

File management and manipulation

Операционни системи, ФМИ, 2021/2022

Filesystem Hierarchy Standard

- Filesystem standard (FHS)
 - Guiding principles for each area of filesystem
 - Predictable location of files and directories
 - Provides uniformity across multiple Linux distributions
- The Linux Standards Base
 - Aims to allow Linux binaries to run unmodified on multiple Linux distributions
 - Specifies system and library interfaces and environment
 - Incorporates the FHS

Navigating the filesystem

- *absolute* vs. *relative* addressing
- changing and displaying directories (cd, pwd)
- cd (without parameters)
- cd ~george
- cd ~
- cd -
- . and ..

Displaying directory contents

- *human-readable*
- `ls`
- `ls -a` - show all files (including *.hidden* files)
- `ls -l` - long listings
- `ls -d` - show directories, not contents
- `touch foo`
- `mkdir bar`

File {group,}ownership

- each file is owned by a specific UID and GID
- `chown` - change the user (UID) ownership
 - only root can change ownership to another user
 - `chown foo:bar`
- `chgrp` - modify just the group (GID) ownership
 - `chown :bar`
- newly created files will usually be given GID ownership based on the current active group of the person who creates the file

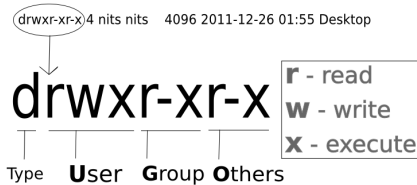
File type & permissions

- type of file
 - - - regular file
 - b - block special file
 - c - character special file
 - d - directory
 - l - symbolic link
 - p - FIFO (named pipe)
 - s - socket
- permission sets
 - user (owner)
 - group (group owner)
 - everyone else (other)
 - symbolic representation `rwxr-xr-x`
 - numeric representation 0755

Recap on logic

- conjunction (AND)
 - $A \wedge B$
 - C-style bitwise conjunction $A \ \& \ B$
- disjunction (OR)
 - $A \vee B$
 - C-style bitwise disjunction $A \ | \ B$
- negation (NOT)
 - $\neg A$
 - C-style bitwise negation $\sim A$
- exclusive disjunction (XOR)
 - $A \oplus B$
 - C-style bitwise exclusive disjunction $A \ \wedge \ B$

File permissions



- r - 100b - 4 - Read
- w - 010b - 2 - Write
- x - 001b - 1 - Execute
- $r \mid w = 6$
- $r \mid x = 5$
- 755, 750, 644, 600

Special permissions

- Set UID upon execution (SUID) 100b
- Set GID upon execution (SGID) 010b
- sticky bit 001b
- numeric representation 4755
- symbolic representation

The diagram illustrates the conversion of symbolic permissions to numeric permissions. It shows two rows of permissions: the top row is 'rwxrwxrwx' and the bottom row is 'rwsrwxrwx'. Above the top row, the octal digits 4, 2, 1 are repeated three times, corresponding to the read, write, and execute bits for each of the three user classes (owner, group, others). Arrows point from the 'x' (execute) bits in the top row to the 's' (set) bits in the bottom row. The first 's' bit is labeled 'SUID' in red. The first three bits of the bottom row ('rws') are underlined and labeled 'USER' in red.

4 2 1 4 2 1 4 2 1

rwxrwxrwx

SUID

rwsrwxrwx

USER

- different behavior for files and directories

Special permissions (cont.)

- SUID and SGID on files
 - an executable with the SUID bit set runs with the security context of the user who owns it, regardless of the executing user
 - SGID
- SGID on directories
 - files or sub-directories created within that directory inherit the group ownership of the SGID directory
- Sticky Bit on directories
 - normally in a directory that is world writable, users can delete each other's files. Setting the sticky bit overrides this behavior

Changing file permissions

- `chmod`
 - numeric notation `chmod 0664 foo.txt`
 - symbolic notation `chmod u=rw,g=rw,o=r foo.txt`
 - `+`, `-`, `=`
- `chmod -R`

umask

- Default permissions for newly created filesystem objects
 - files 666
 - directories 777
- umask
 - defines what permissions to **withhold** from the default permissions
 - display or change umask
 - usually set in the user or system shell dot files
- `effective_perm := default_perm & ~umask`

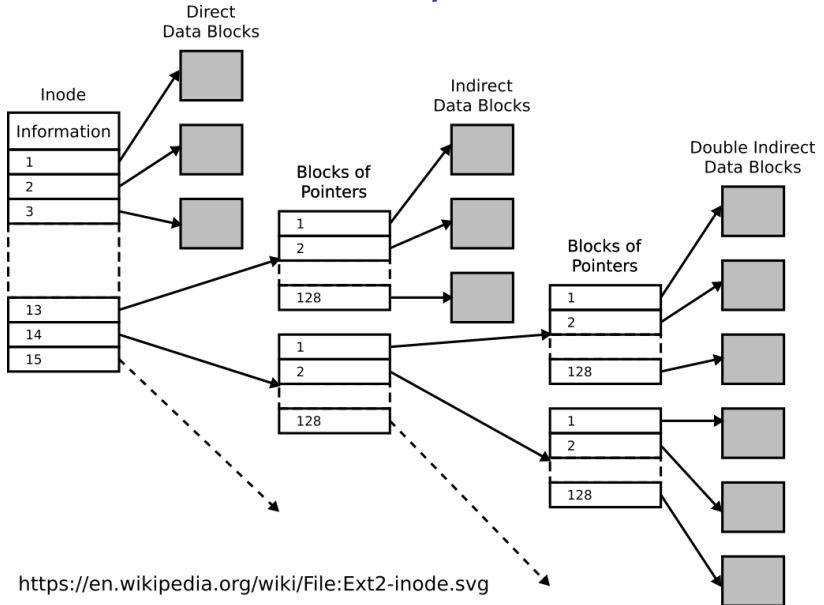
Directory and file manipulation

- `mkdir foo`
 - `mkdir -p foo/bar`
 - `mkdir -m`
- `rmdir`
- `cp`
- `mv`
- `rm`
- `touch - mtime/ctime`

UNIX filesystem structure

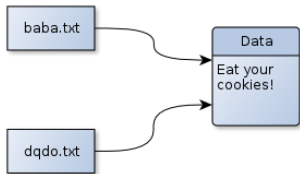
- blocks
- inodes
 - permissions
 - access time, modification time, inode change time
 - owner
 - group
 - size in bytes
 - occupied blocks
 - link count (names of the inode)
 - inode number
- directories (are files that) hold filenames and inodes
- superblock contains filesystem parameters (how many inodes, etc.)

inode pointer structure



Filesystem hard links

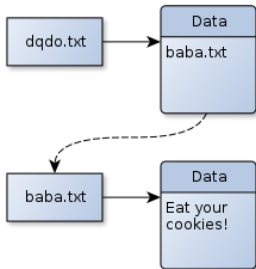
- a directory entry that references the same inode as another directory entry
 - can't span filesystems
 - can't create hard links to non-existent file
 - can't reference directories
 - do not occupy storage space (i.e. blocks)
 - `ln [option]... target link_name`



- `ls -i`

Filesystem symbolic links

- a file that references another file via path and name
 - can reference directories
 - can span filesystems
 - can reference non-existent files
 - `ln -s target link_name`
 - occupy space



- `symlink` / soft link

df, du, stat

- df Report disk space usage per filesystem
 - -h human readable output
 - -i list inode information instead of block usage
 - -T include filesystem type
 - --si use powers of 1000 instead of 1024
 - -P use the POSIX output format
- du Report disk usage per file and directory
 - -h human readable sizes
 - -s summarize, only display total for each argument
 - -x do not include files on a different filesystem
 - --si use powers of 1000 instead of 1024
- stat display file or file system status
 - -L follow links
 - -c --format

File extensions and content

- file extensions are just part of the file name
- some applications may care about extensions
- `file` - reports the type of file by examining the file contents
- `/usr/share/file/magic.mgc`

Displaying text files

- `cat` - concatenate files and print on the standard output
- `more`
- `less`
- `head`
- `tail`
 - `tail -f`
- `-n`

Displaying binary files

- displaying raw binary data may corrupt the display terminal
- `strings` - displays ASCII text inside binary files
- `xxd` - displays HEX and ASCII dump of file
- `clear`

xargs

- build and execute command lines from standard input
- `xargs [options] [command [initial-arguments]]`
- reads items from the standard input
 - delimited by blanks or newlines
- executes the command (`/bin/echo`)
 - one or more times
 - with any initial-arguments
 - followed by items read from standard input
- `-0, --null`
- `-I`
- `-n`

Searching the filesystem

- *machine-readable*
- `find [options] [starting-point] [expression]`
 - global/positional options
 - tests
 - operators
 - `-o, -a` (default)
 - actions
 - `-print` vs `-print0` vs `-printf`
 - `-ls`
 - `-exec`
- `find /foo -name bar -print`

Archiving & compressing

- archiving
 - tar
 - cpio
- compressing
 - compress
 - gzip
 - bzip2
 - lzma
 - xz

Archives with tar

- tar
 - manipulates .tar files (*tarballs*)
 - used for backup and transfer of files
 - creates, extracts or lists the contents of tarballs
 - c, x, t, f, v
 - traditional vs. UNIX-style vs. GNU-style usage
 - `tar cvf foo.tar ./foo/*`
 - GNU tar supports built-in compression methods
 - `-a, --auto-compress`
 - `-J, --xz`
- .tar (*tarball*)
 - records file and directory structure
 - includes metadata about the file: date, timestamps, ownership, permissions, etc.

XZ Utils

- `xz`
- `.xz`
- `unxz` / `xzcat` / `xz -d`
- compression format of choice