CDOT
SICT AR Meeting Area
People

get involved with CDOT

as a Student
as an Open Source Community Member
as a Company

courses

BTC640
BTH740
BTP300
DPI908
DPS901
DPS905
DPS909
DPS911
DPS914
DPS915
DPS924
DPS931
EAC234
ECL500
GAM531
GAM666
GAM670
GPU610
LUX Program
MAP524
OOP344
OPS235
OPS245
OPS335
OPS345
OPS435
OPS445
OPS535
OPS635
OSD600
OSD700
OSL640
OSL740
OSL840

# ULI101 Week 8

Contents [hide]



## What is a file system Link? [edit]

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? [edit]

```
# print i-node number of each file:
[ray@localhost week8]$ ls -i
32764 lab3a.html 37745 lab3b.html 37740 lab3.zip

#  command "ls -il" for 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
```

## What are file links? [edit]

There are two kinds of file links:

Real World Mozilla
RHT524
SBR600
SEC520
SPO600
SRT210
ULI101

course projects

Course Project List
Potential Course Projects
Contrib Opportunities

links

CDOT
Planet CDOT
FSOSS

Tools

What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Page information

**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. 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 cannot be created, either by root or non-root users, to directories, although older Unix systems are rumoured to have allowed hard link creation to directories by root users only.

**Soft Links**

Soft link, also known as Symlink, is an indirect pointer to a file or directory. Soft links in Linux are similar to shortcuts on Windows. A symbolic link can point to a file on a different file system. It can also point to a nonexistent file or directory, in which case the symbolic link is commonly referred to as a 'broken link'.

## Create hard links and display file link information [edit]

```
# begin with an empty directory and create an empty file,
myfile, in it
$ ls
$ touch myfile
$ ls
myfile

# now show i-node (38753) and link counter (1) of myfile
$ ls -il myfile
38753 -rw-rw-r-- 1 uli uli 29 Oct 29 08:47 myfile

# create a hard link, myhlink, to myfile
$ ln myfile myhlink

# use ls -il to check i-node (38753) and link counter values (2)
$ ls -il myfile myhlink
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 myhlink

# create another hard link, newlink, to myfile
$ ln myfile newlink

# use ls -il to check i-node (38753) and link counter values (3)
$ 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 myhlink
38753 -rw-rw-r-- 3 uli uli 29 Oct 29 08:47 newlink
```

## Create soft links and display file link information [edit]

```
# begin with an empty directory and create an empty file,
myfile2, in it
$ ls
$ touch myfile2
$ ls
myfile2
```

```
# now show i-node (44418) and link counter (1) of myfile2
$ ls -li myfile2
44418 -rw-rw-r-- 1 uli uli 49 Oct 29 14:33 myfile2

# create a symbolic link, symlink, to the file myfile2
$ ln -s myfile2 symlink

# use ls -il to check i-node (44418 and 44410) and link counter
values (1)
$ ls -li myfile2 symlink
44418 -rw-rw-r-- 1 uli uli 49 Oct 29 14:33 myfile2
44410 lrwxrwxrwx 1 uli uli 6 Oct 29 14:33 symlink -> myfile2
```

## Properties of Symbolic Links [edit]

- The i-node number is different from the pointed to file
- The link counter of the new symbolic link file is “1”
- Symbolic link file does not affect the link counter of the pointed to file
- The type field of symblic file contains the letter “l”
- The symbolic link file and the pointed to file have different status information (e.g. file size, last modification time etc.)

## What about links to directories? [edit]

The syntax to creating symlinks to directories is the same as creating soft links to a file, meaning any of the following two methods for creating soft links to directories would work:

- `ln -s target_directory link_directory`
- `ln --symbolic target_directory link_directory`

```
# begin with an empty directory and create an empty directory,
empty_dir, in it
$ ls
$ mkdir empty_dir
$ ls -li
38766 drwxrwxr-x 7 uli uli 168 Oct 29 13:32 empty_dir

# try creating a hard link to empty_dir (watch the error
message)
$ ln empty_dir mydir
ln: 'empty_dir': hard link not allowed for directory

# now try creating a soft link to empty_dir
$ ln -s empty_dir mydir
$ ls -li
38766 drwxrwxr-x 7 uli uli 168 Oct 29 13:32 empty_dir
44417 lrwxrwxrwx 1 uli uli 7 Oct 29 15:41 mydir -> empty_dir

# Compare the following two 'ls -l' commands
$ ls -l mydir
lrwxrwxrwx 1 uli uli 7 Oct 29 15:41 mydir -> empty_dir

$ ls -l empty_dir
```

```
total 0

# to remove a link to a directory, delete it with rm:
$ ls -l
drwxrwxr-x 7 uli uli 168 Oct 29 13:32 empty_dir
lrwxrwxrwx 1 uli uli 7 Oct 29 15:41 mydir -> empty_dir

$ rm mydir
$ ls -l
drwxrwxr-x 7 uli uli 168 Oct 29 13:32 empty_dir
```

### Properties of Symbolic Links to Directories [edit]

- The Symbolic link to a directory has a file type of “l” (the first letter of the permission field).
- The permissions on the link are set to “rwx” for all.
- chmod on the link applies to the actual directory (or file), the permissions on the link stay the same
- Can point to a non-existent directory

## UNIX processes [edit]

Almost everything that is “running” on a UNIX system is referred to as a process. Each process has an owner. Each process has a unique ID (PID). Processes in UNIX can run in the foreground or background.

### Process Structure and Information [edit]

- 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 parent sleeps when a child is executing its task. The exception is when the child process is run in the background
- Process Status ( `ps` ) :: Displays snapshot information about processes. By default, the ps command displays information only about current terminal ( `ps -U username` shows all processes started by username).
- Process Monitoring ( `top` ) :: The top command provides a continuous update including resource usage.

### Foreground and Background Processing [edit]

**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 [edit]

A foreground job can be suspended (temporarily stopped) by pressing `Ctrl+z`. Stopped jobs can be restarted by using the `fg` command. The syntax for `fg` is:

- `fg`
- `fg job_number (1,2...)`
- `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 [edit]

To restart the process in the foreground:

- `fg PID`
- `fg job_number`

To restart the process in the background:

- `bg PID`
- `bg job_number`

### Terminating processes [edit]

- 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 [edit]

Terminates a process. One or more processes can be terminated at once. Regular users can only kill processes they own. The syntax for the `kill` command is:

`kill PID`

- In some cases may be ignored by the shell - use `kill -9 PID` instead.
- `kill -9 0` will terminate all background processes.
- `pkill` command can kill processes based on the program name, for example: `pkill firefox`

## Storage Quotas [edit]

All file systems have finite capacity. Try the `du` and `df` commands to find out more information on disk space usage.

Most regular user accounts have a quota. A quota limits the amount of disk space you can consume on the filesytem. This ensures that a single user cannot interfere with the file system use for others.

The amount of quota and disk usage can be obtained using the `quota` command.

If you want more disk space temporarily - use the `/tmp` directory otherwise if you need more disk space permanently you will need to ask your Linux administrator to increase your quota on that server.

Categories: Pages with syntax highlighting errors | ULI101 | ULI101-2018