

ECS518U Operating Systems

Lab 4: Filesystems (inodes & links)

This exercise is not assessed (but must get it ticked off by the demonstrators)

Aims

The aims of this lab exercise are to:

1. Write Python scripts to create a report about files in a directory.
2. Learn about the two types of links in Unix filesystems and how to distinguish between them.

We assume you know how to run Python scripts from Lab 3. You may find looking at the following sources useful. Note that Python 3.6 is used in the links below as this is the version installed in ITL machines (3.9 or 3.10 versions installed but any differences do not influence lab work).

1. The official Python documentation: <https://docs.python.org/3.10/>
2. Introduction to Programming in Python - an on-line textbook: <https://introcs.cs.princeton.edu/python/home/>
3. Look at the functions available at the **os module** – we will be using them a lot:

<https://docs.python.org/3.10/library/os.html>

Also look at the functions in **os.path** – we will be using them today and in the weeks to come: <https://docs.python.org/3.10/library/os.path.html>

4. MIT Lecture slides and code. MIT have a policy of making some of their material for past modules available to the public.
<https://ocw.mit.edu/courses/electricalengineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/>

Step 1: Basic File Information

A sample script ‘fileInfo0.py’ is provided in the scripting folder. It lists the names of files (and directories) in the directory you run it from, and the number of characters in the name of each file or directory.

☒ **Enhance ‘fileInfo0.py’ to create a script fileInfo1.py which shows:**

- **The name of the file or directory,**
- **the size of the file or directory,**
- **the type (at this step, only whether it is a file or a directory) and**
- **the date last modified/date last accessed/date inode was changed (choose only one, but experiment with all 3 before you choose).**

The output should look similar to the following example (but of course for your files, which will be different than this example):

File Name	Size	Type	Modified	-
-----	-----	-----	-----	-----
fileInfo0.py	2417	file	Dec 03 2022 09:34	
fileInfo0_old.py	1854	file	Dec 03 2022 09:50	
fileInfo1.py	2478	file	Dec 03 2022 10:34	
nbproject	170	dir	Dec 03 2022 08:51	

Create some files to check that your program is working, including directories.

Note: dealing with dates and times is unnecessarily complex at Unix/Linux because of the timestamp that the OS returns: you need to include this with other import statements at the top of your script: `from datetime import datetime`. There are examples in the documentation and elsewhere online. One of the ways (but not the only one) to solve parsing the timestamp is by using the following in an appropriate way in your code:

```
datetime.fromtimestamp(modify_time).strftime('%b %d %Y %H:%M:%S'),
```

where `modify_time` is an example name for the variable that stores the last time the file was modified.

- ☑ Look at **the slides from the Week 4 lecture and see the 3 different times that the inode stores for a file**. Look here for information about how they can be captured in Python: <https://docs.python.org/3.10/library/stat.html>
- ☑ Find how you can capture these times and how you can use them in your code in conjunction with `datetime` as shown previously. Make sure you understand what each one of these three times represents. Answer the relevant questions on the Answer Sheet.

Step 2: Links

Unix allows 'links' to be created. Look at the lecture slides from Week 4 on links, read the 'man ln' page about links, consult other resources you may find online if you wish. There are 2 kinds of links: (i) **hard links** and (ii) **symbolic (or soft) links**.

- ☑ **Create at least one of each and experiment with the fileInfo1.py script to see what Type the link files have. Use the stat shell command to get information about the inode of each of the linked files. What do you observe with regards to inode numbers? Explain the results on the answer sheet.**

WARNING: always use the '-i' option on the `ln` command to avoid deleting a file by mistake.

You can create links either via the `ln` shell command, or from your python code by **using the `os.symlink` and `os.link` Python functions**. See how they work and try both methods of creating links. Be prepared to answer questions about these during the lab.

Step 3: Getting information about links

- ☑ **Create a new script fileInfo2.py, based on fileInfo1.py, to give information about links.**

- For symbolic links, the `os.readlink` function gives the target (or alternatively, `os.path.realpath`). Use one of these functions to **display this information (instead of the date if you want, to keep the line length reasonable – see example below)**.
- **For hard links, you cannot say which file is the link and which is the ‘original’, so instead display a list of synonyms (files with the same name) after the table.** (Note that we are assuming here that the linked files are in the same directory. In general, they do not have to be.)

The output should look similar to the following example (these are my files, your output will be different based on your files of course):

```
File Name          Size    Type    Link          -
-----
fileInfo0.py      2417    file    fileInfo0_old.py
1854      file    fileInfo1.py      2440      file
fileInfo2.py      3383    file
mylink            16      link    -> fileInfo0_old.py
mylink1           1854    file    mylink3           1854
file mylink2      2440    file    nbproject         170
dir
The following files are the same: fileInfo0_old.py, mylink1, mylink3
The following files are the same: fileInfo1.py, mylink2
```

Hints for dealing with hard links: In the example above, both ‘mylink’ and ‘mylink1’, ‘mylink3’ are linked to ‘fileInfo0_old.py’. ‘mylink1’ and ‘mylink3’ are **hard link** which is another name for the same file. mylink is a soft (symbolic) link.

A file is uniquely identified by its ‘inode’ number and you can collect the names for each inode number (in the same directory) using a list and you can then list cases where there are multiple names. You can use `os.lstat(filename).st_ino` to identify the inode number of file `filename`.

Answer Sheet

Step 1: Basic File Information

- Make the changes required to `fileInfo0.py` to get `fileInfo1.py` as per the requirements. Show your code and run the script.

Demonstrate this by running the script and explaining your code during your lab.

```

James@James MINGW64 /e/Coding/University (main)
$ cd e:\\Coding ; /usr/bin/env C:\\Users\\James\\AppData\\Local\\Microsoft\\Windows\\System32\\cmd.exe
../../debugpy\\launcher 59552 -- E:\\Coding\\University\\Year2\\
File Name      Size      Type      Modified
AdventOfCode2022 4096     dir       Dec 27 2022 03:29
ChessTools      4096     dir       Dec 13 2022 18:38
FlappyPong      4096     dir       Dec 29 2022 06:13
GameDev         4096     dir       Dec 14 2022 02:57
leetcode        0        dir       Jan 04 2023 03:27
LowLevel        0        dir       Jan 28 2023 17:23
MiniGolf        0        dir       Jan 28 2023 17:16
OperatingSystems 0        dir       Jan 30 2023 21:17
University      4096     dir       Jan 22 2023 18:15
Visualiser      8192     dir       Jan 01 2023 15:03

```

- b) Describe the **three different types of time information stored in the inode**. Which times are affected if you edit a file? Which times are affected if you change access permissions to a file?

There are three different types of time information stored in the inode:

- **Access Time (atime):** This is the time when a file was last accessed (read or written to). This time is updated whenever the file is opened for reading or writing.
- **Modification Time (mtime):** This is the time when the contents of a file were last modified. This time is updated whenever the file is written to or modified in any way.

Question	Symbolic Link	Hard Link
What file type is each link identified as in your scripts?		
Explain why.		
What do you observe with regards to the inode numbers of the target file and the new symbolic or hard link?	Different inode Number	Same inode Number

Explain why.	Symbolic links point to the target file or directory by its name, so changes made to the target file or directory are immediately visible to all symbolic links that point to it.	Hard links share the same inode with the original file, so changes made to one hard link are immediately visible to all other hard links.
--------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------

- **Status Change Time (ctime):** This is the time when the inode was last changed. This time is updated whenever the file's metadata is modified, such as when the file's ownership or permissions are changed.

If you edit a file, both the modification time (mtime) and the status change time (ctime) will be updated, but the access time (atime) will not be affected. This is because simply opening a file for reading does not change its contents, and therefore does not count as a modification.

If you change the access permissions to a file (e.g., by using the `chmod` command), the status change time (ctime) will be updated, but the access time (atime) and modification time (mtime) will not be affected. This is because changing the access permissions does not modify the contents of the file or change the last time it was accessed.

Step 2: Links

a) Explain the difference between Hard and Symbolic Links.

The main difference between hard and symbolic links is how they point to the underlying file data. Another important difference is that hard links cannot point to directories, whereas symbolic links can. Additionally, hard links can only be created on the same filesystem as the original file, whereas symbolic links can point to files or directories on different filesystems.

b) **What do you think is stored inside a symbolic link file?** What information about the symbolic link file can provide evidence about this (e.g. in the output of a shell command such as `ls -l`) ?

```

james@James:~$ nano b.txt
james@James:~$ ls -lrt
total 8
-rw-r--r-- 1 root root 15 Jan 30 22:00 test.txt
drwxr-xr-x 1 james james 512 Jan 30 22:10 ECS518U
lrwxrwxrwx 1 james james 5 Feb 28 14:20 b.txt -> a.txt
-rw-r--r-- 1 james james 1317 Feb 28 14:21 python.py
-rw-r--r-- 1 james james 1284 Feb 28 14:37 sdad.py
-rw-r--r-- 2 james james 51 Feb 28 14:40 c.txt
-rw-r--r-- 2 james james 51 Feb 28 14:40 a.txt
james@James:~$ cat b.txt
sdasd as
sd
s
d
sad
as
d
sa
das
sd
ass
d
a
updated
james@James:~$ ls -lrti
total 8
239816680157482719 -rw-r--r-- 1 root root 15 Jan 30 22:00 test.txt
14355223812472467 drwxr-xr-x 1 james james 512 Jan 30 22:10 ECS518U
1970324838044823 lrwxrwxrwx 1 james james 5 Feb 28 14:20 b.txt -> a.txt
1688849861334170 -rw-r--r-- 1 james james 1317 Feb 28 14:21 python.py
29273397578101783 -rw-r--r-- 1 james james 1284 Feb 28 14:37 sdad.py
32088147345137159 -rw-r--r-- 2 james james 51 Feb 28 14:40 c.txt
32088147345137159 -rw-r--r-- 2 james james 51 Feb 28 14:40 a.txt
james@James:~$ readlink -f b.txt
/home/james/a.txt
james@James:~$ stat b.txt
  File: b.txt -> a.txt
  Size: 5                Blocks: 0                IO Block: 512        symbolic link
Device: 2h/2d   Inode: 1970324838044823   Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/   james)   Gid: ( 1000/   james)
Access: 2023-02-28 14:20:12.927738000 +0000
Modify: 2023-02-28 14:20:12.927738000 +0000
Change: 2023-02-28 14:20:12.927738000 +0000
Birth: -
james@James:~$

```

- c) **Edit the target file of a hard and of a symbolic link.** Using the `stat` shell command see what effect the edit has on the three time types in the inode of the hard and of the symbolic link. **Explain what you observe.**

B is the symbolic link

C is the hard link

```

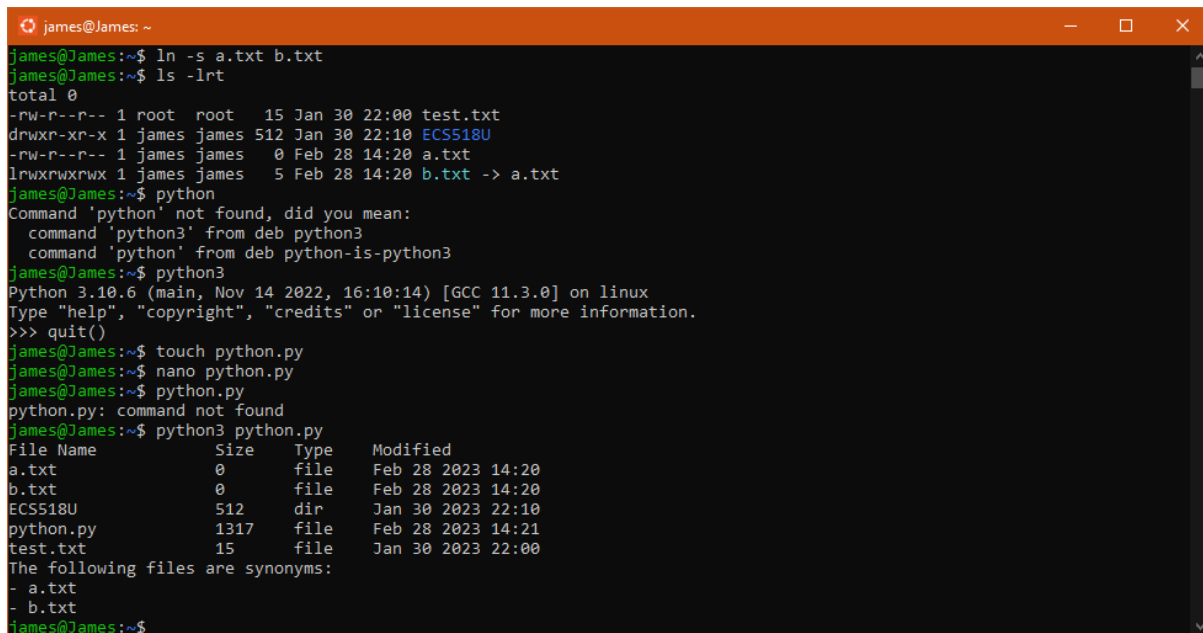
james@James:~$ stat b.txt
  File: b.txt -> a.txt
  Size: 5                Blocks: 0                IO Block: 512        symbolic link
Device: 2h/2d   Inode: 1970324838044823   Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/   james)   Gid: ( 1000/   james)
Access: 2023-02-28 14:20:12.927738000 +0000
Modify: 2023-02-28 14:20:12.927738000 +0000
Change: 2023-02-28 14:20:12.927738000 +0000
Birth: -
james@James:~$ stat c.txt
  File: c.txt
  Size: 51                Blocks: 0                IO Block: 512        regular file
Device: 2h/2d   Inode: 32088147345137159   Links: 2
Access: (0644/-rw-r--r--)  Uid: ( 1000/   james)   Gid: ( 1000/   james)
Access: 2023-02-28 14:40:31.629102500 +0000
Modify: 2023-02-28 14:40:22.545004700 +0000
Change: 2023-02-28 14:40:22.545004700 +0000
Birth: -
james@James:~$

```

Step 3: Getting information about links

Make the changes required to `fileInfo1.py` to get `fileInfo2.py` as per the requirements. Show your code and run the script.

Demonstrate this by running the script and explaining your code during your lab.



```
james@James: ~  
james@James:~$ ln -s a.txt b.txt  
james@James:~$ ls -lrt  
total 0  
-rw-r--r-- 1 root root 15 Jan 30 22:00 test.txt  
drwxr-xr-x 1 james james 512 Jan 30 22:10 ECS518U  
-rw-r--r-- 1 james james 0 Feb 28 14:20 a.txt  
lrwxrwxrwx 1 james james 5 Feb 28 14:20 b.txt -> a.txt  
james@James:~$ python  
Command 'python' not found, did you mean:  
  command 'python3' from deb python3  
  command 'python' from deb python-is-python3  
james@James:~$ python3  
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> quit()  
james@James:~$ touch python.py  
james@James:~$ nano python.py  
james@James:~$ python.py  
python.py: command not found  
james@James:~$ python3 python.py  
File Name      Size      Type      Modified  
a.txt           0         file      Feb 28 2023 14:20  
b.txt           0         file      Feb 28 2023 14:20  
ECS518U        512       dir       Jan 30 2023 22:10  
python.py      1317      file      Feb 28 2023 14:21  
test.txt       15        file      Jan 30 2023 22:00  
The following files are synonyms:  
- a.txt  
- b.txt  
james@James:~$
```