과목 명: 시스템프로그래밍

담당 교수 명: 김 지 환

<<Assignment 3>>

**서강대학교 컴퓨터공학과**

**[20161620]**

**[이수연]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 3

3.1 모듈 이름 : main() 3

3.1.1 기능 3

3.1.2 사용 변수 3

3.2 모듈 이름: int pass1\_load(char\* obf)

3.1.1 기능 3

3.2.2 사용 변수 3

3.3 모듈이름: int pass2\_load(char\* obf) 4

3.3.1 기능 4

3.3.2 사용변수 4

4. 전역 변수 정의 4

4.1 unsigned int progaddr = 0;

unsigned int csaddr;

unsigned int execaddr;

4.2 typedef struct ESTAB{

char csname[MAX];

char symname[MAX];

unsigned int addr;

unsigned int len;

}ESTAB;

ESTAB estab[MAX];

int estab\_i;

4.3 typedef struct EXTREF{

char exname[MAX];

unsigned int refnum;

}EXTREF;

EXTREF extref[MAX];

int extref\_i;

5. 코드

# 프로그램 개요

프로젝트 #1, #2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가한다. 프로젝트 #2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜, 프로젝트 #1에서 구현한 가상 메모리에 올리는 일을 수행한다. 또한 run 명령어와 break point를 통해 disassemble 작업을 수행한다.

# 프로그램 설명

## 프로그램 흐름도

Main()

5. 모든 로드 과정이 끝나면 main에서 load map 출력

1. Main에서 인자 파일 에러 확인하여 에러 없으면 pass1 실행

4. pass2 수행 후 정상 수행 여부 반환

2. pass1 수행 후 정상 수행 여부 반환

Pass1\_load()

3. Main에서 에러 없는 인자파일들에 대하여 pass2 실행

Pass2\_load()

그림 1> 프로그램 흐름도

# 모듈 정의

## 모듈 이름 : main()

### 기능

(프로젝트 3에서 추가된 내용) progaddr 명령어를 수행할 경우 인자가 있을 때 그 인자의 값으로 프로그램내의 주소를 설정해준다. Loader 명령어를 수행할 경우 인자의 파일 순서대로 메모리를 링킹하여 프로젝트 1에서 구현한 가상메모리에 내용을 로드 해 준다. 로드를 수행하는 파일 인자들의 경우 해당 파일이 obj파일인지, 현재 디렉토리에 존재하는 파일인지 등의 여부를 main에서 판단하고 그에 따라 적절한 에레 메시지를 출력하거나 load pass1,2를 실행한다.이때 시작 기준 주소는 progaddr때 설정한 값이 된다. Run과 break point 명령어를 통해 메모리를 disassemble 해준다.

### 사용 변수

DIR\* dir;//디렉토리

struct dirent\* ent;

struct stat buf;

char home[1024];

int i,j;

int isRightCommand=1;

int fileFlag = 0;

signed long temp;

FILE \*fp;

unsigned int id;

char file\_Name\_temp[MAX];

Node\* pnode = NULL;//출력때 이용할 포인터

Table\* new = NULL;

symbolTab tempTab;

int is\_load;//로드가 정상적으로 수행되었는지 반환값을 받음

int obj1\_fileFlag = 1;

int obj2\_fileFlag = 1;

int obj3\_fileFlag = 1;//해당obj파일이 디렉토리에 있는지 검사

unsigned int total\_len;

## 모듈 이름: int pass1\_load(char\* obf)

### 기능

오브젝트 파일을 전달받아 파일을 읽으면서 레코드가 H와 D일 경우 이름과 해당 주소를 저장하여 estab을 만든다. 다음 파일의 주소는 현재 파일 다음이므로 그에 맞게 progaddr을 조정해주고 현재 파일의 progaddr은 csaddr에 별도로 저장한다. Estab에 이미 있는 경우 에러 메시지를 출력한다.

### 사용 변수

FILE \*fp;

int pass1\_return = 1;

char record;

char Name[MAX];

char text[MAX];

unsigned int addr;

unsigned int len;

int i;

## 모듈이름: int pass2\_load(char\* obf)

### 기능

오브젝트 파일을 전달받고 pass1에서 만들어진 estab을 이용하여 링킹과 로딩을 수행한다. T레코드와 M레코드를 읽으면서 relocation을 수행한다. M레코드의 modification이 estab에 없는 경우 에러를 출력한다. 링킹이 완료되면 최종 실제 주소 값을 가상메모리에 올린다. 이를 위해 별도의 변수 execaddr을 선언하여 새로운 주소가 업로드 될 때 저장하고 최종 로드한다.

### 사용변수

int pass2\_return = 1;

FILE\* fp;

int i,j,k,temp\_i;

int find\_flag;//estab에서 찾았는지 여부

int half\_flag;//반비트 사용하는지 여부

int find\_ref = 0;//외부참조 찾았는지 여부

char M\_ch;//modify할때 더할지 뺄지 저장

char tmp\_ch;//half bit 가져올때 버리는 앞비트 저장

unsigned int T\_len, M\_len;

unsigned int T\_addr, M\_addr, fin\_addr;

char record;

char exname[MAX];

char T\_text[MAX], M\_text[MAX], mod\_addr[MAX];

char prog\_name[MAX];

char text[MAX];

unsigned int prog\_start, prog\_len;

unsigned int temp\_addr;

unsigned int mod\_addr\_hex, s\_len;

char put[3];

unsigned int put\_num[MAX],store\_addr[MAX];

# 변수 정의

(프로젝트 3에서 새로 추가되어 유의미하게 쓰인 것을 작성)

typedef struct ESTAB{

char csname[MAX];

char symname[MAX];

unsigned int addr;

unsigned int len;

}ESTAB;

ESTAB estab[MAX];

int estab\_i;

typedef struct EXTREF{

char exname[MAX];

unsigned int refnum;

}EXTREF;

EXTREF extref[MAX];

int extref\_i;

unsigned int progaddr = 0;

unsigned int csaddr;

unsigned int execaddr;

unsigned int progaddr = 0;

unsigned int csaddr;

unsigned int execaddr;

: 프로젝트 3에서 필요한 주소 요소들을 전역 변수로 선언

typedef struct ESTAB{

char csname[MAX];

char symname[MAX];

unsigned int addr;

unsigned int len;

}ESTAB;

ESTAB estab[MAX];

int estab\_i;

: load 과정의 pass1에서 만들어지는 estab를 위한 구조체 와 배열, 전용 인덱스 선언

typedef struct EXTREF{

char exname[MAX];

unsigned int refnum;

}EXTREF;

EXTREF extref[MAX];

int extref\_i;

: load 과정의 pass2에서 M레코드에 이름 대신 reference number를 사용할 경우 그를 위한 구조체와 배열, 전용 인덱스 선언

# 코드

(프로젝트 3에서 새로 추가된 코드 작성)

/\*main의 코드\*/

if(strstr(command,"loader"))

{

if((strlen(obf1) != 0 && !strstr(obf1,".obj")) || (strlen(obf2) != 0 && !strstr(obf2,".obj")) || (strlen(obf3) != 0 && !strstr(obf3,".obj")))

printf("error! : 입력 파일이 object 프로그램이 아닙니다.\n");//입력한 파일중에 .obj확장자가 아닌 파일일 경우 로드 수행X

else

{

estab\_i = 0;

total\_len = 0;

for(i=0; i<MAX; i++)

{

strcpy(estab[i].csname,"\0");

strcpy(estab[i].symname,"\0");

estab[i].addr = 0;

estab[i].len = 0;

}//load 할때마다 초기화

if(strlen(obf1) != 0)

{

obj1\_fileFlag = 0;

dir = opendir("./");

if(dir != NULL)

{

while((ent = readdir(dir)) != NULL)

{

if(!strcmp(ent->d\_name,obf1))//디렉토리에 파일 있으면

{

obj1\_fileFlag = 1;

break;

}

else

continue;

}

if(obj1\_fileFlag == 0)

{

printf("error! : 현재 디렉토리에 해당 object파일이 없습니다.\n");

// return EXIT\_FAILURE;

}

else if(obj1\_fileFlag == 1)

is\_load = pass1\_load(obf1);//정상 파일의 경우 pass1 수행

closedir(dir);

}

else

{

perror("");

return EXIT\_FAILURE;

}

}

if(obj1\_fileFlag!=0 && strlen(obf2) != 0)

{

obj2\_fileFlag = 0;

dir = opendir("./");

if(dir != NULL && (strstr(obf2,".obj")))

{

while((ent = readdir(dir)) != NULL)

{

if(!strcmp(ent->d\_name,obf2))//디렉토리에 파일 있으면

{

obj2\_fileFlag = 1;

break;

}

else

continue;

}

if(obj2\_fileFlag == 0)

{

printf("error! : 현재 디렉토리에 해당 object파일이 없습니다.\n");

// return EXIT\_FAILURE;

}

else if(obj2\_fileFlag == 1)

is\_load = pass1\_load(obf2);

closedir(dir);

}

else

{

perror("");

return EXIT\_FAILURE;

}

}

if(obj1\_fileFlag!=0 && obj2\_fileFlag!=0 && strlen(obf3) != 0)

{

obj3\_fileFlag = 0;

dir = opendir("./");

if(dir != NULL && (strstr(obf1,".obj")))

{

while((ent = readdir(dir)) != NULL)

{

if(!strcmp(ent->d\_name,obf3))//디렉토리에 파일 있으면

{

obj3\_fileFlag = 1;

break;

}

else

continue;

}

if(obj3\_fileFlag == 0)

{

printf("error! : 현재 디렉토리에 해당 object파일이 없습니다.\n");

// return EXIT\_FAILURE;

}

else if(obj3\_fileFlag == 1)

is\_load = pass1\_load(obf3);

closedir(dir);

}

else

{

perror("");

return EXIT\_FAILURE;

}

}//해당 파일이 빈 파일이 아니며 디렉토리에 있으면 load 수행

if(is\_load != 1)

printf("error! : load가 정상적으로 수행되지 않았습니다.\n");//loader 수행 중 에러나면 예외처리

else if(obj1\_fileFlag != 0 && obj2\_fileFlag != 0 && obj3\_fileFlag != 0)

{//로드가 정상적으로 수행되었고 파일이 디렉토리에 존재하며 정상 파일일 경우, 각 입력이 있는것에 대해 load pass2 수행

if(strlen(obf1) != 0)

is\_load = pass2\_load(obf1);

if(strlen(obf2) != 0)

is\_load = pass2\_load(obf2);

if(strlen(obf3) != 0)

is\_load = pass2\_load(obf3);

}

if((is\_load != -1) && (obj1\_fileFlag != 0) && (obj2\_fileFlag != 0) && (obj3\_fileFlag !=0))

{

printf("\n");

printf("control symbol address length\n");

printf("section name\n");

printf("-----------------------------------------------\n");

for(i=0; i<estab\_i; i++)

{

if(strlen(estab[i].csname) == 0)//symbol의 경우 len이 아예 출력되면 안되므로 경우 나누어줌

printf("%-13s%-10s %04X\n",estab[i].csname,estab[i].symname,estab[i].addr);

else

printf("%-13s%-10s %04X %04X\n",estab[i].csname,estab[i].symname,estab[i].addr,estab[i].len);

total\_len += estab[i].len;

}

printf("-----------------------------------------------\n");

printf(" total length %04X\n",total\_len);

} //로드 끝난 후 load map 화면에 출력

}

}

/\*pass1\*/

int pass1\_load(char\* obf)

{//입력 파일에서 H,D레코드를 읽어 estab 생성

FILE \*fp;

int pass1\_return = 1;

char record;

char Name[MAX];

char text[MAX];

unsigned int addr;

unsigned int len;

int i;

csaddr = progaddr;

fp = fopen(obf,"r");

while(1)

{

if(fscanf(fp,"%c",&record) == EOF)

break;

else

{

if(record == 'H')

{

fscanf(fp,"%s %06X %06X",Name,&addr,&len);

fgetc(fp);

progaddr += addr + len;//다음 프로그램을 위한 progaddr 재설정

for(i=0; i<=estab\_i; i++)

{

if(strcmp(estab[i].csname,Name))

continue;

else

{

printf("error : control section이 이미 estab에 있습니다.\n");

pass1\_return = -1;

break;

}

}

if(pass1\_return != -1)

{//estab에 없다면 추가

strcpy(estab[estab\_i].csname,Name);

estab[estab\_i].addr = csaddr + addr;

estab[estab\_i].len = len;

estab\_i++;

}

}//첫줄일때

else if(record == 'D')

{

while(1)

{

fscanf(fp,"%s %06X",Name,&addr);

for(i=0; i<=estab\_i; i++)

{

if(strcmp(estab[i].symname,Name))

continue;

else

{

printf("error : symbol name이 이미 estab에 있습니다.\n");

pass1\_return = -1;

break;

}

}

if(pass1\_return != -1)

{//estab 에 없다면 추가

strcpy(estab[estab\_i].symname,Name);

estab[estab\_i].addr = csaddr + addr;

estab[estab\_i].len = '\0';

estab\_i++;

}

if(fgetc(fp) == '\n')

break;//다음이 개행이면 루프 중지

else

fseek(fp,-1,SEEK\_CUR);//개행 아니면 파일포인터 원위치

}

}

else if(record=='R' || record=='T' || record=='M' || record=='.')

fgets(text,MAX-1,fp);

else if(record == 'E')

break;

else

{

printf("error! : 알맞은 record가 아닙니다.\n");

pass1\_return = -1;

}

}

}

fclose(fp);

return pass1\_return;

}

/\*pass2\*/

int pass2\_load(char\* obf)

{//pass1에서 생성한 estab을 통해 t,m레코드를 읽어 링크,메모리에 로드

int pass2\_return = 1;

FILE\* fp;

int i,j,k,temp\_i;

int find\_flag;//estab에서 찾았는지 여부

int half\_flag;//반비트 사용하는지 여부

int find\_ref = 0;//외부참조 찾았는지 여부

char M\_ch;//modify할때 더할지 뺄지 저장

char tmp\_ch;//half bit 가져올때 버리는 앞비트 저장

unsigned int T\_len, M\_len;

unsigned int T\_addr, M\_addr, fin\_addr;

char record;

char exname[MAX];

char T\_text[MAX], M\_text[MAX], mod\_addr[MAX];

char prog\_name[MAX];

char text[MAX];

unsigned int prog\_start, prog\_len;

unsigned int temp\_addr;

unsigned int mod\_addr\_hex, s\_len;

char put[3];

unsigned int put\_num[MAX],store\_addr[MAX];

fp = fopen(obf,"r");

while(1)

{

if(fscanf(fp,"%c",&record) == EOF)

break;

else

{

if(record == 'H')

{

fscanf(fp,"%s %06X %06X",prog\_name,&prog\_start,&prog\_len);

fgetc(fp);

extref[0].refnum = 1;

strcpy(extref[0].exname,prog\_name);

for(i=0; i<=estab\_i; i++)

{

if(!strcmp(estab[i].csname,prog\_name))

prog\_start = estab[i].addr;

}

}

else if(record == 'R')

{

extref\_i = 1;

while(1)

{

fscanf(fp,"%02X %s",&extref[extref\_i].refnum,exname);

strcpy(extref[extref\_i].exname,exname);

if(fgetc(fp) == '\n')

break;//다음이 개행이면 루프 중지

else

{

extref\_i++;

fseek(fp,-1,SEEK\_CUR);//개행 아니면 파일포인터 원위치

}

}

}

else if(record == 'T')

{

fscanf(fp,"%06X %02X %s",&T\_addr,&T\_len,T\_text);

fgetc(fp);

execaddr = prog\_start + T\_addr;

j=0;

for(i=0; i<MAX; i++)

put\_num[i] = 0;

for(i=0; (unsigned int)i<strlen(T\_text); i+=2)

{

put[0] = T\_text[i];

put[1] = T\_text[i+1];

put[2] = '\0';

put\_num[j] = strtoul(put,NULL,16);

j++;

}

i=0;

temp\_addr = dump\_p;//임시저장

dump\_p = execaddr;

while(dump\_p < execaddr+T\_len)

{

virtual[dump\_p] = put\_num[i];

dump\_p++;

i++;

}

dump\_p = temp\_addr;//원상복구

}//텍스트 레코드 가상 메모리에 올리기;; 중간에 16진수와 문자열 사이의 변환이 계속 있었다. (가상 메모리를 char로 선언)

else if(record == 'M')

{

fscanf(fp,"%06X %02X %s",&M\_addr,&M\_len,M\_text);

fgetc(fp);

M\_ch = M\_text[0];

strcpy(M\_text,M\_text+1);

for(i=0; i<MAX; i++)

store\_addr[i] = 0;

for(i=0; i<=extref\_i; i++)

{

if(strtoul(M\_text,NULL,16) == extref[i].refnum)

{

strcpy(M\_text,extref[i].exname);

find\_ref = 1;

}

else

continue;

}

if(find\_ref != 1)

{

printf("error! : 참조 내용이 estab에 없습니다.\n");

pass2\_return = -1;

return pass2\_return;//외부 참조의 내용이 estab에 없으면 error

}

execaddr = prog\_start + M\_addr;

for(i=0; i<=estab\_i; i++)

{

find\_flag = 0;

if(!strcmp(estab[i].symname,M\_text) || !strcmp(estab[i].csname,M\_text))

{

temp\_i = i;//나중에 해당 주소찾을때 쓰기위함

find\_flag = 1;

half\_flag = 0;

k=0;

if(M\_len % 2 != 0)

{

M\_len++;

half\_flag = 1;

}

M\_len = M\_len/2;

for(j=execaddr;(unsigned int)j<execaddr+M\_len; j++)

{

store\_addr[k] = virtual[j];

k++;

}//우선 t레코드 값 읽어옴

j=0;

for(i=0; i<k; i++)

{

sprintf(mod\_addr+j,"%02X",store\_addr[i]);

j+=2;

}

if(half\_flag == 1)

{

tmp\_ch = mod\_addr[0];

mod\_addr[0] = '0';

}//relocation을 할 때는 half bit를 이용해야 하는 경우가 있다. 이 경우 한 byte에서 앞 비트를 0으로 만들어줬다가 나중에 다시 원상복귀 시켜준다

mod\_addr\_hex = strtoul(mod\_addr,NULL,16);

if(M\_ch == '+')

fin\_addr = mod\_addr\_hex + estab[temp\_i].addr;

else if(M\_ch == '-')

fin\_addr = mod\_addr\_hex - estab[temp\_i].addr;

sprintf(mod\_addr,"%06X",fin\_addr);

if(half\_flag == 1)

mod\_addr[0] = tmp\_ch;

if(strlen(mod\_addr) > M\_len\*2)

{

s\_len = strlen(mod\_addr);

s\_len = s\_len - (M\_len\*2);

strcpy(mod\_addr, mod\_addr+s\_len);//계산 결과가 원래의 길이보다 길어질때 그만큼 잘라준다

}

break;

}

else

continue;

}

if(find\_flag == 0)//심볼이 estab에 없을때

{

printf("error : modify할 심볼이 estab에 없습니다.\n");

pass2\_return = -1;

}

if(pass2\_return != -1)

{

for(i=0; i<MAX; i++)

put\_num[i] = 0;

j=0;

for(i=0; (unsigned int)i<strlen(mod\_addr); i+=2)

{

put[0] = mod\_addr[i];

put[1] = mod\_addr[i+1];

put[2] = '\0';

put\_num[j] = strtoul(put,NULL,16);

j++;

}

i = 0;

temp\_addr = dump\_p;//임시저장

dump\_p = execaddr;

while(dump\_p < execaddr+M\_len)

{

virtual[dump\_p] = put\_num[i];

i++;

dump\_p++;

}

dump\_p = temp\_addr;//원상복구

}//dump메모리에 t와 m레코드를 계산하여 로드한다

}

else if(record == 'D' || record == '.')

fgets(text,MAX-1,fp);

else if(record == 'E')

break;

else

{

printf("error! : 알맞은 record가 아닙니다.\n");

pass2\_return = -1;

}

}

}

fclose(fp);

return pass2\_return;

}