HW1 Report

Computer Network
2013313217 조상연

1. Development Environments

1. Python Version: Python 3.8.1
2. IDE: VS Code
3. OS: Mac OS X 10.15.3

2. Getting Started

➢ 같은 폴더에 복사할 파일과 main.py를 넣고, 해당 폴더로 이동 후 아래 명령어 실행

```
python 2013313217.py

# python3 가 기본이 아니라 오류가 난다면
python3 2013313217.py
```

3. Design

3.1 사용한 모듈

```
import threading
import time
import os
```

3.2 전체 흐름

1. Input File 입력을 받는다.
2. Input File 입력이 exit 일 경우 -> 종료 / 아닐 시 3번 실행
3. New File 입력을 받는다.
4. Worker 함수를 실행하는 Thread를 생성한다. 동시에 1번으로 다시 돌아감
   A. Worker) log.txt에 시작 log를 남긴다.
   B. Worker) Input File에서 New File로 복사 실행 (10KB로 나누어 실행)
   C. Worker) log.txt에 종료 log를 남긴다.

➢ **Thread 생성 코드**

```
def create_worker(input_file, output_file):
    th = threading.Thread(target=worker, args=(input_file, output_file,))
    th.start()
```

➢ **Worker 함수**

```
def worker(input_file, output_file):
    """
    CHUNK_SIZE 단위로 파일 복사
    """

    write_log(output_file, input_file)

    from_file = open(input_file, 'rb')
    to_file = open(output_file, 'ab')

    # 몇번 쪼개어 복사할지 결정
    chunk_time = os.path.getsize(input_file) // CHUNK_SIZE + 1

    while chunk_time:
        to_file.write(from_file.read(CHUNK_SIZE))
        chunk_time -= 1

    write_log(output_file)

    from_file.close()
    to_file.close()
```

➢ **Log 작성 함수**

```
def write_log(to_file, from_file=False):
    """
    log.txt 생성 및 작성
    - from_file_name 유무로 Complete 판단
    """
    log_msg = "{:<8.2f} | Start) [{}] ===> [{}] \n".format(time.time() -
start_time, from_file, to_file) if from_file \
        else "{:<8.2f} | Complete) [{}] \n".format(time.time() - start_time,
to_file)

    global_lock.acquire() # log file 에 대한 Global Lock
    edit_log_file('a', log_msg)
    global_lock.release()
```

- **threading.Lock()** 을 이용하여 log file 접근에 대한 **global_lock** 구현

4. Test

- Test 데이터 생성 및 테스트 자동화
- 직접 입력을 통한 테스트를 통한 재점검

➢ Test Data

```
filelist = [
            ('a_1GB.txt', 'a'*1000*1000*1000), # 1KB
            ('b_10MB.txt', 'b'*10*1000*1000), # 10MB
            ('c_100MB.txt', 'c'*100*1000*1000), # 100 MB
            ('d_1GB.txt', 'd'*1000*1000*1000), # 1GB
            ('e_1MB.txt', 'e'*1000*1000),  # 1MB
            ('f_365MB.txt', 'f'*365*1000*1000), # 365MB
        ]
```

➢ Test Code

```
def setUp():
    edit_log_file('w','') # Init Log File
    for filename, size in filelist:
        create_file(filename, size)

def tearDown():
    # Remove Test Data
    for filename, size in filelist:
        os.remove(filename)

    for filename, size in filelist:
        os.remove(f'copy_'+filename)

def test_main():
    user_inputs = []
    for filename, size in filelist:
        user_inputs.append((filename, 'copy_'+filename))

    # Input
    for input_file, new_file in user_inputs:
        create_worker(input_file, new_file)
        time.sleep(0.5)

    time.sleep(10)

    with open('log.txt','r') as f:
        results = f.read()

    print(results)

    # Accurancy Test
```

```
    for input_file, new_file in user_inputs:
        print("Check) {} --> {} ".format(input_file,new_file))
        with open(input_file,'r') as inputF, open(new_file,'r') as newF:
            assert inputF.read(1) == newF.read(1) # 내용 같은지 확인
            print('✔ Input Content: {} == New Content:
{}'.format(inputF.read(1),newF.read(1)))
        # Chekc File Size
        input_file_size = os.path.getsize(input_file)
        output_file_size = os.path.getsize(new_file)
        assert input_file_size == output_file_size
        print('✔ Input Size: {} == New Size: {}'.format(input_file_size,
output_file_size))
        print(); print('=='*30); print()
```

➢ **Test 결과 1**

```
3.43     | Start) [a_1GB.txt] ===> [copy_a_1GB.txt]
3.93     | Start) [b_10MB.txt] ===> [copy_b_10MB.txt]
3.96     | Complete) [copy_b_10MB.txt]
4.43     | Start) [c_100MB.txt] ===> [copy_c_100MB.txt]
4.85     | Complete) [copy_c_100MB.txt]
4.93     | Start) [d_1GB.txt] ===> [copy_d_1GB.txt]
5.43     | Start) [e_1MB.txt] ===> [copy_e_1MB.txt]
5.43     | Complete) [copy_e_1MB.txt]
5.74     | Complete) [copy_a_1GB.txt]
5.93     | Start) [f_365MB.txt] ===> [copy_f_365MB.txt]
7.15     | Complete) [copy_f_365MB.txt]
7.43     | Complete) [copy_d_1GB.txt]
```

Log.txt 결과

```
Check) a_1GB.txt --> copy_a_1GB.txt
✔ Input: a == New: a
✔ Input Size: 1000000000 == New Size: 1000000000

============================================================

Check) b_10MB.txt --> copy_b_10MB.txt
✔ Input: b == New: b
✔ Input Size: 10000000 == New Size: 10000000

============================================================

Check) c_100MB.txt --> copy_c_100MB.txt
✔ Input: c == New: c
✔ Input Size: 100000000 == New Size: 100000000

============================================================

Check) d_1GB.txt --> copy_d_1GB.txt
✔ Input: d == New: d
✔ Input Size: 1000000000 == New Size: 1000000000

============================================================

Check) e_1MB.txt --> copy_e_1MB.txt
✔ Input: e == New: e
✔ Input Size: 1000000 == New Size: 1000000

============================================================

Check) f_365MB.txt --> copy_f_365MB.txt
✔ Input: f == New: f
✔ Input Size: 365000000 == New Size: 365000000

============================================================
```

Test Report

➢ **Test 결과 2**

1. 테스트용 파일

```
7.8G  3 10 00:30 a.zip
 78M  2 14 16:00 b.mov
210M  1 23 08:17 c.mov
```

2. 실행

```
josang-yeon □ ~/2020/_skku/Computer_Network_2020_Spring/Assignment1 □ □ master ● □ python main.py
Input the file name: a.zip
Input the new name: a.copy.zip
Input the file name: b.mov
Input the new name: b.copy.mov
Input the file name: c.mov
Input the new name: c.copy.mov
Input the file name: exit
```

3. 복사된 파일들

```
7.8G  4  5 19:53 a.copy.zip
7.8G  3 10 00:30 a.zip
 78M  4  5 19:53 b.copy.mov
 78M  2 14 16:00 b.mov
210M  4  5 19:53 c.copy.mov
210M  1 23 08:17 c.mov
```

4. Log.txt 결과

```
12.03    | Start) [a.zip] ===> [a.copy.zip]
17.41    | Start) [b.mov] ===> [b.copy.mov]
17.71    | Complete) [b.copy.mov]
22.00    | Start) [c.mov] ===> [c.copy.mov]
22.83    | Complete) [c.copy.mov]
26.48    | Complete) [a.copy.zip]
```

- 7GB 파일의 경우 가장 늦게 되고, 나머지 파일 들이 먼저 복사됨

5. 영상 재생 (정상)