

Unix-like Operating Systems

Filesystem. Basic file/directory commands.

Jan Trdlička



Czech Technical University in Prague, Faculty of Information Technology
Department of Computer Systems

1 File system

- Logical FS x physical FS
- Some important directories
- File
- Absolute path x relative path

2 Disk layout

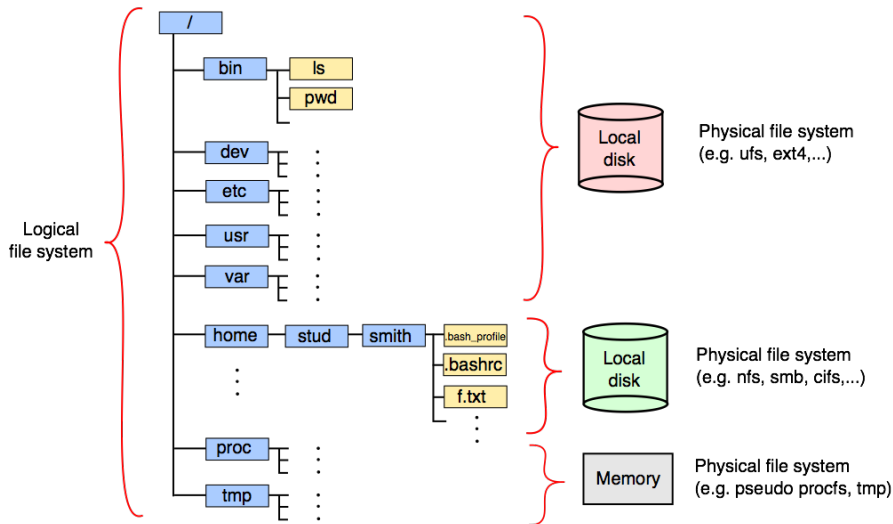
3 FS implementation

- Directory
- Regular file
- Hard link
- Soft link

4 Basic commands

- Directory
- File

File system (FS)



• Logical FS

- It is arranged as a **tree of directories**, that starts at the **root directory** /.
- For users, logical FS appears to be a homogeneous structure.
- User can use commands like `cd`, `pwd`, `ls`, ... to change current working directory, to get info about path to directory, ...
- Logical file system can consist of several physical file systems (use commands `df`, `mount`).

• Physical FS

- Subtree of directories that is saved on one physical devices (e.g. local disk, remote disk, or memory).
 - Disk FS (`ext4`, `ufs`, `btrfs`, `zfs`, ...).
 - Remote FS (`nfs`, `smb`, `cifs`, ..).
 - Pseudo FS (`procfs`, `tmpfs`, `fdfs`, ...).

Some important directories

<code>/home</code>	Directory for user home directories, which store user files.
<code>/tmp</code>	Directory that contains temporary files.
<code>/bin</code>	Symbolic link to the <code>/usr/bin</code> directory.
<code>/sbin</code>	Symbolic link to the <code>/usr/sbin</code> directory.
<code>/usr</code>	Directory that contains system binaries and files.
<code>/usr/bin</code>	Directory that contains contains system exe-cutables and scripts.
<code>/usr/sbin</code>	Contains system administrative exe-cutables and scripts.
<code>/lib</code>	Directory that contains core system libraries.
<code>/dev</code>	Directory that contains special device files.
<code>/etc</code>	Directory that contains administrative and configuration files
<code>/opt</code>	Directory for unbundled application packages.
<code>/var</code>	Directory or file system that contains varying files that are unique to a system but can grow to an arbitrary or variable size.

File = path/name + attributes + data (contents)

- In Unix, the term file is much more general than in other OS.
- The file can represent a regular file, a directory, a special file, a symbolic link, ...
- **Name**
 - Maximal length is file system dependent.
 - Code depends on implementation (ASCII, UTF8,...).
 - Any characters is allowed except of character `/`.
 - **Hidden file**
 - File name beginning with dot.
 - It is not substituted by symbols `*` and `?`.
 - Command `ls` doesn't list them by default (use option `-a`).
 - File names dot (`.`) and double dots (`..`) are reserved for
 - `.` – working directory,
 - `..` – parent directory.

- **Attributes**

- They can be display by command `ls -l`.
- **Type:**

d	directory
-	regular file
l	symbolic link
c	special character device file
b	special block device file

- **Owner:** user and group.
- **Access permissions:** read, write, execution, ACL,...).
- **Times:** time of data modification, time of data access, time of attributes change.
- **Data (contents)**
 - Sequence/stream of bytes stored in data blocks.
- **Access to file**
 - By system calls: `open()`, `read()`, `write()`, `stat()`, ..., `close()`.
 - By OS commands: `less`, `cp`, `rm`, `mv`, `ln`,...

- **Absolute path**

- It is a path beginning in the **root directory /**.
- It contains the hierarchy of directories between root directory / and given file

/home/stud/smith or **/etc/passwd**

- **Working directory**

- Its value is saved in the shell variable **PWD**.
- It can be change by command **cd**.
- Every process can have different working directory.

- **Relative path**

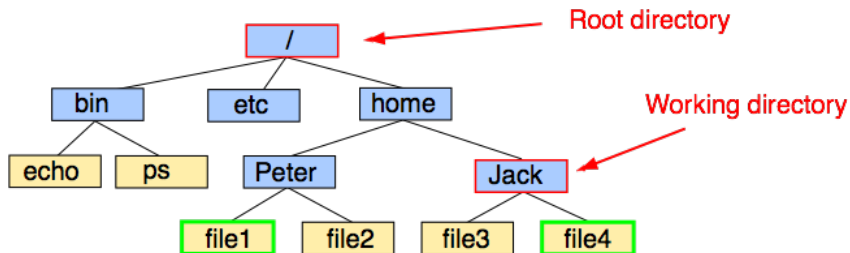
- It is a path relative to the **working directory**.
- It contains the hierarchy of directories between working directory and given file.
- If PWD=/home, then

./stud/smith or **stud/smith** or **../etc/passwd**

- Home directory

- Every user has his home directory.
- During login process working directory is set to the home directory.
- Its value is saved in the shell variable `HOME`.

Path – example



`/home/Peter/file1`

absolute path to the file file1

`../../Peter/file1`

relative path to the file file1

`../Peter/file1`

relative path to the file file1

`/home/Jack/file4`

absolute path to the file file4

`./file4`

relative path to the file file4

`file4`

relative path to the file file4

Disk layout

Disk label + OS loader	Super block	List of free structures (e.g. i-nodes, data blocks,...)	Table of i-nodes	Data blocks (content of files/directories)
------------------------------	-------------	------------------------------------------------------------	---------------------	-----------------------------------------------

- Physical disk layout

- Disk label
 - Table of disk partitions.
 - OS loader (it loads kernel and its modules into the main memory).
- Disk partitions
 - Every partition can contain different file system.

- Unix file system

- Super blok
 - File system specific information.
- List of free structures
- Table of i-nodes
 - It contains file attributes and disk addresses of data blocks where the file content is saved.
- Data blocks.

FS implementation

Commands

```
~> cd /home/peter ; ls -lid .
```

```
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter
```

Table of i-nodes

	File attributes	Data block addresses

236	drwxr-xr-x, 2, peter, users, 4096, Oct 8, 14:58, ...	100,

Data blocks

100

File name	i-node number
.	236
..	235

/home/peter

FS implementation – directory

Commands

~> `cd /home/peter ; ls -lid .`
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter

~> `mkdir dir`

Table of i-nodes

File attributes	Data block addresses
...	...
236 drwxr-xr-x, 3, peter, users, 4096, Oct 8, 14:58, ...	100,
237 drwxr-xr-x, 2, peter, users, 4096, Oct 8, 15:17, ...	152,
...	...

100

File name	i-node number
.	236
..	235
dir	237

/home/peter

152

File name	i-node number
.	237
..	236

/home/peter/dir

Data blocks

FS implementation – regular file

Commands

```
~> cd /home/peter ; ls -lid .  
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter  
  
~> mkdir dir  
  
~> echo "Hello world" > dir/f.txt
```

Table of i-nodes

	File attributes	Data block addresses

236	drwxr-xr-x, 3, peter, users, 4096, Oct 8, 14:58, ...	100,
237	drwxr-xr-x, 2, peter, users, 4096, Oct 8, 15:17, ...	152,
238	-rw-r--r--, 1, peter, users, 12, Oct 8, 15:20, ...	178,

Data blocks

100

File name	i-node number
.	236
..	235
dir	237

/home/peter

152

File name	i-node number
.	237
..	236
f.txt	238

/home/peter/dir

178

Hello world

/home/peter/dir/f.txt

```
ln original_file_name new_file_name
```

- Attributes and data of one file are accessible through several file names.
- It can be created only inside one physical file system.
- It can not point to
 - directory,
 - non existing file.
- After creation of hard link, it is not possible to distinguish between original and new file name.
- Removing
 - i-node and data are removed when the last name are removed.

FS implementation – hard link

Commands

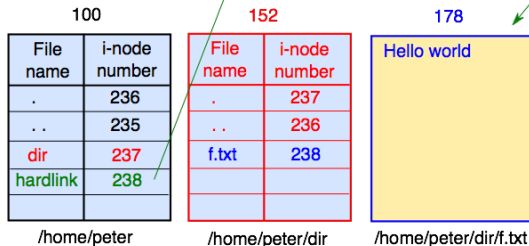
```
~> cd /home/peter ; ls -lid .  
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter  
  
~> mkdir dir  
  
~> echo "Hello world" > dir/f.txt  
  
~> ln dir/f.txt hardlink.txt
```

Table of i-nodes

	File attributes	Data block addresses

236	drwxr-xr-x, 3, peter, users, 4096, Oct 8, 14:58, ...	100,
237	drwxr-xr-x, 2, peter, users, 4096, Oct 8, 15:17, ...	152,
238	-rw-r--r--, 2, peter, users, 12, Oct 8, 15:20, ...	178,

Data blocks




```
ln -s original_file_name new_file_name
```

- Link contains original file name in its data block or in its i-node.
- It is possible create soft link
 - between different physical file systems,
 - to the directory,
 - to non existing files (error during usage of the soft link).
- Some operations are made directly with soft link (`rm`), another ones with the file on which the soft link points (`vi`).

FS implementation – soft link

Commands

```
~> cd /home/peter ; ls -lid .  
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter
```

```
~> mkdir dir
```

```
~> echo "Hello world" > dir/f.txt
```

```
~> ln dir/f.txt hardlink.txt
```

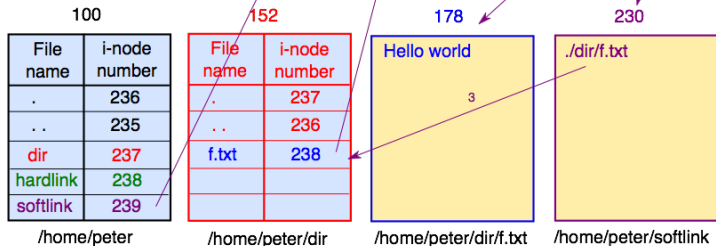
```
~> ln -s ./dir/f.txt softlink.txt
```

Table of i-nodes

	File attributes	Data block addresses

236	drwxr-xr-x, 3, peter, users, 4096, Oct 8, 14:58, ...	100,
237	drwxr-xr-x, 2, peter, users, 4096, Oct 8, 15:17, ...	152,
238	-rw-r--r--, 2, peter, users, 12, Oct 8, 15:20, ...	178,
239	lrwxrwxrwx, 1, peter, users, 13, Oct 8, 15:31, ...	230,

Data blocks



FS implementation – file removing

Commands

```
~> cd /home/peter ; ls -lid .  
236 drwxr-xr-x 2 peter users 4096 Oct 8 15:12 /home/peter
```

```
~> mkdir dir
```

```
~> echo "Hello world" > dir/f.txt
```

```
~> ln dir/f.txt hardlink.txt
```

```
~> ln -s ./dir/f.txt softlink.txt
```

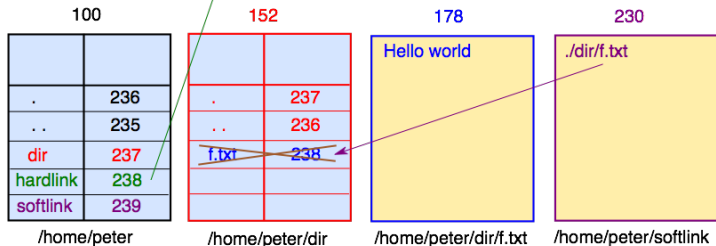
```
~> rm dir/f.txt
```

Table of i-nodes

	File attributes	Data block addresses

236	drwxr-xr-x, 3, peter, users, 4096, Oct 8, 14:58, ...	100,
237	drwxr-xr-x, 2, peter, users, 4096, Oct 8, 15:17, ...	152,
238	-rw-r--r--, 1, peter, users, 12, Oct 8, 15:20, ...	178,
239	lrwxrwxrwx, 1, peter, users, 13, Oct 8, 15:31, ...	230,

Data blocks



Basic commands – directory

<code>cd [dir -]</code>	Change working directory.
<code>ls [aldL] [dir]</code>	List contents of directory
<code>mkdir [-p] dir</code>	Make new diectory.
<code>rm -r dir</code>	Remove dorectory (includinhg its contents).
<code>cp -r dir1 dir2</code>	dir2 doesn't exist: create copy of dir1 by name dir2. dir2 exists: create copy of dir1 in directory dir2 (dir2/dir1).
<code>mv dir1 dir2</code>	dir2 doesn't exist: rename dir1 to dir2. dir2 exists: move dir1 to dir2 (dir2/dir1).

- Beware of recursive copying

`cp -r dir1 dir1`

Basic commands – file

<code>cp f1 f2 dir</code>	Copy files <code>f1</code> and <code>f2</code> to directory <code>dir</code>
<code>cp f1 f2</code>	<code>f2</code> doesn't exist: copy file <code>f1</code> to file <code>f2</code> . <code>f2</code> exists: overwrite file <code>f2</code> by file <code>f1</code>
<code>mv f1 f2</code>	Move/rename file <code>f1</code> to <code>f2</code> .
<code>rm f1</code>	Remove <code>f1</code> .
<code>stat f1</code>	Display attributes of file <code>f1</code> .
<code>file f1</code>	Determine type of file <code>f1</code> .
<code>cat f1 f2</code>	Concatenate and display contents of text files <code>f1</code> and <code>f2</code> .
<code>less f1</code>	Browse or page through a contents of text file <code>f1</code> .
<code>od [-c] f1</code>	Octal dump (print contents of file <code>f1</code>).
<code>strings f1</code>	Display printable strings in file <code>f1</code> .