ULI101

Week 08

Week Overview

- File system links
- Hard and symbolic links
- Process management

What is a file system Link?

A link is a pointer to a file.

- This pointer associates a file name with a number called an *i-node* number
- An *i-node* is the control structure for a file (on a UNIX/Linux file system)
- If two file names have the same *i-node* number, they are links to the same file

What is a file system Link?

- Use the command "**ls -i**" to print i-node numbers of each file:

```
[ray@localhost week8]$ ls -i
  32764 lab3a.html    37745 lab3b.html
  37740 lab3.zip
```

- Use the command "**ls -il**" for a long listing

```
[ray@localhost week8]$ ls -il
total 40
  32764 -rw-r--r--  1 ray  ray 1097 Sep 13 08:53 lab3a.html
  37745 -rw-r--r--  1 ray  ray 6582 Sep 13 08:53 lab3b.html
  37740 -rw-rw-r--  1 ray  ray 6218 Sep 14 00:05 lab3.zip
```

inode

What is a file system Link?

There are two kinds of links:

1. Hard Links
2. Soft or Symbolic Links

Hard Links

- Hard link is a reference to the physical data on a file system
- More than one hard link can be associated with the same physical data
- Hard links can only refer to data that exists on the ***same*** file system
- Hard links cannot be created to a directory
- When a file has more than one link, you can remove any one link and still be able to access the file through the remaining links

Hard Links

Example:

- Assume you used "vi" to create a new file, you create the first hard link (**`vi myfile`**)
- To Create the 2nd, 3rd , etc. hard links, use the command:

```
ln myfile link-name
```

Display Hard Link Info

- Create a new file called “myfile”
- Run the command “ls -il” to display the ***i-node number*** and ***link counter***

```
38753 -rw-rw-r--  1 uli  uli    29 Oct 29 08:47 myfile
  ^               ^
  |-- inode #     |-- link counter (one link)
```

Display Hard Link Info

- Create a 2nd link to the same data:

```
ln myfile mylink
```

- Run the command “ls -il”:

```
38753 -rw-rw-r-- 2 uli  uli    29 Oct 29 08:47 myfile
38753 -rw-rw-r-- 2 uli  uli    29 Oct 29 08:47 mylink
 ^                 ^
 |-- inode #       |--link counter (2 links)
```

Add the 3rd Link

- Create a 3rd link to the same data:

```
ln myfile newlink
```

- Run the command “ls -il”:

```
38753 -rw-rw-r-- 3 uli  uli    29 Oct 29 08:47 myfile
38753 -rw-rw-r-- 3 uli  uli    29 Oct 29 08:47 mylink
38753 -rw-rw-r-- 3 uli  uli    29 Oct 29 08:47 newlink
 ^                 ^
 |-- inode #       |--link counter (3 links)
```

Symbolic Links

Also known as soft links or symlinks

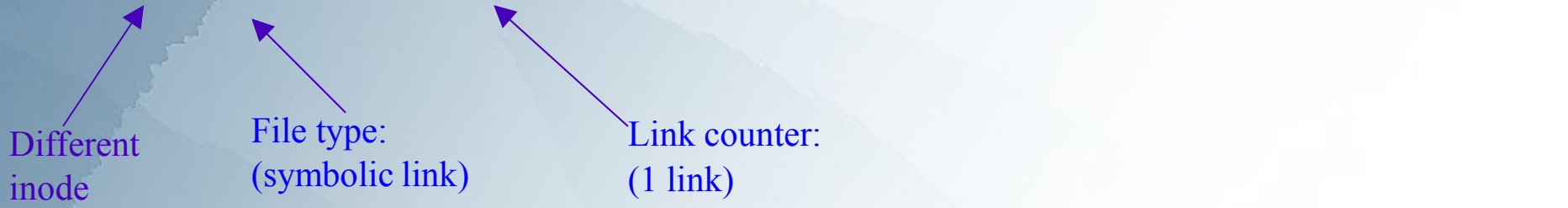
- A Symbolic Link is an indirect pointer to a file – a pointer ***to*** the hard link ***to*** the file
- You can create a symbolic link to a directory
- A symbolic link can point to a file on a different file system
- A symbolic link can point to a nonexistent file (referred to as a "broken link")

Symbolic Links

- To create a symbolic link to the file “myfile”, use

```
ln -s myfile symlink
```

```
[uli@seneca courses] ls -li myfile
44418 -rw-rw-r-- 1 uli uli   49 Oct 29 14:33 myfile
[uli@seneca courses] ln -s myfile symlink
[uli@seneca courses] ls -li myfile symlink
44418 -rw-rw-r--  1 uli uli   49 Oct 29 14:33 myfile
44420 lrwxrwxrwx  1 uli uli    6 Oct 29 14:33 symlink -> myfile
```



Properties of Symbolic Links

- The i-node number is different from the pointed-to file
- The link counter of the new symbolic link file is "1"
- A Symbolic link file does not affect the link counter of the pointed-to file
- The type field of symbolic file contains the letter "l"
- The symbolic link file and the pointed-to file have different status information (file size, last modification time, etc.)
- chmod on the link applies to the actual file, the permissions on the link stay the same

Create Symbolic Link Directory

- The syntax is the same as linking to a file:

```
ln -s target_directory link_directory
```

```
[uli@seneca week8]$ ls -li
38766 drwxrwxr-x 7 uli uli  168 Oct 29 13:32 courses

[uli@seneca week8]$ ln courses mydir
ln: `courses': hard link not allowed for directory
[uli@seneca week8]$ ln -s courses mydir
[uli@seneca week8]$ ls -li
38766 drwxrwxr-x 7 uli uli 168 Oct 29 13:32 courses
44417 lrwxrwxrwx 1 uli uli   7 Oct 29 15:41 mydir -> courses
```

Directory Listing

- To display the contents in a directory, we usually use the command **`ls -l directory_name`**
- Compare the following two commands:

```
[uli@seneca week8]$ ls -l mydir
lrwxrwxrwx  1 uli uli     7 Oct 29 15:41 mydir -> courses

[uli@seneca week8]$ ls -l courses
drwxrwxr-x  2 uli uli    72 Oct 29 11:15 ica101
drwxrwxr-x  2 uli uli    72 Oct 29 11:16 ios110
drwxrwxr-x  2 uli uli   120 Oct 29 11:20 to_do
drwxrwxr-x  2 uli uli    72 Oct 29 11:14 uli101
```

Delete link to a directory

To delete a link to a directory, simply use the **rm** command:

```
[uli@Seneca week8]$ ls -l
drwxrwxr-x 7 uli uli 168 Oct 29 13:32 courses
lrwxrwxrwx 1 uli uli   7 Oct 29 15:41 mydir -> courses

[uli@Seneca week8]$ rm mydir
[uli@Seneca week8]$ ls -l
drwxrwxr-x 7 uli uli 168 Oct 29 13:32 courses
```

UNIX processes

- All programs that are executing on a UNIX system are referred to as processes
- Each process has an owner
- Each process has a unique ID (PID)
- Processes in UNIX can run in:
  - Foreground
  - Background

Process structure

- UNIX processes are hierarchical
- This structure has a root, parents, and children
- Creation of a new process is called forking or spawning
- Parent can fork a child and children can fork their own children
- Processes keep their PID for their entire life
- Usually a parent sleeps when a child is executing
  - The exception is when the child process is executing in the background

Process identification

- ps (process status) command displays snapshot information about processes
- By default, the ps command displays information only about the current terminal (ps -U username shows all)
- The top command provides a continuous update including resource usage

Foreground and background

- Foreground processing:
  - Is the default
  - Takes away the command line until processing is finished
- Background processing:
  - Is invoked by putting the ampersand (&) operator at the end of the command line
  - User gets the command line back immediately
- Both foreground and background processes can be executed on one command line
- Background processes run concurrently (at the same time)

Process suspending

- A foreground job can be suspended (temporarily stopped) by pressing Ctrl+Z
- Stopped jobs can be restarted by using the fg command
  Syntax:

```
        fg
OR      fg %job_number (1,2...)
OR      fg PID
```

  - fg without id/job will bring the last background process to foreground
  - The jobs command will show a list of background/suspended processes

Process restarting

- Restarting in foreground:

  `fg PID` OR

  `fg %job_number`
- Restarting in background:

  `bg PID` OR

  `bg %job_number`

Terminating processes

- Foreground processes can be terminated by using Ctrl+C or can be killed
- Background processes have to be killed unless brought to foreground – then Ctrl+C will work

kill command

- Terminates a process
- One or more processes can be terminated at once
- Regular users can only kill processes they own
- Syntax:

  kill PID OR

  kill %job_number
- In some cases may be ignored by the shell – use kill –9 instead
- pkill command can kill processes based on the program name, for example: pkill firefox