< 과제 8 >

2017253041_홍성우

1.

(1)

```
u17253041@linux-ex:~/sysprog/link$ vi prodvec.c
u17253041@linux-ex:~/sysprog/link$ ls
prodvec.c
u17253041@linux-ex:~/sysprog/link$ more prodvec.c
int prodvec(int *x, int *y, int n)
{
        int i;
        int result=0;

        for(i=0; i<n; i++)
                result += x[i]*y[i];

        return result;
}
```

(2)

```
u17253041@linux-ex:~/sysprog/link$ ls -l
total 16
-rw-r--r-- 1 u17253041 user_student  99 Dec  9 04:54 addvec.c
-rw-r--r-- 1 u17253041 user_student 100 Dec  9 04:55 multvec.c
-rw-r--r-- 1 u17253041 user_student 124 Dec  9 04:50 prodvec.c
-rw-r--r-- 1 u17253041 user_student 125 Dec  9 04:59 vector.h
u17253041@linux-ex:~/sysprog/link$ more vector.h
void addvec(int *x, int *y, int *z, int n);
void multvec(int *x, int *y, int *z, int n);
int prodvec(int *x, int *y, int n);
```

(3)

```
u17253041@linux-ex:~/sysprog/link$ cc -c addvec.c multvec.c prodvec.c
u17253041@linux-ex:~/sysprog/link$ ls -l *.o
-rw-r--r-- 1 u17253041 user_student 1336 Dec  9 05:02 addvec.o
-rw-r--r-- 1 u17253041 user_student 1336 Dec  9 05:02 multvec.o
-rw-r--r-- 1 u17253041 user_student 1320 Dec  9 05:02 prodvec.o
u17253041@linux-ex:~/sysprog/link$
```

(4)

```
u17253041@linux-ex:~/sysprog/link$ ar rs libvector.a addvec.o multvec.o prodvec.o
ar: creating libvector.a
```

-정적 라이브러리를 생성하므로 ar을 썼고, replacement와 archive의 index를 위해 rs를 썼고, 이름은 libvector.a(lib+vector)로 정했고, addvec.o, multvec.o, prodvec.o 3가지 파일로 생성했기 때문에 명령어 뒤에 붙여줬다.

(5)

```
u17253041@linux-ex:~/sysprog/link$ vi main3.c
u17253041@linux-ex:~/sysprog/link$ more main3.c
#include <stdio.h>
#include "vector.h"

int x[2] = {1,2};
int y[2] = {3,4};
int z[2];

int main(int argc, char** argv)
{
        addvec(x, y, z, 2);
        printf("z = [%d, %d]\n", z[0], z[1]);
        multvec(x, y, z, 2);
        printf("z = [%d, %d]\n", z[0], z[1]);
        printf("result = %d\n", prodvec(x,y,2));

        return 0;
}
```

(6)

```
u17253041@linux-ex:~/sysprog/link$ cc -static -o main3s main3.c -L. -lvector
u17253041@linux-ex:~/sysprog/link$ ./main3s
z = [4, 6]
z = [3, 8]
result = 11
```

2.

(1)

```
u17253041@linux-ex:~/sysprog/link$ mv addvec.o addvec1.o
u17253041@linux-ex:~/sysprog/link$ mv multvec.o multvec1.o
u17253041@linux-ex:~/sysprog/link$ mv prodvec.o prodvec1.o
```

-편의성을 위해 1번에서 만들어진 object 파일들 이름 뒤에 1을 붙여 파일이름을 변경함

```
u17253041@linux-ex:~/sysprog/link$ cc -Og -c addvec.c multvec.c prodvec.c -fpic
u17253041@linux-ex:~/sysprog/link$ ls -l *.o
-rw-r--r-- 1 u17253041 user_student 1248 Dec  9 07:43 addvec.o
-rw-r--r-- 1 u17253041 user_student 1336 Dec  9 05:02 addvec1.o
-rw-r--r-- 1 u17253041 user_student 1248 Dec  9 07:43 multvec.o
-rw-r--r-- 1 u17253041 user_student 1336 Dec  9 05:02 multvec1.o
-rw-r--r-- 1 u17253041 user_student 1248 Dec  9 07:43 prodvec.o
-rw-r--r-- 1 u17253041 user_student 1320 Dec  9 05:02 prodvec1.o
u17253041@linux-ex:~/sysprog/link$
```

-디버깅과 관련한 옵션으로 -Og를 포함했고 공유 라이브러리이므로 맨뒤에 -fpic를 붙였다. 파일 이름 뒤에 1이 없는 것들이 방금 생성한 object 파일이며 1-(3)에서 생성한 object 파일들보다 크기가 작다.

(2)

```
u17253041@linux-ex:~/sysprog/link$ cc -shared -o libvector.so addvec.o multvec.o prodvec.o
u17253041@linux-ex:~/sysprog/link$
```

```
u17253041@linux-ex:~/sysprog/link$ ls
addvec.c  addvec1.o    libvector.so  main3s     multvec.o   prodvec.c  prodvec1.o
addvec.o  libvector.a  main3.c       multvec.c  multvec1.o  prodvec.o  vector.h
u17253041@linux-ex:~/sysprog/link$
```

(3)

-일반적으로 컴파일해서 공유 라이브러리를 만들때보다 크기가 커질 것 같다.

(4)

```
u17253041@linux-ex:~/sysprog/link$ cc -o main3l main3.c ./libvector.so
u17253041@linux-ex:~/sysprog/link$ ls -l main3l main3s
-rwxr-xr-x 1 u17253041 user_student   8488 Dec  9 08:36 main3l
-rwxr-xr-x 1 u17253041 user_student 845544 Dec  9 05:44 main3s
u17253041@linux-ex:~/sysprog/link$
```

-실행파일 크기는 공유 라이브러리를 사용했을 때가 훨씬 작다. 왜냐하면, 정적 라이브러리를 사용하면 자기 스스로의 것뿐만아니라 다른 파일도 같이 컴파일하여 쓸데없이 크기가 커지기 때문이다.

(5)

```
u17253041@linux-ex:~/sysprog/link$ ./main3l
z = [4, 6]
z = [3, 8]
result = 11
u17253041@linux-ex:~/sysprog/link$
```

(6)

```
u17253041@linux-ex:~/sysprog/link$ cc -o main3c main3.c -L. -lvector
u17253041@linux-ex:~/sysprog/link$ ./main3c
./main3c: error while loading shared libraries: libvector.so: cannot open shared object file:
 No such file or directory
u17253041@linux-ex:~/sysprog/link$
```

-실행이 안된다.

(7)

```
u17253041@linux-ex:~/sysprog/link$ export LD_LIBRARY_PATH=.
u17253041@linux-ex:~/sysprog/link$ ./main3c
z = [4, 6]
z = [3, 8]
result = 11
u17253041@linux-ex:~/sysprog/link$
```

-경로를 직접추가하니 실행됐다. 저 라이브러리 경로는 리눅스의 라이브러리 환경변수 경로이다. 라이브러리 경로 자체를 추가하니 실행된 것 같다.

3.

(1)

dlopen: 동적으로 공유 라이브러리를 담는 기능을 수행함

dlsym: 담은 공유 라이브러리 중에서 해당되는 함수를 가리키는 기능을 수행함

dlerror: 해당되는 함수가 존재하지 않을 때의 기능을 수행함

dlclose: 공유 라이브러리를 해제하는 기능을 수행함

(2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

int x[2] = {1, 2};
int y[2] = {3, 4};
int z[2];

int main(int argc, char **argv)
{
        void *handle;
        void (*addvec)(int *, int *, int *, int);
        void (*multvec)(int *, int *, int *, int);
        int (*prodvec)(int *, int *, int);
        char *error;

        handle = dlopen("./libvector.so", RTLD_LAZY);

        if(!handle){
                fprintf(stderr, "%s\n", dlerror());
                exit(1);
        }

        addvec = dlsym(handle, "addvec");
        multvec = dlsym(handle, "multvec");
        prodvec = dlsym(handle, "prodvec");

        if((error = dlerror()) != NULL){
                fprintf(stderr, "%s\n", error);
                exit(1);
        }

        addvec(x, y, z, 2);
        printf("z = [%d %d]\n", z[0], z[1]);
        multvec(x, y, z, 2);
        printf("z = [%d %d]\n", z[0], z[1]);
        printf("result = %d\n", prodvec(x, y, 2));

        if(dlclose(handle) < 0){
                fprintf(stderr, "%s\n", dlerror());
                exit(1);
        }

        return 0;
}
```

(3)

```
u17253041@linux-ex:~/sysprog/link$ cc -rdynamic -o main3r dll3.c -ldl
u17253041@linux-ex:~/sysprog/link$ ls
addvec.c   dll3.c        main3.c  main3r     multvec.o   prodvec.o
addvec.o   libvector.a   main3c   main3s     multvec1.o  prodvec1.o
addvec1.o  libvector.so  main3l   multvec.c  prodvec.c   vector.h
u17253041@linux-ex:~/sysprog/link$ ./main3r
z = [4 6]
z = [3 8]
result = 11
```

(4)

```
u17253041@linux-ex:~/sysprog/link$ ls -l main*
-rw-r--r-- 1 u17253041 user_student    298 Dec  9 05:43 main3.c
-rwxr-xr-x 1 u17253041 user_student   8488 Dec  9 08:41 main3c
-rwxr-xr-x 1 u17253041 user_student   8488 Dec  9 08:36 main3l
-rwxr-xr-x 1 u17253041 user_student  12776 Dec  9 10:36 main3r
-rwxr-xr-x 1 u17253041 user_student 845544 Dec  9 05:44 main3s
u17253041@linux-ex:~/sysprog/link$
```

-2번 문제의 실행파일은 main3l이고 3번 문제의 실행파일은 main3r이다. 두 실행파일은 같은 결과를 출력하지만 코드의 길이와 크기 모두 2번 문제의 실행파일이 더 작다.