File management and manipulation

Операционни системи, ФМИ, 2017/2018

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`

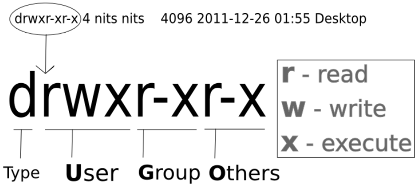
File {group,}ownership

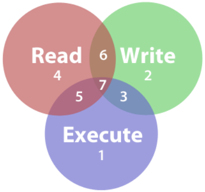
- each file is owned by a specific UID and GID
- `chown` - change the user (UID) ownership
  - only root can change ownership to another user
  - `chown foo:bar`
- `chgrp` - modify just the group (GID) ownership
- newly created files will usually be given GID ownership based on the current active group of the person who creates the file
- `newgrp foo` - log in to a new group
  - newly created files will be owned by the new group
  - users can only change to their own groups
  - root user can change to any group
  - exit to switch back

File 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
- permision sets
  - user (owner)
  - group (group owner)
  - everyone else (other)
  - symbolic representation `rwxr-xr-x`
  - numeric representation `0755`

File permissions (cont.)





- `r – 100b – 4` - Read
- `w – 010b – 2` - Write
- `x – 001b – 1` - Execute

Special permissions

- Set UID upon execution (SUID)
- Set GID upon execution (SGID)
- sticky bit
- different behavior for files and directories

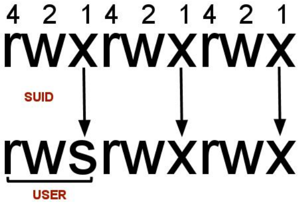


Figure 1: SUID

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

User Private Group (UPG) scheme

- convenient way to share files when working in a group project directory
- each user in their own private group
- `umask 0002`
- set GID of project directory to a commonly shared GID
- set SGID on the project directory

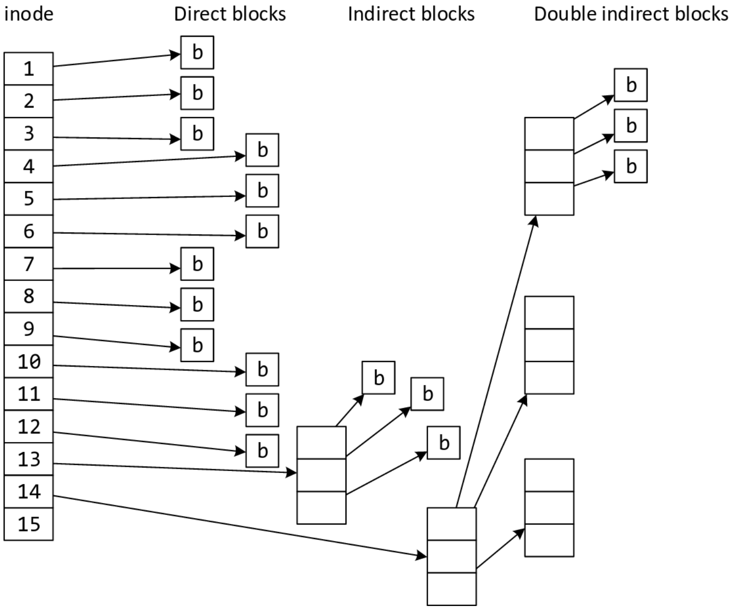
Directory and file manipulation

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

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`

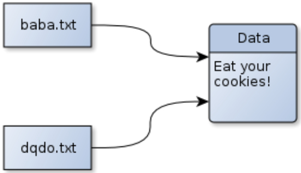
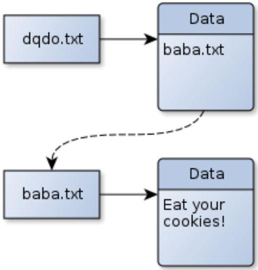


Figure 3: hardlinks

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
- `reset`
- `[Ctrl-J] reset [Ctrl-J]` - if carriage-return fails
- `strings` - displays ASCII text inside binary files
- `xxd` - displays HEX and ASCII dump of file
- `clear`

Searching the filesystem

- machine-readable
- `find [options] [starting-point] [expression]`
- starting point
- criteria
- action
  - `-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
- `.tar` (*tarball*)
  - records file and directory structure
  - includes metadata about the file: date, timestamps, ownership, permissions, etc.

Archives with cpio

- manipulates `.cpio` files
- used as the basis for RPM packages
- doesn't recurse sub-directories, must be passed list of directories
- more robust than tar when media errors encountered
- `-i` input mode, used when feeding a cpio archive into the `cpio` command
- `-o` output mode, used to create cpio archives, which are sent to STDOUT

The gzip compression utility

- `gzip` popular replacement for compress
  - created by the GNU project because of patented algorithms in `compress`
  - default action deletes original after creating new compressed file
  - standard file extension: `.gz`
  - much higher compression ratio than `compress`
- `gunzip` or zcat decompress `.gzip` files
  - `gunzip` decompresses the file on disk (removing the original, compressed file); `zcat` does not
  - `zcat` outputs uncompressed data to STDOUT

The bzip2 compression utility

- `bzip2` typically better compression than `gzip`
- `.bz2`
- `bunzip2` / `bzcat`
- replaces `gzip` as compression format of choice

XZ Utils

- `xz` typically better compression than `bzip2`
- `.xz`
- `unxz` / `xzcat`
- replaces `bzip2` as compression format of choice