞

Charlie

November 20, 2015, 11:07 pm

Very useful in 2015. Thanks for the heads up! This informative blog style is more attractive than scouring the documentation from each distribution! ;-D
Cheers from Florida, -Carlos.

∞

shaloo

November 26, 2015, 3:12 pm

Great …….. good for newbies

∞

Kashinath

January 7, 2016, 6:36 am

Thanks for that…short and sweet 🙂

∞

Yogesh

January 24, 2016, 2:01 pm

Thanks Mate 🙂

∞

matt

April 2, 2016, 9:53 am

Missing is /root. There’s a difference between the / directory and the /root 🙂

∞

rdee

May 20, 2016, 10:52 am

hello ramesh!

Thank you for the nice work its really helpful!!!

∞

Yaro

June 17, 2016, 4:59 pm

This is actually pretty outdated now, not that this is the most accurate description. I strongly recommend looking at this instead, as this article doesn't take into account changes like /run http://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html

∞

murthi

December 20, 2016, 3:38 am

super awesom

∞

arara

December 20, 2016, 11:12 pm

greate, use for newbies

∞

Anonymous

December 20, 2016, 11:33 pm

vah super presentation

∞

Next post: How to Backup Linux? 15 rsync Command Examples

Previous post: How To Use Squid Proxy Cache Server To Control Internet Access

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

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 Twitter

Support Us

Support this blog by purchasing one of my ebooks.

Bash 101 Hacks eBook

Sed and Awk 101 Hacks eBook

you resolve a problem. Read more about Ramesh Natarajan and the blog.

Become a fan on Facebook

Vim 101 Hacks eBook

Nagios Core 3 eBook