OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

WEEK 8: LESSON 1

LINKING FILES

LESSON 1 TOPICS

Linking Files

- i-nodes
- Hard Links / Demonstration
- Symbolic Links / Demonstration

Perform Week 8 Tutorial

- Investigation 1
- Review Questions (Questions 1 – 2)

Perform Assignment 2

- Section 4: Linking Files and Directories

LINKING FILES

inode (index) Number of a File:

The **i-node number** is like a "**finger-print"** which is **unique** for each file on the Unix / Linux file system.

The i-node is an **index** (**data structure**) that provides information about the file such as if the file is a **directory** or **regular file**, etc.

Referring to the diagram below, issuing the **ls** command using the **-i** option displays the **i-node** number for each file. You can see that each file has its own **unique** *i-node* number in the file system.

```
[ 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
```

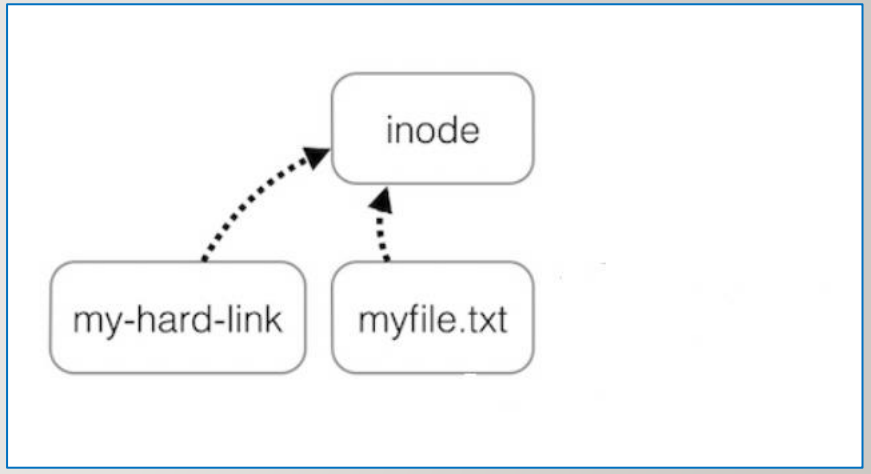
LINKING FILES

Hard Links

A **Hard link** is a **reference** to the **same index** on a file system. It does this by creating a file that **shares the same i-node number** with the other file.

An **advantage** of using hard links is that if one hard link remains (even if original file has been removed), **the data in that hard-linked file is NOT lost**. Also, any change to each file will be reflected in any hard-linked file which is useful for **backups**.

Limitations of hard links are that **they take-up extra space**, you **cannot hard link directories**. Also, you **cannot hard link files from other Unix/Linux servers** (since the i-node number may already be used by the other Unix/Linux server).



LINKING FILES

Hard Links

Examples:

```
touch myfile.txt
ln myfile.txt myfile1.hard.lnk
ln myfile.txt myfile2.hard.lnk
ln myfile.txt ~/backups/myfile.hard.lnk
ls -li myfile*
```

```
[ 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
```

LINKING FILES

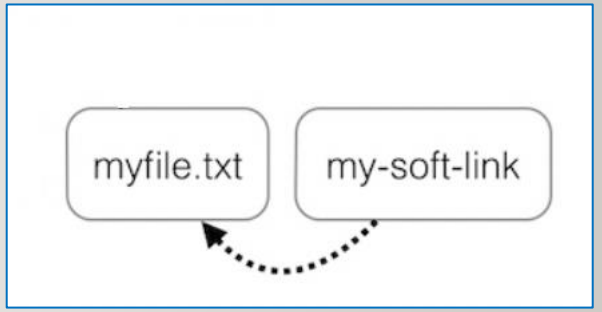
Instructor Demonstration

Your instructor will now demonstrate how to create **Hard Links**.

LINKING FILES

Symbolic Links

A **Symbolic Link** is an **indirect pointer** to a file and are also known as a **soft link** or **symlink**. The symbolic link file contains the **pathname** to the original file.



An **advantage** of using symbolic links is they act as **shortcuts** to other files (in fact, the symbolic linked file only contains the pathname to the original file). Also, you can create symbolic links on **different** Unix/Linux servers, and that you can create symbolic links for **directories**.

A **limitation** of using symbolic links is that they are **NOT good for backup purposes** since a symbolic link can point to a **nonexistent** file (referred to as a "**broken link**").

LINKING FILES

Symbolic Links

Examples:

```
touch otherfile.txt
ln -s otherfile.txt otherfile1.sym.lnk
ln -s otherfile.txt otherfile2.sym.lnk
ln -s otherfile.txt ~/backups/otherfile.sym.lnk
ls -li otherfile*
```

```
[ 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
```

LINKING FILES

Instructor Demonstration

Your instructor will now demonstrate how to create **Symbolic (Soft) links**.

LINKING FILES

Getting Practice

To get practice perform **Week 8 Tutorial:**

- INVESTIGATION 1: LINKING FILES
- LINUX PRACTICE QUESTIONS (Questions **1 – 2**)

OSL640: INTRODUCTION TO OPEN SOURCE SYSTEMS

WEEK 8: LESSON 2

MANAGING PROCESSES
ALIASES AND COMMAND HISTORY

LESSON 2 TOPICS

Processes

- Process Definition / Foreground vs Background Processes
- Running Processes in the Background
- Managing Processes
- Demonstration

Aliases & Command History

- Purpose / Usage / Demonstration

Perform Week 8 Tutorial

- Investigations 2 and 3
- Review Questions (Questions 3 – 8)

MANAGING PROCESSES

Processes Definition

All programs (tasks) that are **running** on a Unix/Linux computer system are referred to as **processes**.

Characteristics of Processes:

- Each process has an **owner**
- Each process has a unique ID (**PID**)
- Processes keep their **PID** for their entire life.
- Usually a parent **sleeps** (i.e. **suspended**) when a **child is running** (the exception is when the child process is running in the background)
- UNIX / Linux processes are **hierarchical**. The process structure can have **children processes**, **great grandchild processes**, etc.

MANAGING PROCESSES

Viewing Process Information

You can issue Linux commands to provide information regarding running processes.
The **ps** (*process status*) command displays a **snapshot** of process information.

The **top** command provides **real-time** status of all running processes (press **ctrl-c** to exit top command)

Linux Command	Purpose
`ps`	Basic listing of processes in current user's terminal, for example: **PID, process names**.
`ps -l`	Detailed listing in current user's terminal for example: **PID**, parent PID (**PPID**), **status**, etc.
`ps -ef`	Detailed listing ALL processes running on entire system.
`ps aux`	Detailed listing of processes for **ALL users** and background running services (i.e. **DAEMONS – background running services**).
`ps –U username`	Basic listing of processes running for a particular **user**.

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to **view** processes.

MANAGING PROCESSES

Foreground vs. Background Processes

Processes in UNIX can run in the **foreground** or **background**

Commands issued from the shell normally run in the **foreground**.

Programs / Commands can be run in the **background** by placing an **ampersand &** after the command.

For example: `command &`

MANAGING PROCESSES

Managing Foreground Processes

Users can **manage processes** to become more **productive** while working in the Unix / Linux Command-line environment.

Below are keyboard shortcuts to manage **foreground** processes.

Linux Command	Purpose
`ctrl-c`	**Terminates** a process running in the **foreground**
`ctrl-z`	Sends a process running in the foreground into the **background**. Process is stopped (suspended) in background and requires **bg** command to run in background.

MANAGING PROCESSES

Managing Background Processes

Below are common Linux commands / **keyboard shortcuts**
to manage **background** processes.

Linux Command	Purpose
`fg`	The **fg** (foreground) command moves a *background* job into the **foreground**. The fg command issued without arguments will place the most recent process in the background to the foreground. *Example:* `fg %job-number`
`bg`	The **bg** utility **resumes suspended jobs** from the current environment. The bg command issued without arguments will run the most recent process that was placed into the background. *Example:* `bg %job-number`
`jobs`	The **jobs** utility displays the status of jobs that were started in the current shell environment

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how
to **manage foreground** and **background** processes.

MANAGING PROCESSES

Terminating Processes

You can use the **kill** command to terminate processes.
You need to be the **owner** of the process to perform this operation.

The **kill** command sends the specified signal to the specified processes or process groups. If no signal is specified, the **SIGTERM** signal **(#15)** is sent.
The default action for this signal is to **terminate** the process.

If the TERM signal does NOT work, you can issue the kill command with the **option -9** (i.e. **SIGKILL, signal #9**).

Examples:

```
kill %jobnumber
kill -9 %jobnumber
kill PID
kill -9 PID
```

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to **terminate** processes.

ALIASES / COMMAND HISTORY

Using Aliases

Using the **alias** command assigns a **nickname** to an existing command
or a series of commands. The **unalias** command is used to remove existent aliases.

Examples:

`alias` (alias command without an argument will display all
the aliases currently set)

```
alias dir=ls
alias lal='ls -al'
alias clearfile='cat /dev/null >'
```

`unalias clearfile` (removes alias **clearfile** from memory)

ALIASES / COMMAND HISTORY

Command History:

The `~/.bash_history` file stores recently executed command lines.

There are several techniques using the `~/.bash_history` file to run previously-issued commands..

Examples:

`<up>` or `<down>`	move to **previous** or **next** command in Bash shell prompt
`fc -l`	display last **16** commands
`history | more`	display all stored commands
`!#`	**re-executes** command by command number (obtained from *history* command)
`!abc`	**re-executes** last command beginning with string "*abc*"

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to use **aliases** and **command history**.

MANAGING PROCESSES / ALIASES / COMMAND HISTORY

Getting Practice

To get practice to help perform **assignment #2**, perform **Week 8 Tutorial:**

- INVESTIGATION 2: MANAGING PROCESSES
- INVESTIGATION 3: ALIASES / COMMAND HISTORY
- LINUX PRACTICE QUESTIONS (Questions **3 – 8**)