운영체제 과제#1

학번 : 20190158
이름 : 김민석

1. 문제 기술

1.1 문제

프로그램의 인자로 2개의 입력파일과 하나의 출력파일(결과값을 저장하는 파일) 이름을 전달받은 후, 두 입력 파일에서 행렬을 입력받아 두 행렬의 행렬곱을 구하여 세 번째 이진파일에 저장하는 C프로그램을 구현한다.

1.2 프로그램의 구현 방식

두 입력파일의 크기로 행렬의 크기를 계산해 2차원 배열을 정의하고, 두 입력파일로부터 값을 입력받아 2차원 배열에 값을 저장한다. 그 후 세 번째 2차원 배열을 정의해 두 2차원 배열의 행렬곱을 구해 3번째 2차원 배열에 저장한 후, 3번째 2차원 배열의 값을 출력파일에 작성 후 저장한다.

2. 세부 구현사항

2.1 입력받은 두 파일의 크기를 이용해 행렬의 크기를 구하는 함수

```
int GetMatrixSize(int fd) {
    int size;
    struct stat st;

    if (fstat(fd, &st) < 0) {
        perror("matrix size err");
        exit(1);
    }
    size = st.st_size;
    return (int)sqrt(size / sizeof(double));
}
```

2.2 그 후 파일의 값을 읽어 배열에 저장하는 로직(보고서에는 코드 중복으로 인해 두 번째 파일을 읽는 코드는 생략함.)

```
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
```

```
        Read(ifd1, &arr1[i][j], sizeof(double));
    }
  }
```

2.3 그 후 행렬곱을 구해 res 2차원배열에 값을 저장하는 로직

```
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      res[i][j] = 0.0;
      for (int k = 0; k < N; k++) {
        res[i][j] += arr1[i][k] * arr2[k][j];
      }
    }
  }
```

2.4 res 2차원 배열의 값을 출력파일에 저장하는 파일

```
  for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            Write(ofd, &res[i][j], sizeof(res[i][j]));
        }
  }
```

행렬 곱을 구하는 로직은 인터넷을 참고해 그 알고리즘을 참고함. 또한 행렬을 프로그램에서 구현하기 위해서 2차원 배열을 사용하였으며, 동적인 2차원 배열 크기 지정을 위해 배열의 크기를 구한 후 malloc 함수를 사용해 동적으로 배열의 크기를 할당함.

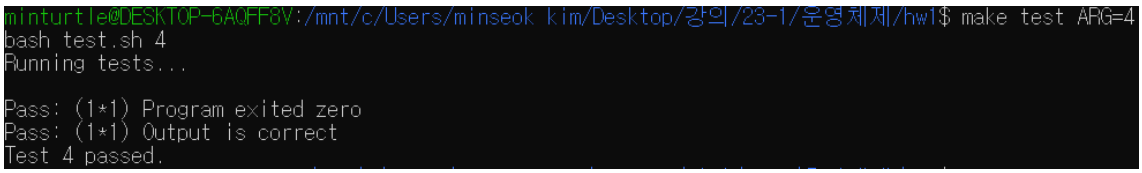
3. 체크 포인트

3.1 ARG 1 테스트 결과(인자 개수 확인)

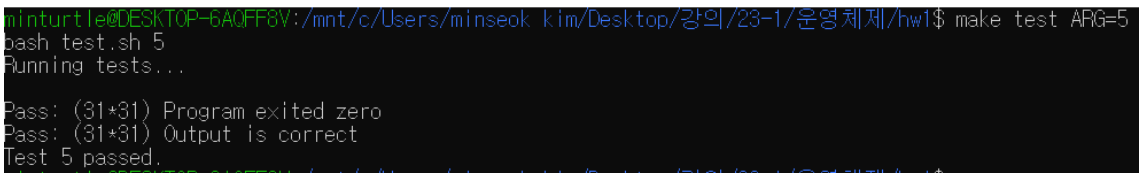
3.2 ARG 2 테스트 결과(정방행렬 여부 확인)

3.3 ARG 3 테스트 결과(동일한 파일 크기 테스트)

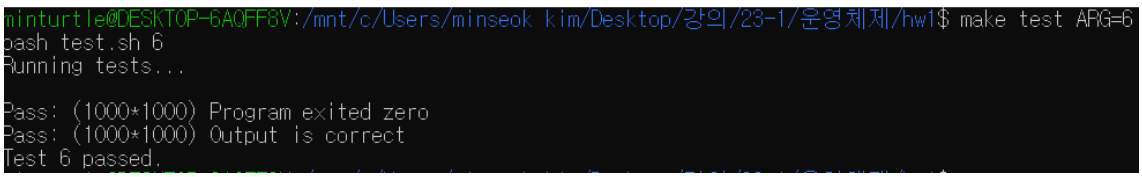
3.4 ARG 4 테스트 결과(1x1 크기 행렬 테스트)



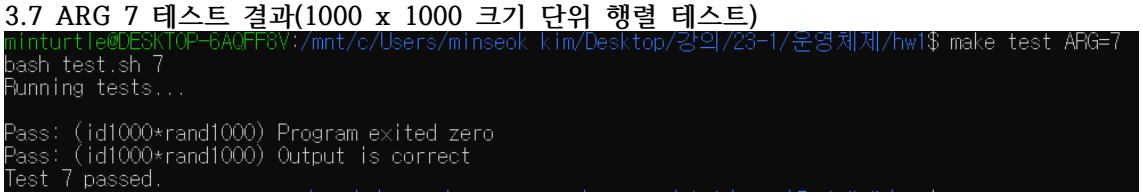
3.5 ARG 5 테스트 결과(31 x 31 크기 행렬 테스트)



3.6 ARG 6 테스트 결과(1000 x 1000 크기 행렬 테스트)



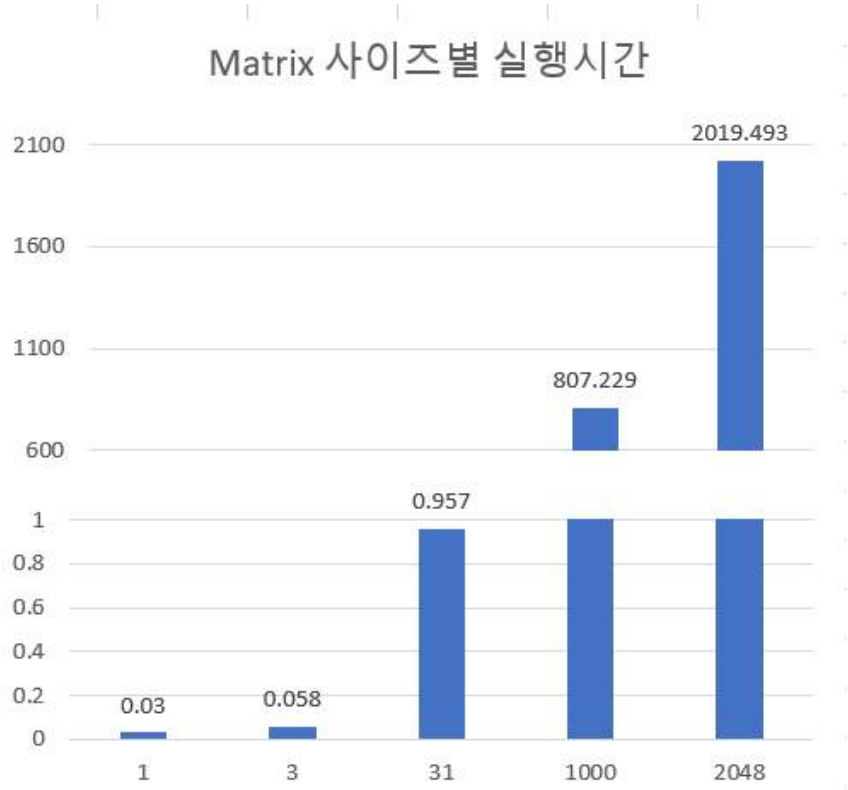
3.7 ARG 7 테스트 결과(1000 x 1000 크기 단위 행렬 테스트)



과제에서 제시된 7가지의 테스트를 모두 통과함으로써 과제에서 요구하는 7가지의 요구사항을 만족함을 확인 할 수 있다.

3.7 2048 x 2048 time 측정 스크린샷

3.8 성능 평가 그래프(y축 단위: 초, x축 : 각 행의 개수)



4. 결론

4.1 전체 요약

정방 행렬이 담긴 두 파일을 입력받아 두 파일의 행렬 값을 읽어 들여 행렬 곱을 구해 다른 결과 파일에 저장하는 프로그램을 작성하였으며, 이때 사용하며 파일 읽기, 파일 쓰기 등의 POSIX API를 사용해 볼 수 있었음.

4.2 후기

POSIX API를 사용해 파일 읽기, 파일 쓰기 등을 개발해 실제 리눅스 환경에서 동작하는 간단한 C 프로그램을 만들어 볼 수 있었으며, 테스트 케이스에 맞춰 개발하는 테스트 주도 개발에 대한 간단한 체험도 해볼 수 있는 기회였다고 생각한다. 또한 메모리 매핑 함수인 mmap을 사용하지 않고 파일 읽/쓰기 기능만 사용해 프로그램을 구현하였는데, 나중에 기회가 된다면 mmap을 사용해 메모리 매핑함수도 한번 사용해보고 싶다.

5. 참고 문헌

5.1 행렬 곱 구하기 알고리즘
https://nate9389.tistory.com/62

5.2 POSIX READ함수
https://man7.org/linux/man-pages/man2/read.2.html

5.3 POSIX WRITE 함수
https://man7.org/linux/man-pages/man2/write.2.html

5.4 fstat으로 파일의 크기 구하기
https://www.techiedelight.com/ko/find-size-of-file-c/