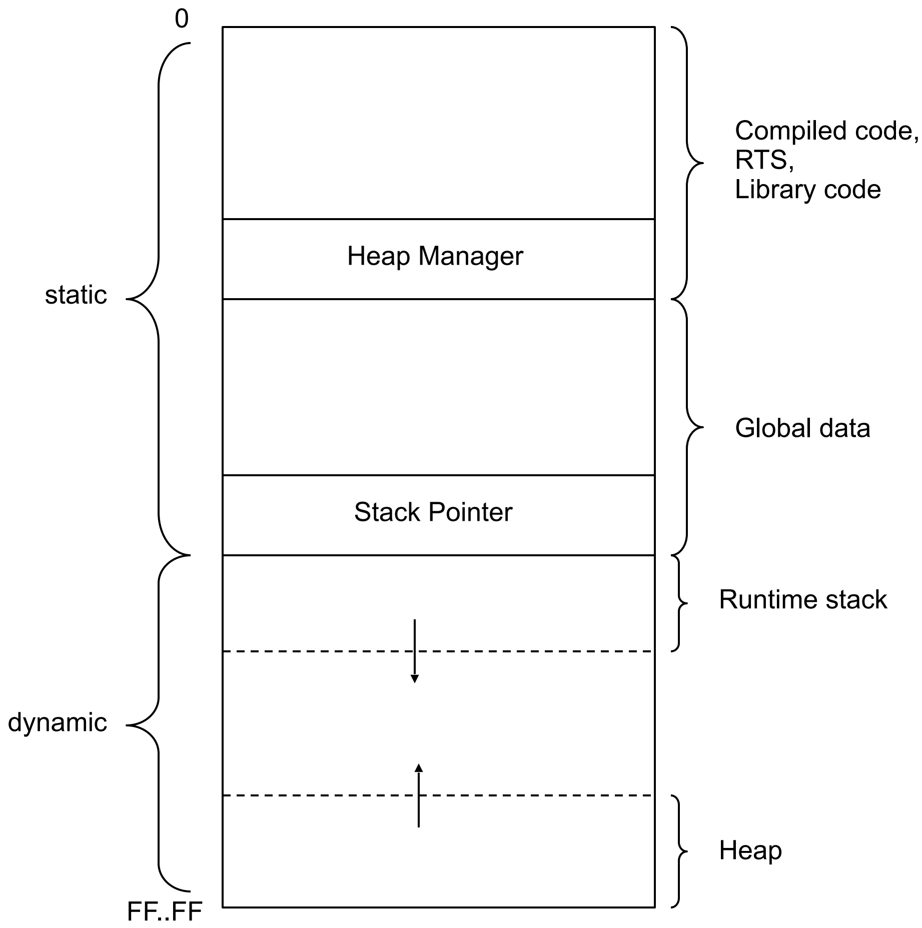
# Memory Management

For most programming languages memory management has two parts:

(1) Static - global data, compiled code, runtime system
(2) Dynamic - runtime stack (activation record stack), heap (!)

# Typical Memory Layout




A typical memory layout for languages such as C and Java

NOTE: if the runtime stack and the heap meet $\Rightarrow$ out of memory

# The Heap

Runtime systems allocate dynamically created objects on the heap by a call to the heap manager.

In Java/C++ the heap manager is called with the new keyword.

In C the heap manager is called using the malloc function.

Observation:
In languages like Java and Python heap memory is reclaimed by the heap manager automatically via garbage collection when it is no longer used.

In C the programmer has to explicitly manage heap memory with malloc/free function calls. This is error prone and leads to the (in)famous dangling pointer reference (free called too early) and the memory leak (free never called) problems.

# Example C (Memory Leak)

Program

```
struct Object * o;

void f()
{
    o = malloc(sizeof(struct Object));



    struct Object * p = o;



    o = NULL;




}

(pop activation record off the runtime stack)
```

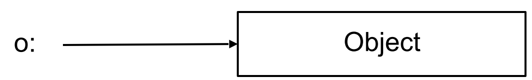
Heap Manager

# Example C (Memory Leak)

Program

```
struct Object * o;

void f()
{
    o = malloc(sizeof(struct Object));



    struct Object * p = o;



    o = NULL;




}

(pop activation record off the runtime stack)
```

Heap Manager



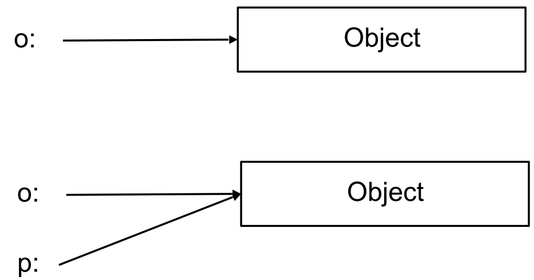
# Example C (Memory Leak)

Program

```
struct Object * o;

void f()
{
    o = malloc(sizeof(struct Object));


    struct Object * p = o;


    o = NULL;



}
```

(pop activation record off the runtime stack)

Heap Manager



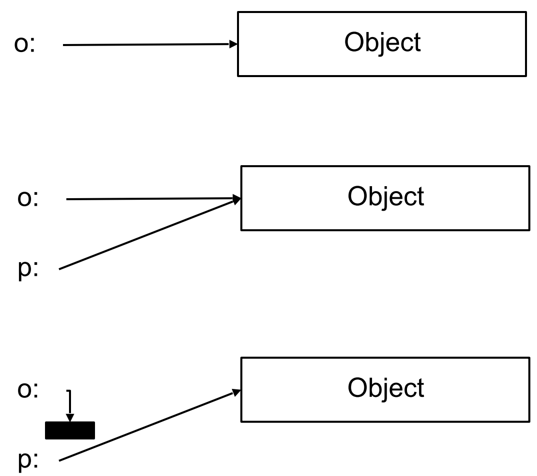
# Example C (Memory Leak)

Program

```
struct Object * o;

void f()
{
    o = malloc(sizeof(struct Object));



    struct Object * p = o;



    o = NULL;



}
```

(pop activation record off the runtime stack)

Heap Manager



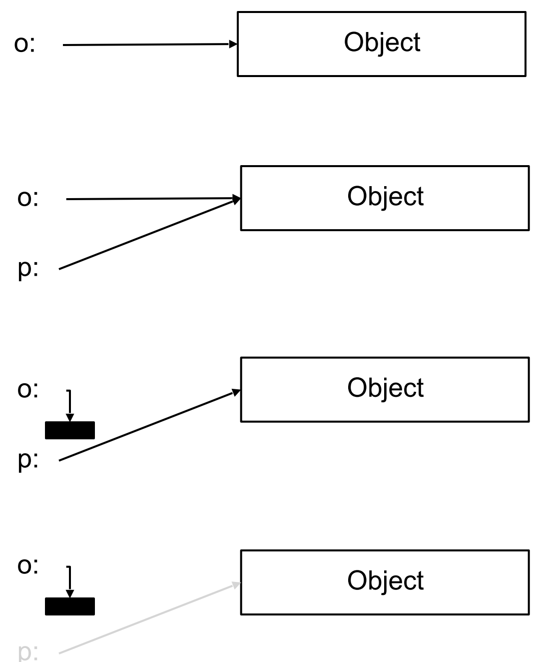
# Example C (Memory Leak)

Program

```
struct Object * o;

void f()
{
   o = malloc(sizeof(struct Object));

   struct Object * p = o;

   o = NULL;

}
```

(pop activation record off the runtime stack)

Heap Manager




Note: the heap manager has not way of knowing that this memory is no longer used ⇒<u>memory leak</u>

# Example C (Dangling Pointer)

Program

Heap Manager

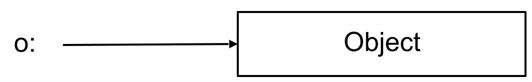
```
void f()
{
    struct Object * o = malloc(sizeof(struct Object));

    free(o);

    struct Foo * p = malloc(sizeof(struct Foo));

     o->ObjectAttribute = value;

     p->Print();
     free(p);
}
```

# Example C (Dangling Pointer)

Program

```
void f()
{
    struct Object * o = malloc(sizeof(struct Object));



    free(o);




    struct Foo * p = malloc(sizeof(struct Foo));




     o->ObjectAttribute = value;

     p->Print();
     free(p);
}
```

Heap Manager

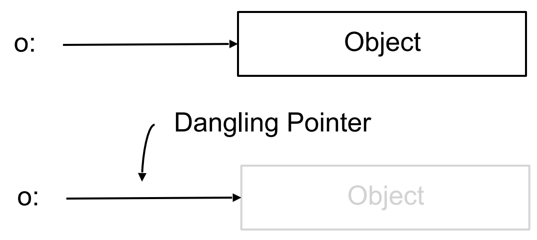


# Example C (Dangling Pointer)

Program

```
void f()
{
   struct Object * o = malloc(sizeof(struct Object));



   free(o);




   struct Foo * p = malloc(sizeof(struct Foo));



    o->ObjectAttribute = value;

    p->Print();
    free(p);
}
```

Heap Manager

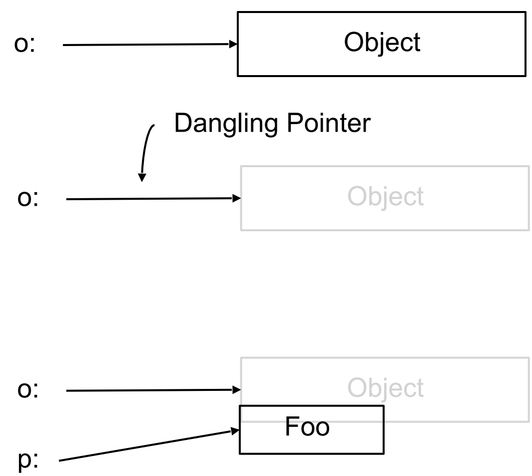


# Example C (Dangling Pointer)

Program

```
void f()
{
   struct Object * o = malloc(sizeof(struct Object));



   free(o);



   struct Foo * p = malloc(sizeof(struct Foo));



    o->ObjectAttribute = value;

    p->Print();
    free(p);
}
```

Heap Manager

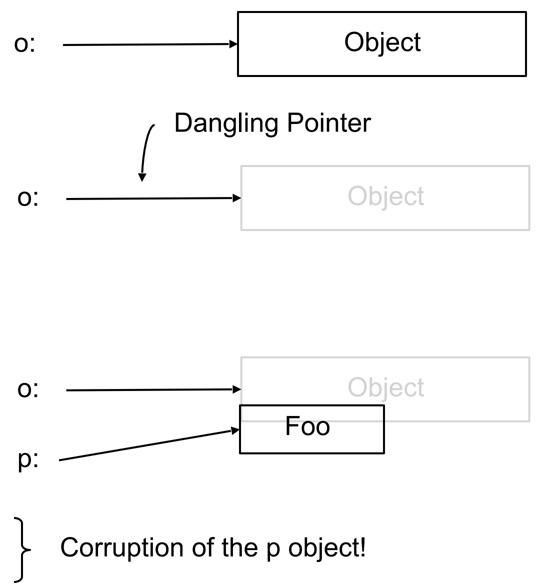


# Example C (Dangling Pointer)

Program

```
void f()
{
   struct Object * o = malloc(sizeof(struct Object));


   free(o);


   struct Foo * p = malloc(sizeof(struct Foo));


    o->ObjectAttribute = value;

    p->Print();
    free(p);
}
```

Heap Manager



# Example Java (Garbage Collection)

Program

```
void f()
{
   Object o = new Object();



   Object p = o;




   p = null;


}
```

(pop activation record off the runtime stack)

Heap Manager

Java uses a garbage collection technique called <u>reference counting</u>.

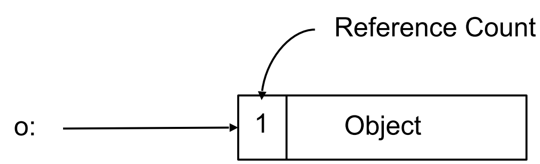
# Example Java (Garbage Collection)

Program

```
void f()
{
   Object o = new Object();



   Object p = o;




   p = null;


}
```

(pop activation record off the runtime stack)

Heap Manager




Java uses a garbage collection technique called <u>reference counting</u>.

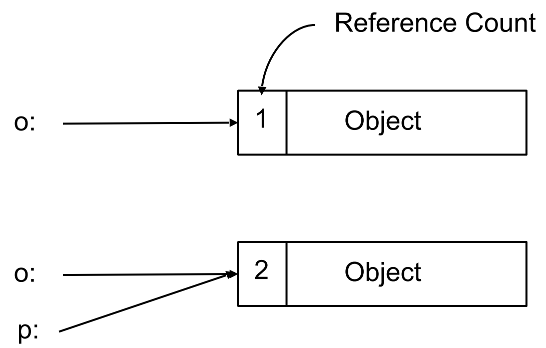
# Example Java (Garbage Collection)

Program

```
void f()
{
   Object o = new Object();



   Object p = o;



   p = null;

}
```

(pop activation record off the runtime stack)

Heap Manager




Java uses a garbage collection technique called <u>reference counting</u>.

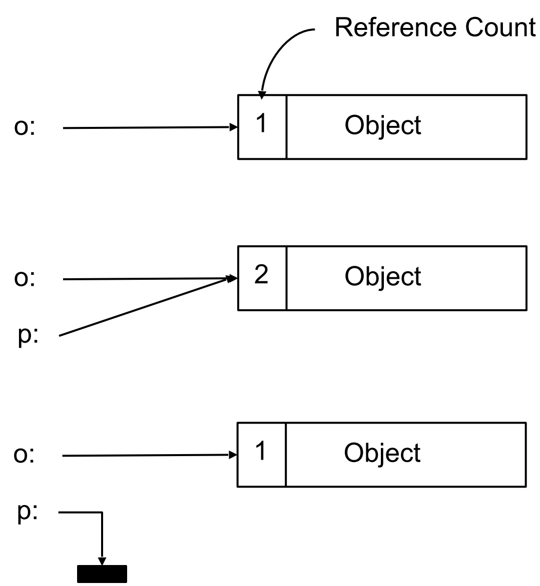
# Example Java (Garbage Collection)

Program

```
void f()
{
   Object o = new Object();



   Object p = o;



   p = null;

}
```

(pop activation record off the runtime stack)

Heap Manager




Java uses a garbage collection technique called <u>reference counting</u>.

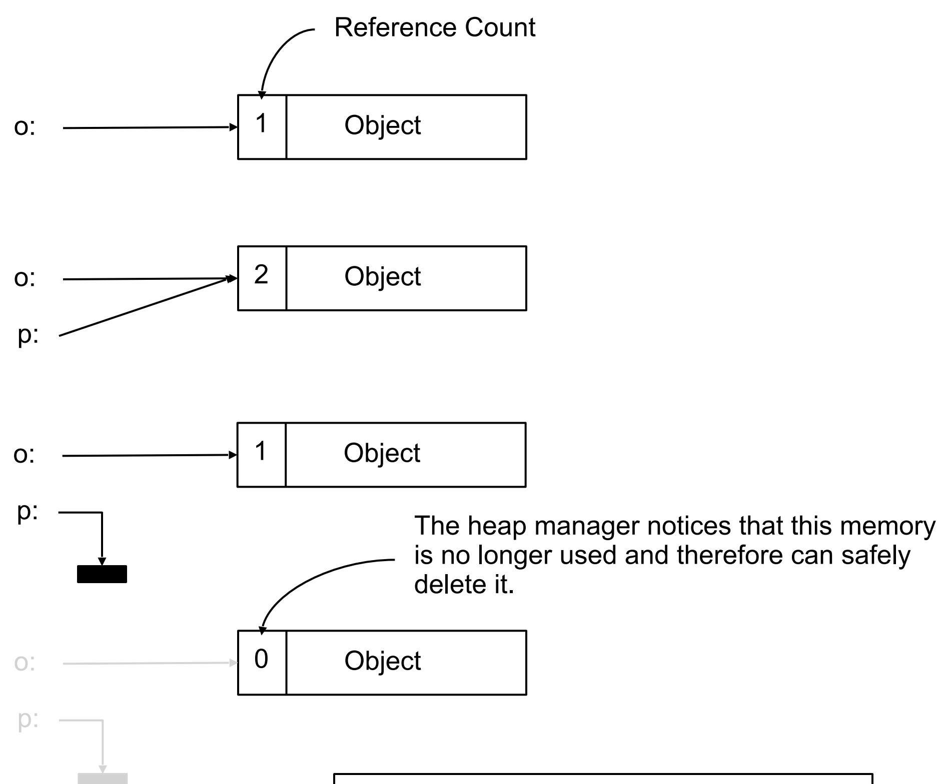
# Example Java (Garbage Collection)

Program

```
void f()
{
   Object o = new Object();

   Object p = o;

   p = null;

}
```

(pop activation record off the runtime stack)

Heap Manager




Java uses a garbage collection technique called <u>reference counting</u>.