CS360 Pre-LAB1 Assignment

PART 1:

1. A binary executable file, a.out, consists of

|header| Code | Data |<-BSS->|

where BSS is for uninitialized globals and uninitialized static locals.

The Unix command size a.out shows the size of TEXT, DATA, BSS of a.out.

Use the following C program, t1.c, to generate t2.c, t3.c... t6.c as specified

below.

//\*\*\*\*\*\*\*\*\*\* t1.c file \*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

int g;

int main ()

{

int a,b,c;

a = 1; b = 2;

c = a + b;

printf ("c=%d\n", c);

}

t2.c: Change the global variable g to int g=3;

t3.c: Change the global variable g to int g[10000];

t4.c: Change the global variable g to int g[10000] = {4};

t5.c: Change the local variables of main () to

int a,b,c, d[10000];

t6.c. Change the local variables of main () to

static int a,b,c, d[10000];

(A). For each case, use cc -m32 t.c to generate a.out.

Then use ls -l a.out to get a.out size.

run size a.out to get its section sizes.

Record the observed sizes in a table:

Case | a.out | TEXT | DATA | BSS |

--------------------------------------------

(1) | | | | |

--------------------------------------------

(2)

--------------------------------------------

(3)

--------------------------------------------

(4)

--------------------------------------------

(5)

--------------------------------------------

(6)

--------------------------------------------

ANSWER THESE QUESTIONS:

1. Variables in C may be classified as

globals ---|--- UNINITIALIZED globals;

|--- INITIALIZED globals;

locals ---|--- AUTOMATIC locals;

|--- STATIC locals;

In terms of the above classification and the variables g, a, b, c, d,

Which variables are in DATA? \_ STATIC locals, INITIALIZED globals\_\_

Which variables are in BSS? UNINITIALIZED globals\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. In terms of the TEXT, DATA and BSS sections,

Which sections are in a.out, which section is NOT in a.out?

WHY? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Text, Data in a.out, BSS not\_\_\_\_\_\_\_\_\_\_\_\_

(B). For each case, use cc -m32 -static t.c to generate a.out.

Record the sizes again and compare them with the sizes in (A).

WHAT DO YOU SEE? \_\_size are the same\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

WHY? \_\_\_because a.out has fixed size of 0x23c bytes\_\_\_\_\_\_\_\_\_\_

Part 2:

Given the following t.c and ts.s files

Under Linux, use gcc -m32 t.c ts.s to generate an a.out

Run a.out as a.out one two three > outfile

DO the requirements 1 to 5 as specified below.

# ts.s file:

.global getebp

getebp:

movl %ebp, %eax

ret

/\*\*\*\*\*\*\*\*\*\*\*\*\* t.c file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

int \*FP;

int main(int argc, char \*argv[ ], char \*env[ ])

{

int a,b,c;

printf("enter main\n");

printf("&argc=%x argv=%x env=%x\n", &argc, argv, env);

printf("&a=%8x &b=%8x &c=%8x\n", &a, &b, &c);

(1). Write C code to print values of argc and argv[] entries

a=1; b=2; c=3;

A(a,b);

printf("exit main\n");

}

int A(int x, int y)

{

int d,e,f;

printf("enter A\n");

// write C code to PRINT ADDRESS OF d, e, f

printf("&d=%8x &e=%8x &f=%8x\n", &d, &e, &f);

d=4; e=5; f=6;

B(d,e);

printf("exit A\n");

}

int B(int x, int y)

{

int g,h,i;

printf("enter B\n");

// write C code to PRINT ADDRESS OF g,h,i

printf("&g=%8x &h=%8x &i=%8x\n", &g, &h, &i);

g=7; h=8; i=9;

C(g,h);

printf("exit B\n");

}

int C(int x, int y)

{

int u, v, w, i, \*p;

printf("enter C\n");

// write C cdoe to PRINT ADDRESS OF u,v,w,i,p;

printf("&u=%8x &v=%8x &w=%8x\n", &u, &v, &w);

u=10; v=11; w=12; i=13;

FP = (int \*)getebp(); // FP = stack frame pointer of the C() function

print FP value in HEX

(2). Write C code to print the stack frame link list.

p = (int \*)&p;

p = FP;

while (p != NULL){

printf (“FP -> %8x ->\n”, p);

}

i = 0;

while (i<128){

printf (“%d(FP)->%8x\n%d(FP) = %d\n”I, FP, I, \*FP);

FP ++;

i ++;

}

(3). Print the stack contents from p to the frame of main()

YOU MAY JUST PRINT 128 entries of the stack contents.

(4). On a hard copy of the printout, identify the stack contents

as LOCAL VARIABLES, PARAMETERS, stack frame pointer of each function.