Nathan Bender

Professor Katsinis

CS 283

H7

**7.6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | .Symtab Entry? | Symbol Type | Module Defined | Section |
| buf | Yes | External | Main.o | .data |
| bufp0 | Yes | Global | Swap.o | .data |
| bufp1 | Yes | Local | Swap.o | .bss |
| swap | Yes | Global | Swap.o | .text |
| temp | No | - | - | - |
| incr | Yes | Local | Swap.o | .text |
| count | Yes | Local | Swap.o | .data |

**7.7**

The problem with the program is that the definition of x in foo5.c, “int x = 15213;” is a strong symbol, while the declaration of x in bar5.c, “int x;” is a weak symbol. Global functions and *initialized* global variables are classified as strong while *uninitialized,* which is seen in bar5.c, is classified as weak. To fix this issue, we can assign the definition of x in bar5.c as static. By declaring the variable as static, the variable will only be visible to the file that it is declared in.

/\* foo5.c \*/

#include <stdio.h>

void f(void);

int x = 15213;

int y = 15212;

int main()

{

f();

printf("x = 0x%x y = 0x%x\n", x, y);

}

/\* bar5.c \*/

**static double x;**

void f(void)

{

x = -0.0;

}

**7.9**

The definition of the function main in foo6.c is classified as a strong symbol definition, while the definition of the local char variable main in bar5.c is classified as a weak symbol definition. Strong symbols include both functions/procedures as well as initialized global variables. Weak symbols include the uninitialized global variables. Since the char main in bar5.c is uninitialized, it is classified as a weak symbol. Similarly, since main in foo6.c is a function, it is classified as a strong symbol. Due to this, the linker chooses the function main from foo6.c instead of the local char variable from bar6.c. Therefore, the value that is being printed to the screen is the address of the function main.