ICS Homework 11

April 27, 2023

1 Organization

1.1 Sections and Symbol Table

Assume we compile the following code, foo.c to object file foo.o. Fill the table.

```c
const int const_ten = 10;
int ten = 10;
int zero = 0;
int uninit;
static int static_ten = 10;
static int static_uninit;

int bar();
int f() {
    static int f_static_i = 10;
    int f_i = 0;
    return bar();
}

int main() {}
```

Symbol	.symtab entry?(T/F)	Symbol type	Global/Local	Section
const_ten				
ten				
zero				
uninit				
static_ten				
static_uninit				
f				
f_static_i				
f_i				
bar				
main				

1.2 Symbol Resolution

foobar.c:

```
1: #include <stdio.h>
2:
3: int x;
4: int y;
5: int z = 10;
6: static int a = 10;
7: static int b = 20;
8:
9: void bar();
10:
11: void foo() {
12:     static int b = 10;
13:     int c = 5;
14:
15:     printf("%d\n", x);
16:     printf("%d\n", b);
17:     printf("%d\n", c);
18:     bar();
19: }
20:
21: __attribute__((weak)) void bar() {
22:     static int d = 10;
23:     int c = 5;
24:
25:     printf("%d\n", x);
26:     printf("%d\n", b);
27:     printf("%d\n", c);
28:     printf("%d\n", d);
29: }
```

barbaz.c:

```
1: #include <stdio.h>
2:
3: int x = 5;
4: int y = 10;
5: extern int z;
6: static int a = 40;
7: static int b = 30;
8:
9: void bar();
10:
11: void baz() {
12:     static int b = 10;
13:     int c = 5;
14:
15:     printf("%d\n", x);
16:     printf("%d\n", b);
17:     printf("%d\n", c);
18:     printf("%d\n", y);
19:     bar();
20: }
21:
```

```
22: void bar() {
23:     static int d = 10;
24:     int c = 5;
25:
26:     printf("%d\n", x);
27:     printf("%d\n", b);
28:     printf("%d\n", c);
29:     printf("%d\n", z);
30: }
```

Please show the actually used variable in each reference:

Reference	Variable
foobar.c:15:x	
foobar.c:16:b	
foobar.c:17:c	
foobar.c:18:bar	
foobar.c:25:x	
foobar.c:26:b	
foobar.c:27:c	
foobar.c:28:d	
barbaz.c:15:x	
barbaz.c:16:b	
barbaz.c:17:c	
barbaz.c:18:y	
barbaz.c:19:bar	
barbaz.c:26:x	
barbaz.c:27:b	
barbaz.c:28:c	
barbaz.c:29:z	

1.3   Static Libraries

Let a and b denote object modules or static libraries in the current directory, and let a -¿ b denote that a depends on b, in the sense that b defines a symbol that is referenced by a. For each of the following scenarios, show the minimal command line (i.e., one with the least number of object file and library arguments) that will allow the static linker to resolve all symbol references.

A. p.o -¿ libx.a

B. p.o -¿ libx.a -¿ liby.a

C. p.o -¿ libx.a -¿ liby.a and liby.a -¿ libx.a-¿ p.o

D: p.o -¿ libx.a -¿ liby.a -¿ libz.a and liby.a -¿ libx.a -¿ libz.a

2   System Software

2.1   Shared Varibles in multi-threading

```
#include "csapp .h"
#define N 4
void *print_thread(void *vargp)
{
    int myid = *((int)vargp);
    printf("in thread %d\n", myid);
    return NULL;
}
```

```
int main() {
    pthread t tid[N];
    int *ptr;
    for (int i = 0; i < N; i++) {
        ptr = malloc(sizeof(int));
        *ptr = i;
        // Creat a thread to run the "print_thread func with arg ptr
        // Your core here: _____
        free(ptr);
    }

    for (int i = 0; i < N; i++)
        pthread join(tid[i], NULL);
}
```

1. Complete the previous code according to the comment.
2. Is there any race condition in the previous code? Why or why not?