In class 1A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In class 2

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Access code: link

1.1

- We have same symbol but different types. One is a function and one is a symbol

- main in module 2 is static, so it is not a strong or weak symbol and it only belongs to module 2. Since it is a local symbol, there are no multiply defined global symbols.

Ref(main.1) -> def(main.1)

Ref(main.2) -> def(main.2)

1.2

- both int x and double x are weak, so the gcc compiler will pick one randomly

Ref(x.1) -> def(unknown)

Ref(x.2) -> def(unknown)

1.3

- same problem, but we initialize both of the multiple definitions of x. so they are both strong

- since we have 2 strong definitions of the same global variable, we get an error

Ref(x.1) -> def(error)

Ref(x.2) -> def(error)

2.0

- this is like the one on the quiz but x is defined before y in foo.c

- we have strong and weak, and different data type. So the compiler will consider both defs of x as one copy. The linker favors the strong definition

- y starts right after x ends

- bar.c will write 8 bytes to x because it believes it’s a double, but foo.c has strong def of x so the compiler looks at the x in foo and only gives 4 bytes to x, so y is overwritten.

- note: %x is hexadecimal

X = 0, y = some garbage value

- note: the y garbage value has an 8, this is because the x is set to a negative 0.0, so there is some bit for sign which is bleeding into y memory

2.1

same prob but y is defined first

* The int x is strong and thus is favored, so x has 4 bytes
* Y is unaffected because it is defined before x

In class 3

A screenshot of a computer

Description automatically generated

In class 4 // code: lkm

A screenshot of a computer program

Description automatically generated

A white text with black text

Description automatically generated

A computer code on a white background

Description automatically generated

**Code: sched**

1. SJF has problem A. it favors the shortest jobs, so long jobs may never run. It has problem B, because we don’t actually know how long a job will take. Answer: C. average turnaround time …

2. option 1: start with the highest queue and never demote anything. Option 2: (were not allowed to touch the scheduler) just have 1 queue. This works because we are not touching the scheduler logic, we are just controlling the number of total queues.

3. boost it every 20 time slices, so that it gets 1/20 of the total CPU time (5%). So 200 ms

The first cat gives Helloworld

The second one were writing some string Cougs to the buffer, but were not interacting with the user level buffer, so were not overwriting anything. So the helloworld buffer will remain the same so it will still print helloworld

To fix it so it overwrites helloworld then change the function to the one from the in class program