SOFTENG 370 A2 Answers

Name: Jack Chamberlain

UPI (Login): jcha928

Question 1

Output:

```
jack@DESKTOP-Q6QKAUO:~/SE370A2$ ls -l source
total 12
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 one
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 three
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 two

jack@DESKTOP-Q6QKAUO:~/SE370A2$ ls -l mount
total 0
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 one
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 three
-rw-r--r-- 1 jack jack 2 Oct  1 01:59 two
```

Answer:

The output for both reading the source folder contents directly and the mounted file system are the same. This is because a passthrough file system is being used for the mount directory. The passthrough file system is defined in `passthrough.py`. All file system operations defined 'recall' the same operations using the python 'os' module. When recalling, the root file system passed in to the Main function (in this case "source") is used. Therefore all file system operations are redirected to be performed by the OS on the "source" folder.

Question 2

cd mount

Output:

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521110.04, 'st_ctime':
1601521109.25, 'st_gid': 1000, 'st_mode': 16877, 'st_mtime': 1601521109.25,
'st_nlink': 2, 'st_size': 4096, 'st_uid': 1000}
DEBUG:fuse.log-mixin:-> access / (1,)
DEBUG:fuse.log-mixin:<- access None
```

Explanation:

"cd mount" changes the current working directory to the "mount" folder

- getattr / (None,)

  - Gets the attributes of the mount directory (this is "/")
- access / (1,)
  - checks if the mount directory can be accessed by the user (permissions)

cat > newfile

Output and Explanation:

- getattr / (None,)
  - Gets the attributes of the mount directory

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521110.04, 'st_ctime':
1601521109.25, 'st_gid': 1000, 'st_mode': 16877, 'st_mtime': 1601521109.25,
'st_nlink': 2, 'st_size': 4096, 'st_uid': 1000}
```

- getattr /newfile (None,)
  - attempts to get the attributes of newfile before the file is created, therefore a "FileNotFound" error is thrown
  - this is done to see if the file already exists or must be created

```
DEBUG:fuse.log-mixin:-> getattr /newfile (None,)
DEBUG:fuse.log-mixin:<- getattr "[Errno 2] No such file or directory:
'source/newfile'"
DEBUG:fuse:FUSE operation getattr raised a <class 'FileNotFoundError'>, returning
errno 2.
Traceback (most recent call last):
  File "/home/jack/SE370A2/fuse.py", line 731, in _wrapper
    return func(*args, **kwargs) or 0
  File "/home/jack/SE370A2/fuse.py", line 771, in getattr
    return self.fgetattr(path, buf, None)
  File "/home/jack/SE370A2/fuse.py", line 1024, in fgetattr
    attrs = self.operations('getattr', self._decode_optional_path(path), fh)
  File "/home/jack/SE370A2/fuse.py", line 1240, in __call__
    ret = getattr(self, op)(path, *args)
  File "/home/jack/SE370A2/passthrough.py", line 43, in getattr
    st = os.lstat(full_path)
FileNotFoundError: [Errno 2] No such file or directory: 'source/newfile'
```

- create /newfile (33188,)
  - creates newfile to receive input from terminal (cat)
  - the path is 33188
  - a file descriptor "4" is returned

```
DEBUG:fuse.log-mixin:-> create /newfile (33188,)
DEBUG:fuse.log-mixin:<- create 4
```

- getattr /newfile (4,)
  - get attributes of newfile after its creation
  - passes in the file descriptor "4"

```
DEBUG:fuse.log-mixin:-> getattr /newfile (4,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521307.29, 'st_ctime':
1601521307.29, 'st_gid': 1000, 'st_mode': 33188, 'st_mtime': 1601521307.29,
'st_nlink': 1, 'st_size': 0, 'st_uid': 1000}
```

- flush /newfile (4,)
  - flushes newfile file to disk
  - passes in the file descriptor "4"

```
DEBUG:fuse.log-mixin:-> flush /newfile (4,)
DEBUG:fuse.log-mixin:<- flush None
```

- getattr / (None,)
  - get attributes of mount directory
  - passes in no file descriptor
- opendir / ()
  - Opens directory stream and returns a pointer to the stream
  - Initial stream position is the first entry, the mount directory
- readdir / (0,)
  - reads the mount directory using the directory stream pointer from opendir
  - a dir object is returned ("<generator object Passthrough.readdir at 0x7fa8aad89dd0>")
  - this dir object is a representation of the next directory in the directory stream
- releasedir / (0,)
  - releases and closes the mount directory that has just been read
- getattr for /one, /three, /two, "/", /newfile
  - getattr is called for all directories read in from the directory stream
    - this gets the attributes of all files in the mount folder, and the mount folder

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521307.29, 'st_ctime':
1601521307.29, 'st_gid': 1000, 'st_mode': 16877, 'st_mtime': 1601521307.29,
'st_nlink': 2, 'st_size': 4096, 'st_uid': 1000}
DEBUG:fuse.log-mixin:-> opendir / ()
DEBUG:fuse.log-mixin:<- opendir 0
DEBUG:fuse.log-mixin:-> readdir / (0,)
DEBUG:fuse.log-mixin:<- readdir <generator object Passthrough.readdir at
0x7fa8aad89dd0>
DEBUG:fuse.log-mixin:-> releasedir / (0,)
DEBUG:fuse.log-mixin:<- releasedir 0
DEBUG:fuse.log-mixin:-> getattr /one (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601470779.13, 'st_ctime':
1601470763.1, 'st_gid': 1000, 'st_mode': 33188, 'st_mtime': 1601470763.117,
'st_nlink': 1, 'st_size': 2, 'st_uid': 1000}
DEBUG:fuse.log-mixin:-> getattr /three (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601470762.89, 'st_ctime':
1601470762.86, 'st_gid': 1000, 'st_mode': 33188, 'st_mtime': 1601470762.873,
'st_nlink': 1, 'st_size': 2, 'st_uid': 1000}
DEBUG:fuse.log-mixin:-> getattr /two (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601470779.13, 'st_ctime':
1601470763.02, 'st_gid': 1000, 'st_mode': 33188, 'st_mtime': 1601470763.022,
'st_nlink': 1, 'st_size': 2, 'st_uid': 1000}
```

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521308.12, 'st_ctime':
1601521307.29, 'st_gid': 1000, 'st_mode': 16877, 'st_mtime': 1601521307.29,
'st_nlink': 2, 'st_size': 4096, 'st_uid': 1000}
DEBUG:fuse.log-mixin:-> getattr /newfile (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521307.29, 'st_ctime':
1601521307.29, 'st_gid': 1000, 'st_mode': 33188, 'st_mtime': 1601521307.29,
'st_nlink': 1, 'st_size': 0, 'st_uid': 1000}
```

hello world

Output:

```
DEBUG:fuse.log-mixin:-> getxattr /newfile ('security.capability',)
DEBUG:fuse.log-mixin:<- getxattr '[Errno 95] Operation not supported'
DEBUG:fuse:FUSE operation getxattr raised a <class 'fuse.FuseOSError'>, returning
errno 95.
Traceback (most recent call last):
  File "/home/jack/SE370A2/fuse.py", line 731, in _wrapper
    return func(*args, **kwargs) or 0
  File "/home/jack/SE370A2/fuse.py", line 906, in getxattr
    ret = self.operations('getxattr', path.decode(self.encoding),
  File "/home/jack/SE370A2/fuse.py", line 1240, in __call__
    ret = getattr(self, op)(path, *args)
  File "/home/jack/SE370A2/fuse.py", line 1124, in getxattr
    raise FuseOSError(ENOTSUP)
fuse.FuseOSError: [Errno 95] Operation not supported
DEBUG:fuse.log-mixin:-> write /newfile (b'hello word\n', 0, 4)
DEBUG:fuse.log-mixin:<- write 11
```

Explanation:

- getxattr not implemented in passthrough.py hence an "Operation not supported" is thrown when calling getxattr /newfile
- concatenates hello world to newfile file using write operation
- 11 bytes is returned

^D (CTRL+D)

Output:

- flush /newfile (4,)
  - flushes "hello world" text in file in memory to the new file on disk
- release /newfile (4,)
  - releases and closes newfile so can be accessed by other processes

```
DEBUG:fuse.log-mixin:-> flush /newfile (4,)
DEBUG:fuse.log-mixin:<- flush None
DEBUG:fuse.log-mixin:-> release /newfile (4,)
DEBUG:fuse.log-mixin:<- release None
```

cd ../

- getattr / (None,)
  - get attributes of new working directory
  - passes in no file descriptor

Output:

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1601521308.12, 'st_ctime':
1601521307.29, 'st_gid': 1000, 'st_mode': 16877, 'st_mtime': 1601521307.29,
'st_nlink': 2, 'st_size': 4096, 'st_uid': 1000}
```

fusermount -u mount

Output:

- unmounts passthrough file system from "mount folder"
- file system is destroyed
- destroy / ()
  - called upon file system destruction

```
DEBUG:fuse.log-mixin:-> destroy / ()
DEBUG:fuse.log-mixin:<- destroy None
```

Contents of source and mount:

Assumption: specific ls commands for checking source and mount directories are not specified in the brief, so assuming only the
content of these directories is sufficient for this answer. Therefore no FUSE output is provided.

```
jack@DESKTOP-Q6QKAUO:~/SE370A2$ ls -l mount
total 0
jack@DESKTOP-Q6QKAUO:~/SE370A2$ ls -l source
total 16
-rw-r--r-- 1 jack jack 11 Oct  1 16:02 newfile
-rw-r--r-- 1 jack jack  2 Oct  1 01:59 one
-rw-r--r-- 1 jack jack  2 Oct  1 01:59 three
-rw-r--r-- 1 jack jack  2 Oct  1 01:59 two
```

Note, newfile was saved to source containing "hello world" as mount was a passthrough,
therefore the changes in mount were reflected in source.
Mount is empty as the passthrough file system was destroyed.

Question 3

__init__

- creates an empty self.files dictionary to store file attributes. The key is the file path, and the value is a dictionary that stores file attributes.
- creates a self.data dictionary with the key as the file path and the value as the files data (bytes???).

- creates a self.fd file descriptor and initialises it to the default '0'. It will be used as a unique identifier for a file.
- gets current time
- sets file attributes of the root of the file system. The file system root has a creation, modified and accessed times as now (from when the current time was obtained). The root of the file system has 2 links.

getattr

- gets the dictionary of file attributes from the self.files dictionary based on the file path (key). If the file path doesn't exist in self.files, an ENOENT FuseOSError is raised.

readdir

- reads the directory specified by the path parameter and returns a list of contained directories / files, excluding the root directory "/"
- "." and ".." are added to the start of this list (these are the current and previous directory)
- to get the files contained in the directory, self.files dictionary is used

open

- Opens the file specified by the path parameter
- increments the file descriptor and returns its value

create

- This method creates a new file in the location specified by the path parameter
- This is created in memory
    - a new dictionary is added to self.files containing the file attributes of the created file. These attributes are discussed below:
        - the mode parameter is used to specify the file type (st_mode). This is OR-ed with S_IFREG.
        - The size in bytes (st_size) is set to zero as the file has no data.
        - st_nlink is set to 1 to indicate only one hard link.
        - last modification, last status change and last access times are set to the current time.
- The file descriptor is incremented and the new value of it is returned

unlink

- removes the file entry in the self.files dictionary corresponding the the path parameter

write

- Stores the new data in the self.data dictionary
- Updates the file size attribute to reflect the change in size resulting from the write
- returns the size of the data variable (number of items in list)

read

- gets the file data from self.data using the path parameter
- only the data in the range of offset to offset + size is returned