CSCI 2132
Software Development

Lecture 6:

Pipes; Links and Inodes

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Previous Lecture

- SVN
- wc command, pipelines
- Changing file permissions
- Changing user and group owners
- Redirection
  - standard input, output, and error

Pipes

- Pipes are created by a shell
- Connect standard output of a process to standard input of another process
- Use of pipe metacharacter (`|`)
- A sequence of 'piped' processes is called a *pipeline*
- Example: Count the number of files in a directory
- Solution: `ls | wc -l`

An Approach to Create Pipeline

- Break the problem into subproblems doable by individual commands
- Gradually build and test pipeline if using command line
- Consider sorting files in a directory and printing names of some of them, for example as in:
  - `ls | sort | tail`
  - `ls | sort | tail -n 3 | head -n 1`

Problem Example

The file `/etc/passwd`, is in the following format:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
user1:x:1000:1000:John Doe:/home/user1:/bin/tcsh
```

Fields are separated by colon (‘`:`’)
The last (7th) field is a shell path. Write a command line to get the number of distinct shell paths in the file.
For example, this number would be `3` for the above file.

Solution

```
cut -d":" -f 7 /etc/passwd | sort | uniq | \
                                    wc -l
```

or

```
cut -d":" -f 7 < /etc/passwd | sort | \
                                  uniq | wc -l
```

Inodes and Links

- Each Unix file represented by *inode* (index node) internally
- Each file has a unique inode number
- Inode structure contains the following information:
  - file type
  - permissions
  - owner and group IDs
  - last modification and access time
  - size of the object being stored
  - location of the data on the disk

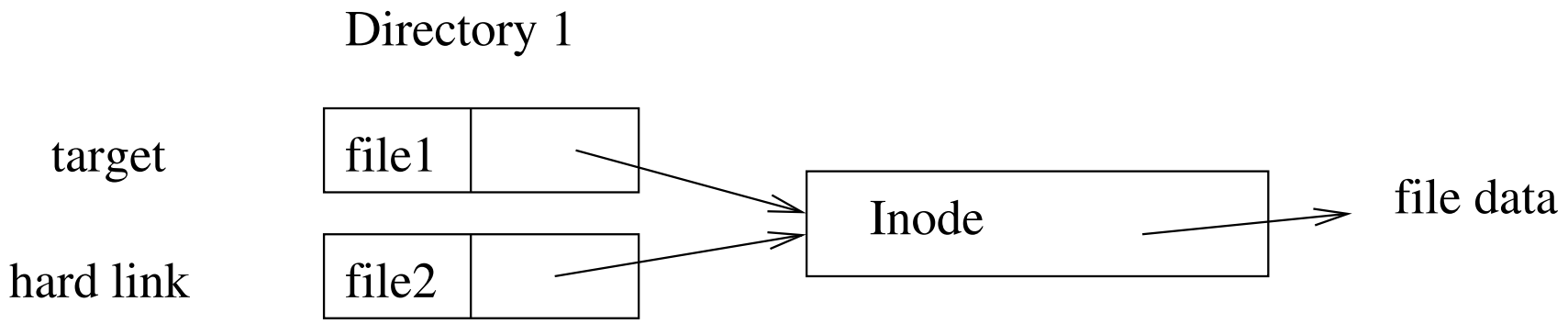
Use of Inodes

- file data is scattered on a disk
- inode contains location information
- one inode table for the system
- a good example of nice unifying design
- `-i` option of `ls` command prints inodes of files
- Example:

```
ls -lid tmp
84492732 drwx------ 2 ...  tmp
```

Hard Links

- Conceptual representation:



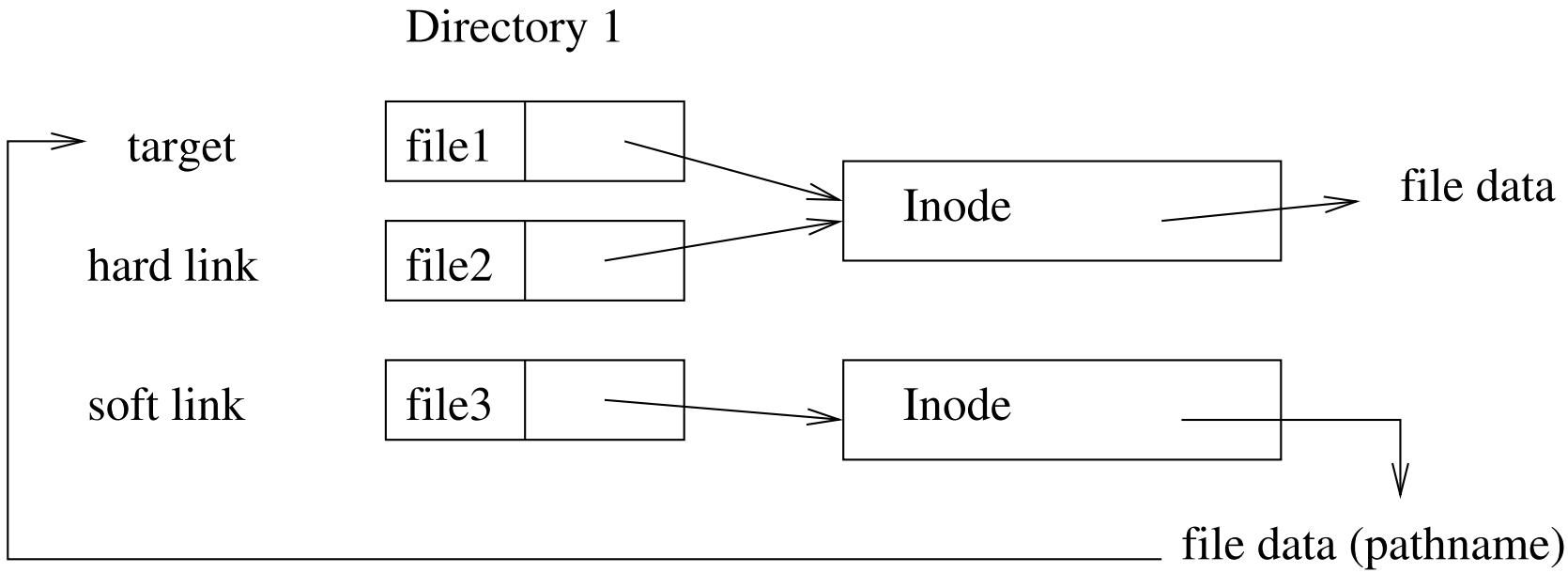
- Command: `ln target linkname`
- Example:
  `ln lab1/HelloWorld.java HelloWorld.java`
- Checking number of hard links: `ls -l`
- Looking up the inode number: `ls -i`

Characteristics of Hard Links

- Programs cannot tell difference between ‘original’ and ‘target’
- Deleting a file just removes a link (unlink system call)
  - Only when number of links is 0, the space is freed
- Limitations:
  - same file system
  - no directories (exceptions possible on some systems)

Soft Links (Symbolic Links)

- Comparison to hard links



- Act as shortcuts

Soft Links Characteristics

- Do not have hardlink restrictions
- A user can link to another user's file/dir
- Advantages and disadvantages vs. hardlinks
  - existence of a 'master' file
  - efficiency issues in time and space
  - special situations (backups)
  - behaviour of some programs (cp)

Wildcards and Regular Expressions

- Similar patterns for string matching
  - . . . but there are differences
- Wildcards or Filename Substitution
  - Used in command line, understood by shell
- Regular Expressions
  - Used with tools such as `grep`