James Jessen
10918967

Pre-Work #2

CptS 360

(A). For each case, use cc -m32 t.c to generate a.out. Then use ls -l a.out to get a.out size, and run size a.out to get its section sizes. Record the observed sizes in a table:

Case	a.out	TEXT	DATA	BSS	Difference
(1)	7313	1158	276	8	
(2)	7317	1158	280	4	Initialized Global (g)
(3)	7313	1158	276	40032	Uninitialized Global Array (g)
(4)	47349	1158	40304	4	Initialized Global Array (g)
(5)	7313	1190	276	8	Uninitialized Local Automatic Array (d)
(6)	7405	1174	276	40068	Uninitialized Local Static Array (d)

1. Variables in C may be classified as

```
     globals ---|--- UNINITIALIZED  globals;
                |---   INITIALIZED  globals;

     locals  ---|--- AUTOMATIC locals;
                |--- STATIC    locals;
```

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

Which variables are in DATA?
Initialized Global/Static variables like g(t2.c,t4.c).
Which variables are in BSS?
Uninitialized Global/Static variables like g(t1.c,t3.c) and d(t6.c).

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 and DATA are in a.out but BSS is not. The BSS section is excluded from a.out to avoid unnecessarily wasting memory on uninitialized variables when the program is not being executed.**

(B). For each case, use cc -static t.c to generate a.out.
Record the sizes again and compare them with the sizes in (A).

Case	a.out	TEXT	DATA	BSS
(1)	721014	649142	4108	5692
(2)	721014	649142	4108	5660
(3)	721014	649142	4108	45660
(4)	761046	649142	44140	5660
(5)	721014	649174	4108	5692
(6)	721106	649158	4108	45724

WHAT DO YOU SEE? **The sizes are much larger, particularly TEXT.**

WHY? **With static linking, the loader is not used. Thus, the linker must include every needed library in a.out; significantly increasing its size.**

James Jessen
10918967

Pre-Work #2

CptS 360

```
enter main
&argc=bf8c0230 argv=bf8c02c4 env=bf8c02d8
&a=bf8c021c &b=bf8c0218 &c=bf8c0214
enter A
&d=bf8c01ec &e=bf8c01e8 &f=bf8c01e4
enter B
&g=bf8c01bc &h=bf8c01b8 &i=bf8c01b4
enter C
&u=bf8c0188 &v=bf8c0184 &w=bf8c0180
ebp=bf8c0198
```

FP -> Stack Frame of current function

PC -> Next instruction to be executed

```
        Address     Contents
        bf8c0180    b              local var w
        bf8c0184    a              local var v
        bf8c0188    9              local var u
        bf8c018c    3              local var iterator(for loop 100)
C       bf8c0190    0
        bf8c0194    0
        bf8c0198    bf8c01c8       FP
        bf8c019c    80485b6        PC -> B():printf("exit B\n");
        bf8c01a0    6              Arg1  (g value)
        bf8c01a4    7              Arg2  (h value)
        bf8c01a8    bf8c01b8
        bf8c01ac    bf8c01b4       temps
        bf8c01b0    b7730ac0
        bf8c01b4    8              local var i
        bf8c01b8    7              local var h
        bf8c01bc    6              local var g
        bf8c01c0    0
B       bf8c01c4    0
        bf8c01c8    bf8c01f8       FP
        bf8c01cc    804854e        PC -> A():printf("exit A\n");
        bf8c01d0    3              Arg1  (d value)
        bf8c01d4    4              Arg2  (e value)
        bf8c01d8    bf8c01e8
        bf8c01dc    bf8c01e4       temps
        bf8c01e0    b7730ac0
        bf8c01e4    5              local var f
        bf8c01e8    4              local var e
        bf8c01ec    3              local var d
        bf8c01f0    bf8c0230
A       bf8c01f4    b776a8f8
        bf8c01f8    bf8c0228       FP
        bf8c01fc    80484e6        PC -> main():printf("exit main\n");
        bf8c0200    1              Arg1  (a value)
        bf8c0204    2              Arg2  (b value)
        bf8c0208    bf8c0218
        bf8c020c    bf8c0214       temps
        bf8c0210    b77303c4
        bf8c0214    3              local var c
        bf8c0218    2              local var b
        bf8c021c    1              local var a
        bf8c0220    80486a0
main    bf8c0224    0
        bf8c0228    0              FP (end of linked-list)
        bf8c022c    b758fb73       PC -> crt0
        bf8c0230    4              argc
        bf8c0234    bf8c02c4       -> argv[]
        bf8c0238    bf8c02d8       -> env[]
        bf8c023c    b77466b0
        bf8c0240    1
```

Low

↕

High

Garbage

↓

```
bf8c0244     1
bf8c0248     0
bf8c024c     804a018
bf8c0250     804822c
bf8c0254     b7730000
bf8c0258     0
bf8c025c     0
bf8c0260     0
bf8c0264     495866db
bf8c0268     e0aa62ca
bf8c026c     0
bf8c0270     0
bf8c0274     0
bf8c0278     4
bf8c027c     8048350
bf8c0280     0
bf8c0284     b775efc0
bf8c0288     b758fa89
bf8c028c     b7769fbc
bf8c0290     4
bf8c0294     8048350
bf8c0298     0
bf8c029c     8048371
bf8c02a0     8048460
bf8c02a4     4
bf8c02a8     bf8c02c4
bf8c02ac     80486a0
bf8c02b0     8048710
bf8c02b4     b7759870
bf8c02b8     bf8c02bc
bf8c02bc     1c
bf8c02c0     4
bf8c02c4     bf8c044b
bf8c02c8     bf8c0451
bf8c02cc     bf8c0455
bf8c02d0     bf8c0459
bf8c02d4     0
bf8c02d8     bf8c045f
bf8c02dc     bf8c046a
bf8c02e0     bf8c047b
bf8c02e4     bf8c049a
bf8c02e8     bf8c04cf
bf8c02ec     bf8c04e0
bf8c02f0     bf8c04f4
bf8c02f4     bf8c0504
bf8c02f8     bf8c051b
bf8c02fc     bf8c0529
bf8c0300     bf8c0541
bf8c0304     bf8c0553
bf8c0308     bf8c0587
bf8c030c     bf8c05a8
exit C
exit B
exit A
exit main
```