# Assignment1 Report

2016314365 박수헌

1. 개발 환경

Version of OS : Windows 10 Education

IDE : Pycharm

파이썬 버전 : Python 3.8

사용한 라이브러리 : os, time, threading

2. 자료구조와 알고리즘

```
1   import os
2   import time
3   from threading import Thread
4
5   # open log.txt
6   log = open('log.txt', 'w+')
7
8   # slice is 10Kbytes
9   slice = 10000
10
11  current = time.time()
```

1줄 ~ 3줄 : os, time, threading 라이브러리를 import한다.

6줄 : log를 쓸 log.txt 파일을 연다. log.txt 파일이 존재하지 않으면 생성하고, 두 번째 인자를 'w+'로 설정해 기존에 있던 내용을 삭제한 후 새로 쓴다.

9줄 : source file을 10Kbyte씩 반복적으로 읽어 들여야 하므로 slice라는 변수에 10000을 선언했다.

11줄 : 시간 측정을 위해 코드가 처음 돌 때의 시각을 current 변수에 저장했다.

```
14  # defining function that copies and pastes in binary format
15  def binary_copy(fileName, newName):
16      init = time.time() - current
17      log.write('{:<7.2f}Start copying {} to {}\n'.format(init, fileName, newName))
18      file = open(fileName, 'rb')
19      newFile = open(newName, 'wb')
20      while True:
21          bdata = file.read(slice)
22          newFile.write(bdata)
23          if not bdata:
24              break
25      end = time.time() - current
26      log.write('{:<7.2f}{} is copied completely\n'.format(end, newName))
27      newFile.close()
```

위 부분은 binary하게 파일을 읽어 복사해 새로 만들어내는 함수이다.

16줄 : 복사를 시작한 시점의 시각에서 위의 코드를 돌리기 시작한 시간인 current를 빼, 코드를 실행하고 몇 초가 지났는 지의 정보가 포함된 init 변수를 선언한다.

17줄 : log.txt 파일에 파일 이름과 새로운 파일 이름, 복사 시작 시간의 정보를 포함한 log를 쓴다.

18줄 ~ 19줄 : file 변수에 복사할 파일을 'rb' 인자를 이용해 binary 형식으로 읽어 들이고, newFile에 binary 형식으로 붙여넣기 위해 'wb' 인자를 이용해 파일을 생성한다.

20줄 ~ 24줄 : bdata에 binary 형식으로 읽어온 데이터를 slice단위(위에 설정한 값에 의해 10Kbytes)로 newFile에 쓴다. 이를 반복적으로 while문 안에서 실행하다 23줄에 의해, 만약 bdata가 존재하지 않는다면, 즉 복사한 파일이 다 복사가 됐다면 24줄에 의해 break된다.

25 ~ 26줄 : 끝난 시점의 시각에서 코드를 돌리기 시작한 시각을 빼고 그 값을 포함해 파일의 복사가 완료되었다는 log를 log.txt에 써넣는다.

27줄 : 복사된 파일을 닫는다.

```
30  # main loop
31  while True:
32      fileName = input('Input the file name : ')
33      if fileName == 'exit':
34          exit()
35      newName = input('Input the new name : ')
36      th = Thread(target=binary_copy, args=(fileName, newName))
37      th.start()
```

32 ~ 35줄 : 복사할 파일 이름과 새로운 파일 이름을 input 받는다. 복사할 파일 이름을 입력 받았을 때 그 값이 'exit'이면 33~34줄에 의해 코드 실행을 종료한다.

36줄 ~ 37줄 : th라는 thread를 생성한다. 인자에 의해, 이 thread는 binary_copy 함수를 실행하는

thread이며, 그 인자로는 32 ~ 35줄에서 입력 받은 파일 이름과, 새로운 파일 이름이 된다. 그리고 37줄에 의해 그 thread를 실행한다.

31 ~ 37줄 : 이는 while 문으로 33줄에서 'exit'이라는 입력이 들어와 break할 때까지 실행된다. 계속해서 파일 이름과 새로운 파일 이름을 입력 받으면 이를 binary copy하는 thread가 계속 병렬적으로 생성되고 실행된다는 의미이다.

3. 테스트

테스트를 할 때 용량이 아주 큰 파일을 찾을 수 없었기에 위에 설명한 slice값을 1byte로 변경했고 매우 빠르게 콘솔창에 입력했다. 여러 확장자의 파일들의 복사를 실행해 병렬적으로 실행되는지 확인했다.

샘플로 사용한 파일들의 크기 정보는 아래와 같다.

1.mp4 (16,639KB) → 1new.mp4

2.jpeg (2,051KB) → 2new.jpeg

3.mp4 (14,444KB) → 3new.jpeg

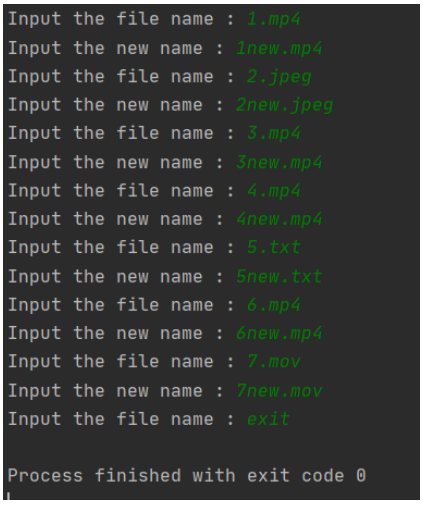
4.mp4 (1,372KB) → 4new.mp4

5.txt (2KB) → 5new.txt

6.mp4 (16,639KB) → 6new.mp4

7.mov (28,333KB) → 7new.mov

순차적으로 이 파일 이름들과 새로운 파일 이름들을 아래와 같이 입력하고 마지막에 exit을 입력하여 프로그램을 종료했다.



아래는 위의 실행이 끝난 후의 log.txt 파일이다.

```
3.55   Start copying 1.mp4 to 1new.mp4
7.76   Start copying 2.jpeg to 2new.jpeg
9.37   2new.jpeg is copied completely
11.16  Start copying 3.mp4 to 3new.mp4
12.43  1new.mp4 is copied completely
14.80  Start copying 4.mp4 to 4new.mp4
15.89  4new.mp4 is copied completely
18.31  Start copying 5.txt to 5new.txt
18.32  5new.txt is copied completely
19.26  3new.mp4 is copied completely
21.91  Start copying 6.mp4 to 6new.mp4
25.07  Start copying 7.mov to 7new.mov
30.47  6new.mp4 is copied completely
35.60  7new.mov is copied completely
```

복사된 파일들인 1new~7new을 확인해본 결과, txt 파일들은 모두 같은 정확한 내용이 복사되었으며, 동영상 파일인 mov, mp4 파일들 모두 영상 내용과 소리까지 정상적으로 복사된 것을 확인할 수 있었다.

위의 로그를 보며 복사가 시작된 시점 순으로 1~7의 파일들을 정렬하면 아래와 같다.

1 → 2 → 3 → 4 → 5 → 6 → 7

콘솔 창에 이 순선대로 입력을 했으니 당연한 결과이다. 하지만 복사가 완료된 시점 순으로 파일들을 정렬하면 아래와 같다.

2 → 1 → 4 → 5 → 3 → 6 → 7

한 파일의 복사가 시작되고 끝나지 않았지만, 다른 파일들의 복사가 시작되었다는 것을 알 수 있다. 즉 복사하는 프로세스는 병렬적으로 이루어졌다.