**과목명: 시스템프로그래밍**

**2분반**

**<<Project #3>>**

**서강대학교 [심리학과]**

**[20140424]**

**[문성혁]**

목 차

1. **프로그램 개요**
2. **프로그램 흐름도**
3. **모듈 정의**
   1. 정의한 모듈의 변수 및 함수 소개
4. **파싱 전략**
5. 프로그램 개요

본 프로그램은 시스템 프로그래밍 수업의 세 번째 프로젝트 결과물입니다. 이번 프로젝트의 목표는 프로젝트 1에서 구현한 Shell에 Linking Loader의 기능을 추가하는 것이 핵심입니다. 즉, ‘loader’ 커맨드와 ‘run’ 커맨드로 ‘\*.obj’ 파일을 링킹하고 로딩하여 (프로젝트 1에서 생성한) 가상 메모리에 적재 및 실행하는 것입니다. 링킹은 pass1과 pass2를 거쳐 External Symbol Table 결과를 출력하고, 실행 전 Breakpoints를 설정해 특정 지점에서 프로그램 execution을 중지할 수 있습니다.

최종적으로 추가 또는 수정된 커맨드들은 다음과 같습니다.

* h[elp] (커맨드 추가됨)
* progaddr (추가됨)
* loader (추가됨)
* bp (추가됨)
* run (추가됨)

1. 프로그램 흐름도

전체적인 흐름도는 기존과 같으므로, 여기서는 loader 커맨드가 입력되었을 때의 흐름과 run 커맨드가 입력되었을 때의 흐름을 각각 상세히 전개하겠습니다. 참고로 모든 흐름도는 [이 링크](https://drive.google.com/file/d/1cgfVnlScFPZojPmojl7xPFtUvtpAl-_x/view?usp=sharing)에서 원 다이어그램을 확인할 수 있습니다.

우선, loader 메인 함수의 흐름도입니다.

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

아래 흐름도는 Loader 내부 Pass1의 흐름도입니다. 주어진 object 코드를 읽어서 ESTAB (External Symbol Table)을 갱신합니다.

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

아래 흐름도는 Loader 내부 Pass2의 흐름도입니다. 주어진 object 코드를 읽어서 ESTAB (External Symbol Table)을 갱신합니다.

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

다음은 run 메인 함수의 흐름도입니다.

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

각 명령어에 대한 수행 조건을 모두 작성한 뒤, 파싱에 집중했습니다.

1. 모듈 정의
   1. 정의한 모듈의 변수 및 함수 소개
2. linkingLoader.c : loader, run, bp, progaddr 함수가 모두 들어있습니다.

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
#include "linkingLoader.h"  
  
#define MEMORY\_SIZE 1048576  
  
extern unsigned char VM[MEMORY\_SIZE];  
static unsigned int prog\_addr = 0x00, prog\_end\_addr = 0x00;  
static int bps[MEMORY\_SIZE]; *// 0: off, 1: on*static int REGS[10];  
static char CC;  
static int bp\_done = 0; *// check if current breakpoint was handled or not yet*static estab\_node \_estab\_head = NULL;  
  
void \_init\_estab(void);  
  
int \_convert\_string\_to\_hex(char \*str);  
  
void \_add\_estab\_node(int cs\_length, unsigned int address, char \*symbol);  
  
void \_print\_regs(void);  
  
estab\_node \_find\_estab\_node(char \*symbol);  
  
void \_execOP(unsigned int OP, unsigned int N, unsigned int I, unsigned int X, unsigned int B, int TA);

1. linkingLoader.h : external symbol table을 구현하는 코드가 포함되어 있습니다.

#ifndef LINKER\_H  
#define LINKER\_H  
  
*/\*\*  
 \* ESTAB (External Symbol Table) structure  
 \*/*typedef struct \_estab\_node \*estab\_node;  
struct \_estab\_node {  
 int cs\_length; *// 0: this node is external symbol / 1 or more: this is control section and the value is control section's length* unsigned int address;  
 char symbol[20];  
 estab\_node next\_block;  
};  
  
*/\*\*  
 \* set program address to start linker(by `loader` or loader(by 'run')  
 \*/*int progaddr(char \*addr);  
  
  
*/\*\*  
 \* load object file(s)  
 \*/*int loader(char \*filename1, char \*filename2, char \*filename3);  
  
*/\*\*  
 \* add/clear/show breakpoint  
 \*/*int bp(char \*addr);  
  
*/\*\*  
 \* run the program starting from current prog\_addr  
 \*/*int run(void);  
#endif

1. 파싱 전략

프로젝트 2의 strtok와는 다르게 sscanf를 사용했습니다. loader()의 pass1, pass2에 이 과정이 잘 드러나 있습니다.

*// Pass 1*for (int i = 0; i < file\_num; ++i) {  
 FILE \*fp = fps[i];  
 int CSLTH = 0;  
  
 while (1) {  
 fgets(line, sizeof(line), fp);  
 if (line[strlen(line)] == '\n') {  
 line[strlen(line) - 1] = '\0'; *// delete '\n'* }  
 if (line[0] == 'H') {  
 char prog\_name[7], start\_addr[7], end\_addr[7];  
 memset(prog\_name, '\0', sizeof prog\_name);  
 memset(start\_addr, '\0', sizeof start\_addr);  
 memset(end\_addr, '\0', sizeof end\_addr);  
 sscanf(line + 1, "%6c", prog\_name);  
 sscanf(line + 1 + 6, "%6c", start\_addr);  
 sscanf(line + 1 + 6 \* 2, "%6c", end\_addr);  
 CSLTH = \_convert\_string\_to\_hex(end\_addr) - \_convert\_string\_to\_hex(start\_addr);  
 \_add\_estab\_node(CSLTH, CSADDR, prog\_name);  
 } else if (line[0] == 'E') {  
 CSADDR += CSLTH;  
 break;  
 } else if (line[0] == 'D') {  
 char es\_name[7], es\_addr[7];  
 int es\_cnt = ((int) strlen(line) - 1) / 12;  
 for (int j = 0; j < es\_cnt; ++j) {  
 memset(es\_name, '\0', sizeof es\_name);  
 memset(es\_addr, '\0', sizeof es\_addr);  
 sscanf(line + 1 + j \* 12, "%6c", es\_name);  
 sscanf(line + 1 + 6 + j \* 12, "%6c", es\_addr);  
 \_add\_estab\_node(0, CSADDR + \_convert\_string\_to\_hex(es\_addr), es\_name);  
 }  
 } else { *// line[0] == ('.' || 'T' || 'M' || 'R')* continue;  
 }  
 }  
}

*// Pass 2  
/\*PROGADDR = prog\_addr, \*/* CSADDR = prog\_addr;  
int EXECADDR = 0;  
estab\_node cs\_node = \_estab\_head;  
  
for (int i = 0; i < file\_num; ++i) {  
 FILE \*fp = fps[i];  
 fseek(fp, 0, SEEK\_SET);  
 int CSLTH = 0;  
 while (cs\_node->cs\_length == 0) cs\_node = cs\_node->next\_block;  
 int ref\_symbols[10];  
  
 while (1) {  
 fgets(line, sizeof(line), fp);  
 if (line[strlen(line) - 1] == '\n') {  
 line[strlen(line) - 1] = '\0'; *// delete '\n'* }  
 if (line[0] == 'H') {  
 CSLTH = cs\_node->cs\_length;  
 *// set 01 reference number (CS addr)* ref\_symbols[1] = (int) CSADDR;  
 } else if (line[0] == 'R') {  
 char ref\_num[3], symbol[7];  
 int ref\_cnt = ((int) strlen(line) - 1) / 8 + (((int) strlen(line) - 1) % 8 != 0 ? 1 : 0);  
 for (int j = 0; j < ref\_cnt; ++j) {  
 memset(ref\_num, '\0', sizeof ref\_num);  
 memset(symbol, '\0', sizeof symbol);  
 sscanf(line + 1 + j \* 8, "%2c", ref\_num);  
 sscanf(line + 1 + 2 + j \* 8, "%6c", symbol);  
 ref\_symbols[\_convert\_string\_to\_hex(ref\_num)] = (int) (\_find\_estab\_node(symbol)->address);  
 }  
 } else if (line[0] == 'T') {  
 char start\_addr[7], text\_len[3], text\_val[3];  
 memset(start\_addr, '\0', sizeof start\_addr);  
 memset(text\_len, '\0', sizeof text\_len);  
 sscanf(line + 1, "%6c", start\_addr);  
 sscanf(line + 1 + 6, "%2c", text\_len);  
 int target\_addr = \_convert\_string\_to\_hex(start\_addr);  
 for (int j = 0; j < \_convert\_string\_to\_hex(text\_len); ++j) {  
 memset(text\_val, '\0', sizeof text\_val);  
 sscanf(line + 1 + 6 + 2 + j \* 2, "%2c", text\_val);  
 VM[CSADDR + target\_addr] = (unsigned char) \_convert\_string\_to\_hex(text\_val);  
 target\_addr += 1;  
 }  
 } else if (line[0] == 'M') {  
 char target\_addr[7], text\_sz[3], ref\_num[3];  
 memset(target\_addr, '\0', sizeof target\_addr);  
 memset(text\_sz, '\0', sizeof text\_sz);  
 memset(ref\_num, '\0', sizeof ref\_num);  
 sscanf(line + 1, "%6c", target\_addr);  
 sscanf(line + 1 + 6, "%2c", text\_sz);  
 sscanf(line + 1 + 6 + 2 + 1, "%2c", ref\_num);  
 int target\_addr\_num = \_convert\_string\_to\_hex(target\_addr);  
 int num = VM[CSADDR + target\_addr\_num], backup = VM[CSADDR + target\_addr\_num];  
 if (\_convert\_string\_to\_hex(text\_sz) == 5) {  
 backup = (backup / 16) \* 16; *// first hex* num %= 16; *// second hex* }  
 num \*= 256;  
 num += VM[CSADDR + target\_addr\_num + 1];  
 num \*= 256;  
 num += VM[CSADDR + target\_addr\_num + 2];  
 if (\*(line + 1 + 6 + 2) == '+') {  
 num += ref\_symbols[\_convert\_string\_to\_hex(ref\_num)];  
 } else { *// if '-'* num -= ref\_symbols[\_convert\_string\_to\_hex(ref\_num)];  
 }  
 if (num > 0xFFFFF) num %= 0x100000;  
 else if (num < 0) num += 0x100000;  
 int first\_part = (num / 256) / 256;  
 if (\_convert\_string\_to\_hex(text\_sz) == 5) {  
 first\_part += backup;  
 }  
 VM[CSADDR + target\_addr\_num] = (unsigned char) first\_part;  
 VM[CSADDR + target\_addr\_num + 1] = (unsigned char) (num / 256);  
 VM[CSADDR + target\_addr\_num + 2] = (unsigned char) (num % 256);  
 } else if (line[0] == 'E') {  
 if (strlen(line) > 1) {  
 char exec\_addr[7];  
 memset(exec\_addr, '\0', sizeof exec\_addr);  
 sscanf(line + 1, "%6c", exec\_addr);  
 EXECADDR = \_convert\_string\_to\_hex(exec\_addr);  
 }  
 CSADDR += CSLTH;  
 break;  
 } else {  
 continue;  
 }  
 }  
 cs\_node = cs\_node->next\_block;  
}

이상으로 프로젝트3 보고서를 마칩니다.