실습 8. 링커와 라이브러리

1. 정적 라이브러리

(1) 배열로 표현된 두 개의 vector 인수에 대해서 vector 내적을 계산하여 반환하는 함수 prodvec를 정의한 소스파일 prodvec.c를 작성하시오.

```
int prodvec(int *x, int *y, int n)
{
        ...
}
```

(2) 강의자료에서 제공한 두 소스 파일 addvec.c, multvec.c을 작성하고, 헤더 파일 vector.h에 세 개의 소스파일에서 작성한 세 개의 함수에 대한 prototype 선언을 작성하시오.

(3) 세 개의 파일을 다음과 같이 컴파일하여 각 소스파일에 대한 object 파일을 생성하고, 파일이 생성되었는지 확인하시오.

```
$ cc -c addvec.c multvec.c prodvec.c     // 컴파일
$ ls -l *.o                              // 확인 (크기도 함께 확인)
```

(4) ar 명령어를 사용하여 정적 라이브러리 파일 libvector.a 생성하시오. (적절한 옵션을 사용하고, 사용한 옵션의 의미를 설명하시오.)

(5) 강의자료의 main2.c를 참고하여 addvec뿐 만 아니라 multvec 함수를 호출하여 벡터 곱을 계산하고, prodvec 함수를 호출하여 벡터 내적을 계산하여 출력하는 main 함수를 작성하시오. (파일 이름 main3.c)

(6) 이 프로그램을 다음과 같이 정적 라이브러리를 사용하여 컴파일한 후 실행시키시오. (컴파일러에서 사용한 옵션에 대한 설명을 각각 적으시오.)

```
$ cc -static -o main3s main3.c -L. -lvector       // 컴파일
$ ./main3s                                        // 실행
```

2. 공유 라이브러리 (적재 시 링킹)

(1) addvec.c, multvec.c, prodvec.c에서 작성한 함수를 공유 라이브러리에 등록하기 위해서 다음과 같이 컴파일하시오(사용한 옵션을 설명하시오.). 그리고 생성된 object 파일의 크기를 확인하여 1번(3)에서 확인한 내용과 비교하시오.

```
$ cc -c addvec.c multvec.c prodvec.c -fpic        // 컴파일
$ ls -l *.o                                       // 확인 (크기도 함께 확인)
```

(2) 다음과 같은 명령어를 실행하여 공유 라이브러리 libvector.so를 만드시오.

```
$ cc -shared -o libvector.so addvec.o multvec.o prodvec.o
```

(3) 다음과 같은 명령어를 사용하여 (1), (2)의 과정을 통합할 수도 있는 데, 이 방법은 어떤 단점이 있는가?

```
$ cc -shared -o libvector.so addvec.c multvec.c prodvec.c -fpic
```

(4) 앞에서 작성한 main3.c를 다음과 같이 공유 라이브러리를 사용하여 컴파일하고, 실행파일의 크기를 정적 라이브러리를 사용했을 경우와 비교하여 차이가 나는 이유를 설명하시오.

```
$ cc -o main3l main3.c ./libvector.so        // 컴파일
$ ls -l main3l main3s                  // 실행파일 크기 확인
```

(5) 공유라이브러리를 사용하는 실행파일을 실행하시오.

```
$ ./main3l
```

(6) 다음과 같이 컴파일하여 실행시켜 보시오. 실행이 잘 되는가?

```
$ cc -o main3c main3.c -L. -lvector
$ ./main3c
```

(7) 다음과 같이 라이브러리 경로를 설정한 후 다시 실행해보시오. 여기서 LD_LIBRARY_PATH의 의미는 무엇인지 알아보시오.

```
$ export LD_LIBRARY_PATH=.
$ ./main3c
```

3. 공유 라이브러리 (실행 시 링킹)

(1) 동적 라이브러리를 실행 시에 적재하여 사용하는 데 필요로 하는 다음 함수의 기능을 조사하시오.

```
dlopen(), dlsym(), dlerror(), dlclose()
```

(2) 강의자료의 dll.c를 참고하여 main3.c와 같은 기능을 수행하는 프로그램을 실행시에 링크를 수행하도록 dll3.c를 작성하시오.

(3) 다음과 같이 컴파일하여 실행하시오. (사용한 옵션에 대해서 설명하시오.)

```
$ cc -rdynamic -o main3r dll3.c -ldl
$ ./main3r
```

(4) 2번 문제와 3번 문제의 실행파일은 동작에 어떠한 차이가 있는가?