File System

- Basic terms
- Access rights
- Attributes
- Quotas
- ACL
- Cryptography
- Disk partitions

File System

- Presents an abstract view on a file as a sequence of bytes.
- Translates requests for operations on files to requests on disk blocks
- Organises:
  - Sectors into blocks, blocks into block groups,
  - blocks into files (physical organisation),
  - files into directories (locial organisation).
- **Controls access to files.**
- Manages information about files.

i-Node

- Data structure that represents file (abstraction)
- Each file is represented by single i-node.
- Contains pointers to blocks with the file data.
- Contains all file metadata:
  - Access permissions (*i_mode* 16 b), owner, group, flags,
  - size, number of blocks, number of links,
  - timestamps (modification of file contents, modification of i-node contents, access, delete…)
  - Does not contain file name.

List i-Node Contents

- Contents of i-node, i.e. information about a file (not file contents), can be viewed using:
  - `stat`
  - `stat /; stat .; stat /etc/passwd`
  - `stat -f /`
- Commonly it is enough to use:
  - `ls -l; ls -id /`
  - `find`

Owner and Group

- A user is identified by UID and belongs to a group with GID.
- After login, a shell is executed with UID and GID of the user that has been authenticated.
- Every <u>process</u> has UID and GID (based on the user that started it).
- Every <u>file </u>has owner UID and group GID.
- When operation is requested on a file, the system checks whether the calling process has permissions to access it.

Owner and Group - Example

- Only root can change the owner or group.
- It is possible to change owner and group with one command:
  - `chown user file`
  - `chown -R 0:0 /root/`
- Changing group:
  - `chgrp`
  - `chgrp wheel /tmp`

File Types

- Bits 12 – 15 (*i_mode*).
  - 0140000: named socket
  - 0120000:symbolic link
  - 0100000: regular file (data)
  - 0060000: block device
  - 0040000: directory
  - 0020000: character device
  - 0010000: pipe
- In addition to data, file abstraction includes also inter-process communication and hardware.

Access Permissions

- A file has 3 types of access permissions:
  - View file contents (**R**ead), weight 4,
  - Change file contents (**W**rite), weight 2,
  - Execute file (e**X**ecute), weight 1.
- for 3 groups of users:
  - bits 6-8: for owning **user**,
  - bits 3-5: for owning **group**,
  - bits 0 – 2: for **others**.
- Commonly specified in octal representation.

Access Permissions - Example

- Permissions can be specified in octal or symbolic form
  - `644`
  - `u=rw,g=rw,o=r`
- Setting permissions on a file:
  - `chmod`
  - `chmod +x sync.pl`
  - `chmod 700 ~`
  - `chmod g=r,o-rwx group/file.txt`
  - `chmod 660 file.txt file2.txt`
  - `chmod -R g-rw ~/group/`

Permissions of a New File

- Newly created file generally inherits UID and GID from the process that created it (*creat, open, mkdir*).
- Permissions are set according to the requested *mode* and *umask* settings.
  - *mode & ~umask*
- Bits set in *umask* will be removed from the new file’s permissions.
- Common values of *umask*: *0022, 0002*.

Symbolic links

- Permissions on the symbolic links are ignored.
- Only permissions of the target file are significant.
- For example:
  - `ls –l /bin/sh`

eXecute/search Permissions

- Permission **X** on a regular file means it can be executed.
  - directly by kernel if it is a binary file or interpreted if it is a text script.
- Permission **X** on a directory means permission to “enter” the directory.
  - Access to a file will be denied if the user does not have permission to enter all directories in the path to this file.
  - We cannot set a directory that we do not have **X** permission on, as a working directory.
- Search for all files belonging to the user
  - *find / -user student*
- Before deleting, it is necessary to end user’s processes

Directory Permissions

- Directory is a file that contains file names of the files in the directory and their inode numbers..
- Furthermore:
  - Permission **R** allows reading its contents, i.e. view list of files that the directory contains..
  - Permission **W** allows changing the list of files that the directory contains, i.e.:
    - create new files,
    - rename files,
    - delete files.

Directory Permissions - Example

- It is possible that we cannot execute programs from a directory we do not have **X**, even if we have **X** on specific programs in this directory.
- Also, it is possible to create a directory where we can create and edit files but we do not see the directory contents.
  - i.e. we can work with the files in this directory but we must know their names.

Special Permissions

- Bits 9 – 11 (*i_mode*):
  - SetUID (SUID), weight 4, `u+s`
  - SetGID (SGID), weight 2, `g+s`
  - StickyBit, weight 1, `o+t`
- Examples:
  - `ls -l /usr/bin/passwd`
  - `ls -l /bin/mount`
  - `ls -ld /tmp`

Special Permissions

- SUID and SGID:
  - On a file: the program will be executed with effective permissions (EUID) of the program’s owner and group respectively.
  - On a directory: new files created in the directory will have owner or group set the same as this directory.
- Sticky Bit:
  - On a file: normally ignored.
  - On a directory: file in this directory can be deleted only by its owner.

Special Permissions - Example

- Restricting access (delete, rename) to files in publicly writable folder only to the owner of the file or directory:
  - `chmod o+t /tmp`
  - `chmod 1777 /tmp`
- Newly created files will have the same GID as this folder, not as the parent process:
  - `chmod g+s /tmp/shared`
- Program will run with UID of the file's owner, not with UID of the parent process:
  - `chmod u+s /bin/ping`

File Attributes

- Attributes in filesystems based on *Ext2* allow further fine-tuning of access to files.
- Available attributes (man chattr):
  - a, append only
  - I, immutable
  - j, journalled
  - s, secure delete (N)
  - S, synchronous write
  - -t no tail merging
  - u, undelete (N)

File Attributes - Example

- List attributes
  - `lsattr`
  - `lsattr ~`
- They can be changed only by root
  - `chattr [-R] +-=[AsacDdIijsTtu] files`
  - `chattr +i /boot/{vmlinux, initrd}*`
  - `chattr +a /var/log/messages`

Quotas

- In order to prevent filling up the filesystem, it is possible to set limits on used disk space for users.
- Hard/soft limit.
- Working with quotas:
  - `quota`
  - `edquota`
  - `quotacheck`
  - `quotastats`
  - `quotaon,quotaoff`

Access Control Lists

- ACL
- In addition to the 3 permission groups this extension allows to define specific permissions for named users and groups.
  - *Owner* – `user::rwx`
  - Named user – `user:name:rwx`
  - Owning group – `group::rwx`
  - Named group – `group:name:rwx`
  - Mask – `mask::rwx`
  - Others – `other::rwx`

Access Control Lists

- List file ACL: `getfacl file`
- Set file ACL: `setfacl`
  - `setfacl -m u:admin:rwx /root`
  - `setfacl -m g:wheel:rx /root`
  - `setfacl -m m::rx /root`
- In Linux ACLs are implemented using extended file attributes.
- Number of ACL entries in ext2 and ext3 is limited to 32.
- Output of `ls` signifies presence of ACL using '+' character.

Capacity Information

- Amount of disk space used by the files or directories can be viewed using
  - `du`
  - `du -sh ~/*`
- Preview of disk usage for mounted filesystems
  - `df`
  - `df -h .`

Loopback Devices

- Mounting file as a device.
- Attaching file to ‘loop’ device
  - `losetup`
  - `losetup /dev/loop0 fs.ext3`
  - `losetup –a`
  - `losetup –d /dev/loop0`
- Encryption (obsolete)
  - `losetup –e blowfish /dev/loop0 en.fs`
- May require loading kernel modules
  - `modprobe cryptoloop`
  - `modprobe blowfish`

Block Device Encryption

- Create encrypted device
  - `man cryptsetup`
  - `cryptsetup luksFormat /dev/hda1`
- Activate/deactivate device
  - `cryptsetup luksOpen /dev/hda1 encdev`
  - `ls /dev/mapper`
  - `cryptsetup luksClose encdev`
- Supports multiple passphrases – LUKS
- Add/remove passphrase
  - `luksAddKey, luksDelKey`

Disk Partitions

- Create, modify, view information
  - `sfdisk -l`
  - `parted -l, parted -i`
  - `fdisk`
- Create filesystem
  - `mkfs`
  - `tune2fs`
  - `fsck`
- Mount filesystem
  - `mount, umount`

Mounting Disk Partitions

- Automatically during start-up
  - `/etc/fstab, man 5 fstab`
- Mount options
  - `noexec`
  - `nosuid`
  - `ro,rw`
  - `user,users`
  - `acl, quota, ...`
- VFAT
  - `uid, gid, umask`

Directory Structure

- All mounted filesystems are organised in a single tree.
- Usual structure: `man hier`
- Configuration files: `/etc/`
- Logs: /`var/log/`
- User data: `/home/`

Related Topics

- Encrypted filesystems
  - `cryptsetup`
- Backup and restore filesystem or its parts
  - `dump/restore, cpio, dd`
- Reliability, redundancy, expanding FS
  - `mdadm, lvm, resize2fs`
- Change root directory for a process
  - `chroot`
- Security enhancements SELinux
  - `getenforce, man selinux`
- Other file systems (ext4, reiserfs, btrfs)

References

- Manual pages
  - https://linux.die.net/man/
- Design and Implementation of the Ext2fs
  - http://e2fsprogs.sourceforge.net/ext2intro.html
- Seecurity Mechanisms and Policies
  - http://www.st.cs.uni-saarland.de/edu/secdesign/mechanisms.ps
- Linux Partition HOWTO
  - http://tldp.org/HOWTO/Partition/index.html