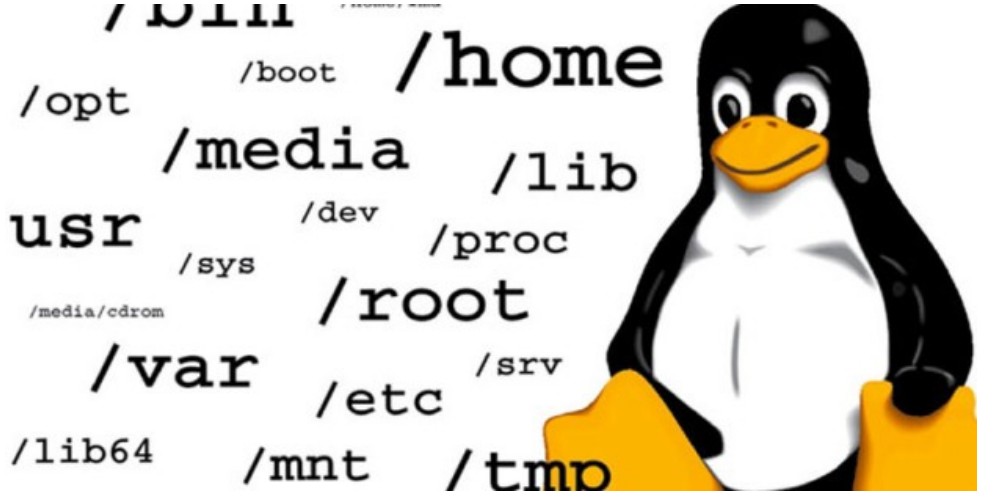
Technische Universität Berlin



Linux II – Filesystem Introduction

Training for Afghan Administrators 2015

Technische Universität Berlin

What is a filesystem?

In computing, a filesystem is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins.

By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system".

There are numerous different types of filesystems, such as ext2/3/4, btrfs, ReiserFS (etc...) on Linux, Fat(16/32) and NTFS on Windows, HFS(+) on Mac OS, or ISO9660 for CD-ROMs, etc...

ZIIK

How is the Linux filesystem organized?

The Linux ***Filesystem Hierarchy Standard*** is a convention which most Linux distributions follow and which governs the way data is saved and organized in a tree-like structure. It is maintained by the Linux Foundation. The latest version is 3.0, released on 3. June 2015.

The top-most hierarchy level is called the root of the filesystem, or in short just “/”. Below the root (“/”) level, there are more than a dozen directories, such as /bin, /boot, /dev, /etc/, /home, /lib, /lost+found, /media, /mnt, /opt, /proc, /root, /run, /sbin, /srv, /sys, /tmp, /usr and /var

Some of these directories need to be located on the same partition as the root (“/”) level, otherwise the system can not boot. E.g. /bin and /sbin need to be on the same partition like the root level, because they contain important utilities, such as the “mount” utility, which mounts other partitions into the filesystem tree in the later stages of the system booting up.

The Linux Filesystem Hierarchy Standard

/	→ Primary hierarchy root and root directory of the entire filesystem hierarchy.
/bin	→ Essential command binaries that need to be available in single user mode; for all users, e.g., cat, ls, cp.
/boot	→ Boot loader files, e.g., kernels, initrd.
/dev	→ Essential devices, e.g., /dev/null, /dev/sda.
/etc	→ Host-specific system-wide configuration files, e.g., /etc/hosts
/home	→ Users' home directories, containing saved files, personal settings, etc.
/lib	→ Libraries essential for the binaries in /bin/ and /sbin/.
/media	→ Mount points for removable media such as CD-ROMs (first appeared in FHS-2.3).
/mnt	→ Temporarily mounted filesystems.
/opt	→ Optional application software packages
/proc	→ Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount.
/root	→ Home directory for the root user.
/run	→ Run-time variable data: Information about the running system since last boot, e.g., currently logged-in users
/sbin	→ Essential system binaries, e.g., fsck, init, route.
/srv	→ Site-specific data which are served by the system.
/tmp	→ Temporary files (see also /var/tmp). Often not preserved between system reboots, and may be severely size restricted.
/usr	→ Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications
/var	→ Variable files whose content is expected to continually change, such as logs, spool files, and temporary e-mail files.

In Linux (almost) everything is a file...

In Linux, contrary to other operating systems like e.g. Windows, a wide range of input/output resources such as documents, directories, hard-drives, modems, keyboards, printers and even some inter-process and network communications are simple streams of bytes exposed through the filesystem name space. This means, that the same set of tools, utilities and APIs can be used on a wide range of resources. **Not everything is really a file, but (almost) everything can be accessed through the filesystem name space.**

Examples:

- Show computer memory usage → cat /proc/cpuinfo
- Show CPU temp →cat /sys/bus/platform/devices/coretemp.0/hwmon/hwmon0/temp2_input
- Overwrite a (pen) drive with random data → dd if=/dev/urandom of=/dev/sdX

The Linux Command Line (CLI)

- Accessible on **every** Linux system, no matter which desktop interfaces they provide
- The way to go when the graphical user interface is broken or *nonexistent*
  → This is how you will work on Linux servers!
- Often the only/easiest way to access system configuration
- Very useful to automate tasks and for batch processing

If you're confident with the CLI, you can use every Linux system!

Introduction to the Linux Command Line (CLI)

The general pattern of an OS command line interface is:

- prompt command –option1 –option2 –optionN jparam1 param2 ... paramN

Example:

- jan@ubuntu$ grep -i “jan” /etc/passwd

The **prompt** is a symbol that indicates the CLI's readiness / waiting for user input.

The **command** is the program the user wants to run.

A **command line argument** or **parameter** is an item of information provided to a program when it is started. A program can have many command line arguments that identify sources or destinations of information, or that alter the operation of the program.

A **command line option** or simply **option** (also known as a flag or switch) modifies the operation of a command; the effect is determined by the command's program. Options follow the command name on the command line, separated by spaces.

Finding help on your Linux system

- **Manual pages**

  Manual pages can be accessed by writing “man COMMAND”, e.g. “man crontab”
- **Apropos**

  “apropos KEYWORD” is a shortcut for “man -k KEYWORD” and gives a brief overview instead of the full manual pages
- **Whatis**

  “whatis COMMAND” lists a brief explanation of COMMAND
- **Info**

  The info documents are often even more complete and verbose than the manual pages. Access the info pages for any command like “info COMMAND”
- COMMAND **–help** / COMMAND **-h**

  Most utilities / commands have an inbuilt help function which can be accessed by adding –help (2 dashes) or -h (1 dash). This usually shows all the options that can be passed to COMMAND

Linux Manual Pages Sections

The Linux manual pages are divided into different sections. If there is information on a search term in different sections, use the man command and supply the number of the section as well. For example, usually you would just write “man crontab”. This gives you the info from section 1, meaning for the command itself. But if you need information on the crontab file syntax, write “man 5 crontab”

1 Executable programs or shell commands
2 System calls
3 Library calls
4 Special files
5 File formats and conventions
6 Games
7 Miscellaneous
8 System administration commands
9 Kernel routines

File names, absolute and relative paths

In Linux, **all files are case-sensitive** (document != Document != DOCUMENT). Although Linux supports **different languages, scripts and special characters (spaces, punctuation symbols), avoid them if possible**, because they require different locales / charsets and some special characters need to be “escaped” in file operations on the CLI.

Absolute paths give the location of a file by *starting from the “/”* (root level) of the filesystem.

- Examples for absolute paths:
  - /home/userXY/Documents
  - /home/userXY/Pictures/lolcat.jpg
  - /var/log/syslog

Relative paths give the location of a file by *starting from the working directory* (use “pwd” to **p**rint **w**orking **d**irectory)

- Examples for relative paths (same files as above; assuming the working directory is /home/userXY):
  - Documents
  - Pictures/lolcat.jpg
  - ../../var/log/syslog

Hands-on: copy, move, delete files and directories

- Change to your personal home directory (/home/<USERNAME> → replace <USERNAME> with your actual username)
- List the contents of your personal home directory
- Create a directory “dir1” with a 2 sub-directories “subdir[1,2]” and “SUBDIR[1,2] in your personal home directory
- Change into the newly created “dir1”
- Create 4 empty files “file[1,2]” and “FILE[1,2]” in each of the sub-directories you created above
- Copy one file from one sub-directory to another one
- Copy a few files from one sub-directory to another one (by using the wild-card “ * “)
- Copy a sub-directory (recursively / with all its content) into another sub-directory
- Move a file into another directory
- Move all files in the current sub-directory into the parent directory
- Create an archive with gzip compression that contains a few files
- Extract the archive to a folder of your choice
- Rename a file
- Delete a file
- Delete a few files (use the wild-card “ * “ again)
- Delete an empty directory
- Delete the directory and all its sub-directories and files

Use the following utilities: cd, cp, ls, mkdir, mv, rm, rmdir, touch, tar and the Linux commands cheat sheet

Alternate between using absolute paths and relative paths (you can even combine them in the same command!)

Use all available help options (whatis, --help / -h, man, apropos) on your local system (don't google!)

Using removable media (usb drives, cd/dvd, etc)

Removable media such as USB pen drives, CDs, DVDs, tapes or even hot-pluggable hard disks) can be added to the system at runtime. The process of adding a new media to the filesystem tree is called mounting. Mounting media is done with the “mount” utility, which needs at least 2 arguments: the device to be mounted and the location where it should be added in the filesystem (mount point). Usually only root can use the mount utility, so we need to prefix it with “sudo”!

Some examples (mount points must already exist!):

- sudo mount /dev/sdb1 /mnt/usb-pen-drive
- sudo mount /dev/cdrom /mnt/cdrom
- sudo mount /dev/st0 /mnt/tape
- sudo mount -o loop /path/to/disk-image.img /mnt/image

Searching for utilities / commands

There are a two utilities that will find other utilities / commands for you: **“whereis”** and **“which”**

The **“whereis”** utility finds commands / utilities and will also list locations of their manual pages (and source code files if they are installed).

- whereis cp

The **“which”** utility finds only commands (executable files) that are in your $PATH and will print their location. Useful if there are multiple commands of the same name / multiple versions, and you need to know which is being executed when you use it.

- which cp

Searching for (arbitrary) files of any type

For finding arbitrary files of any type (= not just utilities / commands) there are two utilities: “locate” and “find”

The “**locate**” utility takes just a file name search pattern (case-sensitive) as an argument. It can't search for other file attributes. The “locate” utility is very fast because it uses an indexed database. However, the database must be regularly updated (with the “updatedb” utility).

- locate “apt” vs. locate -b “\apt”

The “**find**” utility takes at least 2 arguments: the search path and the search pattern (if path is not given then “find” will search the current directory). It can search for names (case-sensitive), but also for many other attributes, like age, owner or certain permissions. Using find takes time because it checks all files on disk in the given search path.

- find /usr/share/doc -name “apt” vs. find /usr/share/doc -name “*apt*”
- find -type f -size -1M vs. find -type f -size 1M vs. find -type f -size +1M
- find . -type d -mtime -5 vs. find . -type d -mtime 5 vs. find . -type d -mtime +5

Hands-on: searching for files

Using the “find” utility:

- Find all files in /usr/share/doc that end with “.pdf”
- Find all files in /usr/share/doc that contain the word readme or README or ReadMe (→ case-insensitive search)
- Find all files in your personal home directory that are bigger than 5MB
- Find all files in your personal home dir that are between 5MB and 10MB (use “-a” option!)
- Find all files in your personal home dir modified at least 14 days ago
- Find all files in your personal home dir modified between in the last 24 hours
- Find all directories modified between 7 and 14 days ago (use “-a” option!)
- Find all files in /tmp which don't belong to root

Using the “locate” utility:

- Create some files “new_file1”, “new_file2”, etc....
- Use locate to search for “new_file”

  → *No results because the database wasn't re-indexed, so it does not know about the new files yet!*
- Update the locate database with the “updatedb” utility (prefix with “sudo”)
- Search again ...

Advanced CLI usage

Output redirecting: Redirect the output of a program to a file

Example:

- echo “hello world” > output.txt
  # write “hello world” in file output.txt

Pipelining: Use output of one program as input for another. Build a “chain of commands”

Example:

- dmesg | less
  # open the output of dmesg in the less pager

For more information See the CLI Input/Output redirection slides!

Sources and further reading:

Linux Filesystem Hierarchy Standard:

- https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
- http://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

Finding help and information:

- http://www.tldp.org/LDP/intro-linux/html/sect_02_03.html

Command line usage:

- http://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/book1.htm