Spring 2019 Operating Systems
Lab 10: Final Project

0. 과제 수행 전 유의사항

프로그래밍은 자기 손으로 작성해야 한다. 아이디어를 서로 공유하고 도움을 받는 것은 좋으나, 다른 사람의 코드를 그대로 가져다 쓰는 것은 자신에게도 손해이고 다른 사람에게 피해를 끼치는 결과를 낳게되므로 허용하지 않는다. 과제 체크 시 두 사람의 코드가 유사하면, 두 사람 모두에게 과제 점수를 부여할 수 없으니 유의바란다.

1. 준비사항

- <Lab 06: Mini Project> 에서 작성한 프로젝트 폴더로 이동한다. 그 폴더에는 data, sha1.c, sha1.h 등의 파일이 존재할 것이다.
- 다음 명령을 수행하여 프로젝트에 필요한 파일들을 현재 디렉토리로 복사한다.

```
$ cp   -r   /opt/1MB_*.txt   .
```

- 복사가 정상적으로 완료되었다면, 아래와 같은 구조를 갖게될 것이다.

```
/home/s_201xxxxx
        ㄴ project용 폴더
                ㄴ 1MB_A.txt
                ㄴ 1MB_B.txt
                ㄴ data 폴더
                ㄴ sha1.c
                ㄴ sha1.h
                ㄴ . . .
```

- 1MB_*.txt 파일은 정확히 1048576 바이트 (1MB)이다. 수정할 필요는 없다. 만약, 깔끔하게 시작하고 싶다면 폴더를 하나 만들고 그 폴더 내에서 `cp -r /opt/* .` 명령을 수행한다. 참고로 data 폴더는 본 프로젝트에서 필요없으니 삭제해도 된다.

2. 목표

상기의 1MB_A.txt 와 1MB_B.txt 는 파일의 일부분이 서로 다르다. 따라서, 해시값이 다르게 계산될 것이다. 하지만, 해시값이 다르다고 해서 파일의 전체 내용이 다르다는 보장은 없다. 그러므로, 특정 크기로 파일을 잘라서 해시값을 비교하는 방법을 통해 파일 간의 유사도를 산출해 볼 수 있다.

가령, 1MB의 파일을 256KB로 나눈다면 4개의 파일로 나눠질 것이다 (cf. 파일을 나누는 것을 Chunking 기법이라고 한다). 이후, 각각의 조각 파일을 사용하여 해시값을 산출한 후, 대응되는 각각의 파일을 비교하면 조각 파일 중 해시값이 다른 파일이 파악될 것이다. 여기서, 파일을 각각 256KB로 잘랐기 때문에 파일이 4번째에 근접하면 뒷 부분, 1번째에 근접하면 앞 부분이라고 볼 수 있다.

4개로 나눈 각각의 조각 파일 중 한 개만 해시값이 다르다면, 두 파일의 유사도는 (3 / 4) * 100 = 75% 가 될 것이다. 하지만, 이 유사도는 파일의 내용이 얼마나 많이 다른지를 정확히 나타내고 있다고 볼 수 없다. 예를 들어, 1글자만 다른 경우도 75%의 유사도가 출력된다는 점이다.

본 프로젝트에서는 앞선 미니 프로젝트에서의 SHA1 해시값 산출 방법을 활용하여 상기 대상 파일의 유사도를 측정하는 것이 목표이다. 수강생은 아래와 같은 Basic, Intermediate, Advanced의 기준을 보고 구현할 수 있도록 한다. 또한, Appendix를 참조한다.

Basic

- 준비된 1MB_A.txt, 1MB_B.txt 파일을 각각 64KB 단위로 나눠서 저장한다. 이 때, 파일의 확장자는 .001 ~ .016 과 같은 형식을 취하도록 한다. 예) 1MB_A.txt.001, 1MB_A.txt.002, ...
- 나눠진 파일들을 1:1 매칭하여 해시값 산출 코드를 작성하여 해시값을 비교한다. 이후 해시값이 다른 파일명과 그 해시값을 출력한다.
- 판별이 완료되었다면, 나눠진 파일 개수를 기준으로 유사도를 산출한다.

Intermediate

- 파일을 나누는 크기를 16KB로 변경한 후, Basic과 동일하게 유사도를 산출한다.
- Basic에서 산출한 유사도와 본 Intermediate에서 산출하는 유사도가 어떻게 다르며, 그 이유를 서술한다.

Advanced

- Intermediate 에서는 16KB의 청크 중에서 한 개의 청크가 다를 것이다. 그렇다면, 다른 한 개의 청크파일을 1KB로 다시 나눠서 유사도를 산출한다.

3. 제출

제출은 자신의 소스코드, 파일목록, 실행결과, 유사도 등을 워드파일로 추합하여 제출한다.

Appendix. 참고자료

: 프로그램에서 사용되는 파일명은 argv 를 통해 입력받는다. 프로그램의 구동 예는 아래와 같다.

```
kwp@A1409-OSLAB-01:~/project$ ./final_project 1MB_A.txt 1MB_B.txt
Incorrect files
1MB_A.txt.016:  d6f8dfd161eaba044ad54ab6bd7b6b30bb6555b9
1MB_B.txt.016:  9181992a0819b2eed61a008cf965894b1ad3c587
1MB_A.txt:1MB_B.txt files similarity: 94.117647%
```

프로그램의 출력은 달라질 수 있으나, 호출은 ./프로그램명 <파일1> <파일2> 를 고수한다. 참고로 94.117647%는 예시일 뿐 더 높게 구현할 수 있다.

: 파일 디스크립터인 fd와 파일의 이름인 fname을 인자로 받아 청크로 나눠 저장한 후, 청크 갯수를 리턴하는 함수

여기서 CHUNKSIZE가 파일을 나누는 크기이다. 예를 들어, 256KB로 나누고자 한다면 1024 * 256 = 262144 로 값을 정하면 된다.

```c
int read_and_chunk(int fd, char *fname)
{
    int n;
    int chunkfd, nbytes;
    char chunk_name[128];
    char buffer[CHUNKSIZE];

    n = 1;
    while (nbytes = read(fd, buffer, CHUNKSIZE)) {
        sprintf(chunk_name, "%s.%03d", fname, n++);
        if ((chunkfd = open(chunk_name, O_WRONLY | O_CREAT, 0666)) == -1) {
            printf("read_and_chunk(): chunk file create failed\n");
            return -1;
        }
        write(chunkfd, buffer, CHUNKSIZE);
        close(chunkfd);
    }

    return n;
}
```