Tutorial7: Links / Process Management

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

LINKING FILES / MANAGING PROCESSES [edit]

Main Objectives of this Practice Tutorial [edit]

- Understand the purpose and why links are used in Unix / Linux
- Define the term **inode** number as it relates to a file on Unix / Linux
- Define the terms: **Hard** Link and **Symbolic** Link
- Issue the **ln** command to create hard and symbolic links
- Define and understand the purpose of a **process** in Unix / Linux
- **Run** and **terminate** processes in the foreground and background
- **Display** and **manipulate** background and foreground processes

Tutorial Reference Material [edit]

Course Notes	Concepts / Commands		YouTube Videos
Course Notes: • PDF | PPTX	Links • Hard Links • Symbolic Links Managing Processes • Process Information • Manipulating Processes • Running commands /	Linux Commands • ln • ps • top • fg • bg	Brauer Instructional Videos: • Inodes and Links • Processes and Jobs

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

programs in background with &

- jobs
- kill

KEY CONCEPTS [edit]

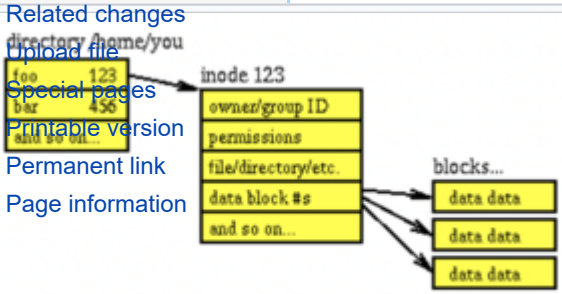
Linking Files [edit]

Links are powerful and add flexibility to Linux filesystems because everything is a file.

There are two types of Linux filesystem links: hard and soft. The difference between the two types of links is significant, but both types are used to solve similar problems. They both provide multiple directory entries (or references) to a single file, but they do it quite differently.

Reference: https://opensource.com/article/17/6/linking-linux-filesystem

inode (index) Number of a File:



Each inode stores the attributes and disk block locations of the object's data.
(Image licensed under cc)

```
[ murray.saul ] ls -li
total 0
1162999961 -rw-r--r-- 1 murray.saul users 0 Jan 31 07:26 a.txt
1164541350 -rw-r--r-- 1 murray.saul users 0 Jan 31 07:26 b.txt
1165743019 -rw-r--r-- 1 murray.saul users 0 Jan 31 07:26 c.txt
2248130583 drwxr-xr-x 2 murray.saul users 6 Jan 31 07:26 mydir
```

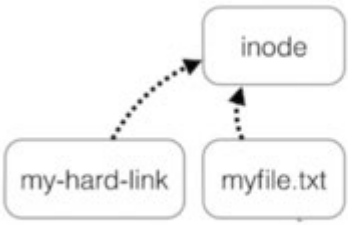
The **inode number** is like a **finger-print**, and usually is **unique** for each file on the Unix / Linux file system.

The inode (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data. File-system object attributes may include metadata (times of last change, access, modification), as well as owner and permission data.

Reference: https://en.wikipedia.org/wiki/Inode

The **inode number** is like a **finger-print**, and usually is **unique** for each file on the Unix / Linux file system.
Referring to the diagram on the far right, issuing the **ls** command with the **-i** option displays the inode number for each file. You can see that each file (whether it is a directory or regular file) has its own unique inode number.

Hard Links:



(Image licensed under cc)
Image manipulated by author]

```
[ murray.saul ] pwd
/home/murray.saul/link-demo1
[ murray.saul ] touch myfile.txt
[ murray.saul ] ln myfile.txt myfile1.hard.lnk
[ murray.saul ] ln myfile.txt myfile2.hard.lnk
[ murray.saul ] ln myfile.txt ~/myfile3.hard.lnk
[ murray.saul ]
[ murray.saul ] ls -li . ~/myfile3.hard.lnk
3261599590 -rw-r--r-- 4 murray.saul users  0 Feb  3 08:39 /home/murray.saul/myfile3.hard.lnk

.:
total 0
3261599590 -rw-r--r-- 4 murray.saul users 0 Feb  3 08:39 myfile.txt
3261599590 -rw-r--r-- 4 murray.saul users 0 Feb  3 08:39 myfile1.hard.lnk
3261599590 -rw-r--r-- 4 murray.saul users 0 Feb  3 08:39 myfile2.hard.lnk
```

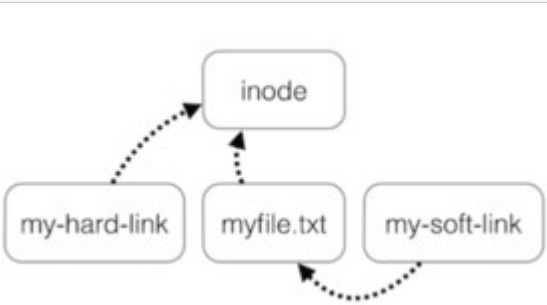
(Image licensed under cc)

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

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

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

Symbolic Links:



(Image licensed under cc)

```
[ murray.saul ] pwd
/home/murray.saul/link-demo2
[ murray.saul ] touch otherfile.txt
[ murray.saul ] ln -s otherfile.txt otherfile1.sym.lnk
[ murray.saul ] ln -s otherfile.txt otherfile2.sym.lnk
[ murray.saul ] ln -s ~murray.saul murray
[ murray.saul ] ls -li
total 0
3267712746 lrwxrwxrwx 1 murray.saul users 17 Feb  3 09:08 murray -> /home/murray.saul
3267712744 -rw-r--r-- 1 murray.saul users  0 Feb  3 09:08 otherfile.txt
3267712742 lrwxrwxrwx 1 murray.saul users 13 Feb  3 09:08 otherfile1.sym.lnk -> otherfile.txt
3267712745 lrwxrwxrwx 1 murray.saul users 13 Feb  3 09:08 otherfile2.sym.lnk -> otherfile.txt
```

(Image licensed under cc)

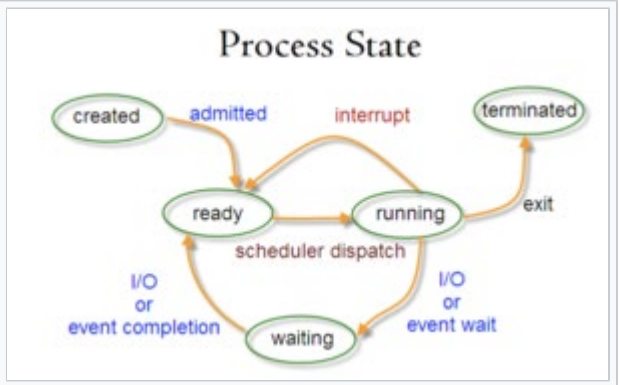
A Symbolic Link is an indirect pointer to a file – a pointer to the hard link to the file You can create a symbolic link to a directory A symbolic link can point to a file on a different file system A symbolic link can point to a nonexistent file (referred to as a "broken link")

Also known as soft links or symlinks

Managing Processes [edit]

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

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



(Image licensed under cc)]

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

INVESTIGATION 1: LINKING FILES [edit]

In this section, you will learn how to ...

Perform the Following Steps:

1. x

In the next investigation, you will ...

INVESTIGATION 2: MANAGING PROCESSES [edit]

In this section, you will learn how to ...

Perform the Following Steps:

1. x

In the next investigation, you will ...

LINUX PRACTICE QUESTIONS [edit]

The purpose of this section is to obtain **extra practice** to help with **quizzes**, your **midterm**, and your **final exam**.

Here is a link to the MS Word Document of ALL of the questions displayed below but with extra room to answer on the document to simulate a quiz:

https://ict.senecacollege.ca/~murray.saul/uli101/uli101_week8_practice.docx

Your instructor may take-up these questions during class. It is up to the student to attend classes in order to obtain the answers to the following questions. Your instructor will NOT provide these answers in any other form (eg. e-mail, etc).

Review Questions:

1. Write a single Linux command to create a hard link called **~/backup/myfile.txt.lnk** to the existing file called **~/myfile.txt**
   Write a single Linux command to display detailed information for those files above displaying their i-node numbers.
   In this case, will the inode numbers for those files above be the same or different?

2. Write a single Linux command to create a symbolic link called **~/shortcuts/murray.saul.lnk** to the existing directory called **~murray.saul**
   Write a single Linux command to display detailed information for those files above displaying their i-node numbers.
   In this case, will the inode numbers for those files above be the same or different?

   What data is contained in the file called **~/shortcuts/murray.saul.lnk**?
   What would be the size of the file called **~/shortcuts/murray.saul.lnk**?

3. Write a single Linux command to run the program called **~/clean.sh** in the background.
   What command would you issue to place the previously issued program in the foreground?
   What command would you issue to confirm that this program is running in the background?
   What key-combination would you issue to send that program again into the background?

4. Write a single Linux command to display running processes in “real-time”.
5. Write a single Linux command to terminal a process that has the following PID: **22384**
6. Use the following diagram to answer the accompanying questions.
   Each of the following questions will use the diagram below and are treated as independent situations.

   ```
   [1] Stopped vim a
   [2]- Stopped vim b
   [3]+ Stopped vim c
   ```

   Write a single Linux command to bring the second-recently process placed in the background into the foreground.

Write a single Linux command to terminate the **job #3**.

7. Create a **table** listing each Linux command, useful options and command purpose for the following Linux commands: **ln** , **ps** , **top** , **fg** , **bg** , **jobs** , **kill**

Tutorial8: Links / Process Management

Category: ULI101