The Heap

Embedded Software Essentials

C1M3V7

Be Boulder.

University of Colorado Boulder

Data Segment

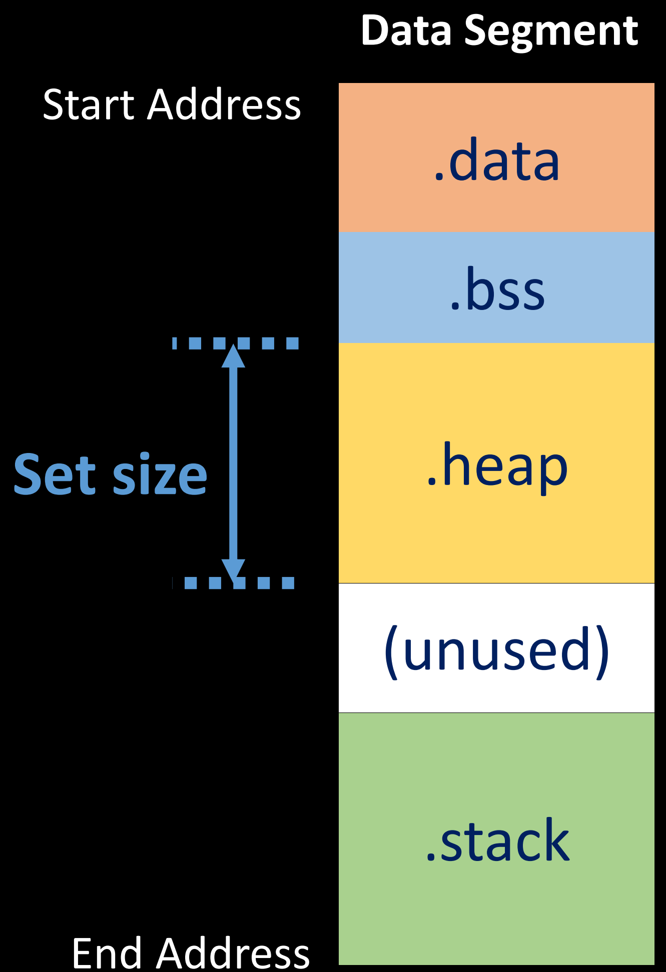
- Four Main Segments
  - Stack
  - Heap
  - Data
  - BSS

```
void foo(){
  char * ptr_TO_HEAP;

  ptr_TO_HEAP = (char *)malloc(8);

  /* More Code Here */

  free((void *)ptr_TO_HEAP);

}
```

- Heap space is reserved at compile time, data is allocated at runtime by directly calling memory functions

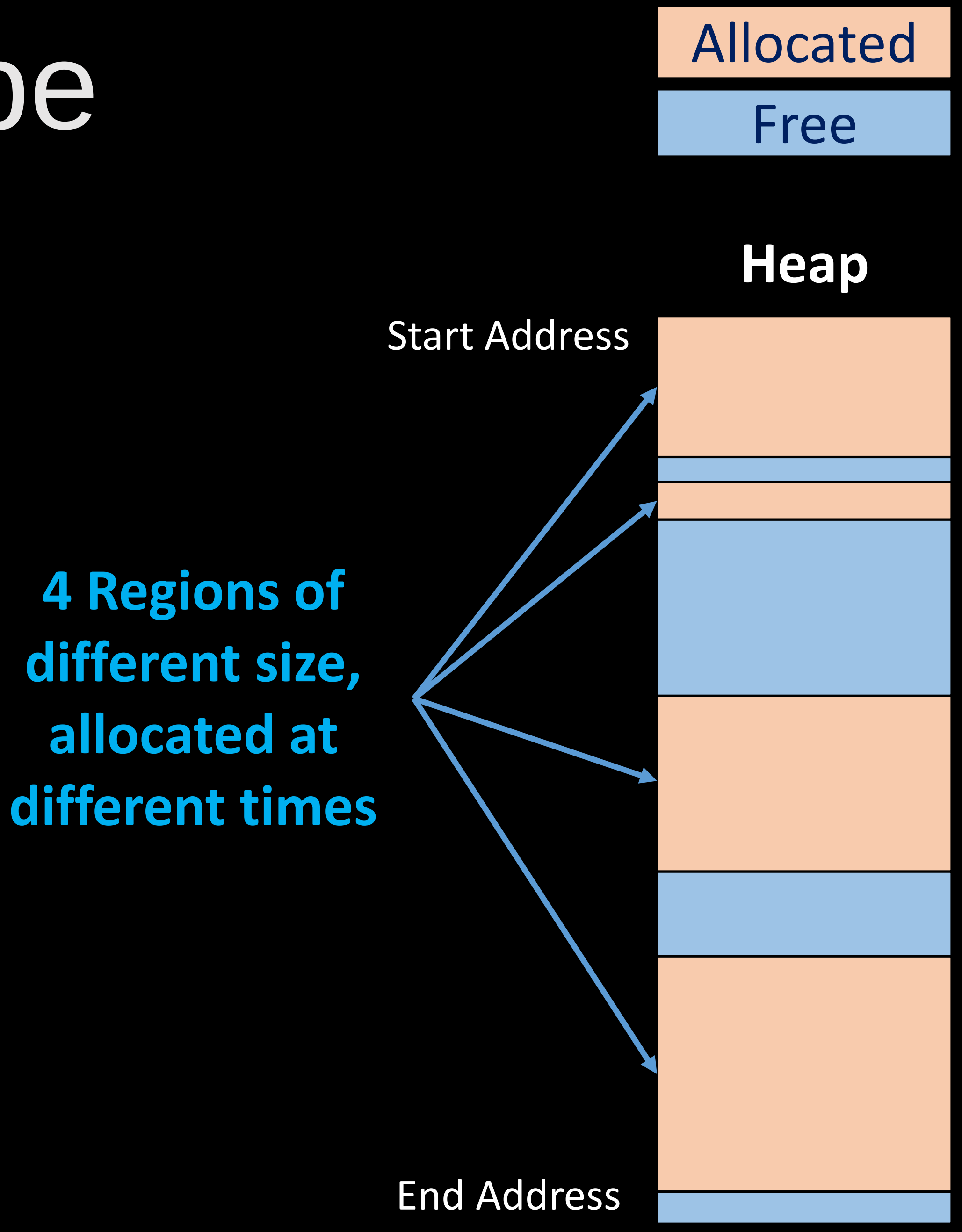
The Heap

- Heap space is a sub-segment of data memory reserved at compile time with a set size
- Data is allocated dynamically at runtime and managed by developer
- Each allocation can (pending space)
  - Vary in size
  - Be resized



Heap Lifetime and Scope

- Heap Data can have a lifetime longer than a function but less than the program
- Heap data can have a local scope or global scope
- Allocation/deallocation adds execution overhead



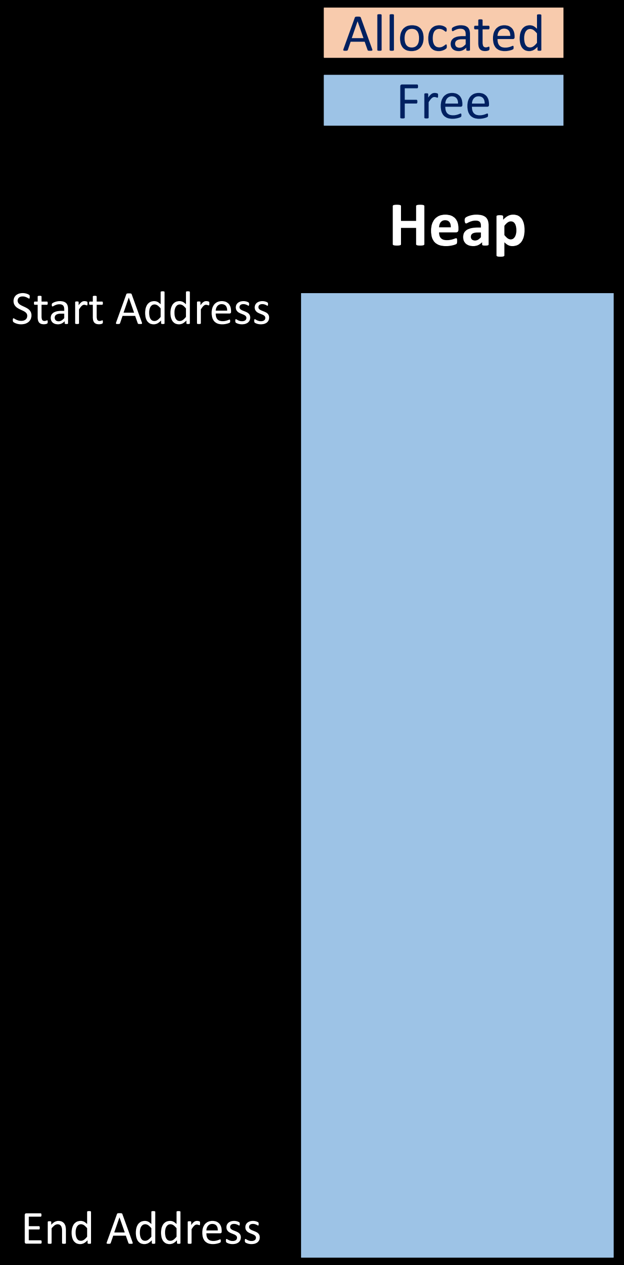
Heap Functions

- void * malloc (size_t size)

- void * calloc (size_t nitems, size_t item_size)

- void * realloc (void * ptr, size_t size)

- void free (void * ptr)

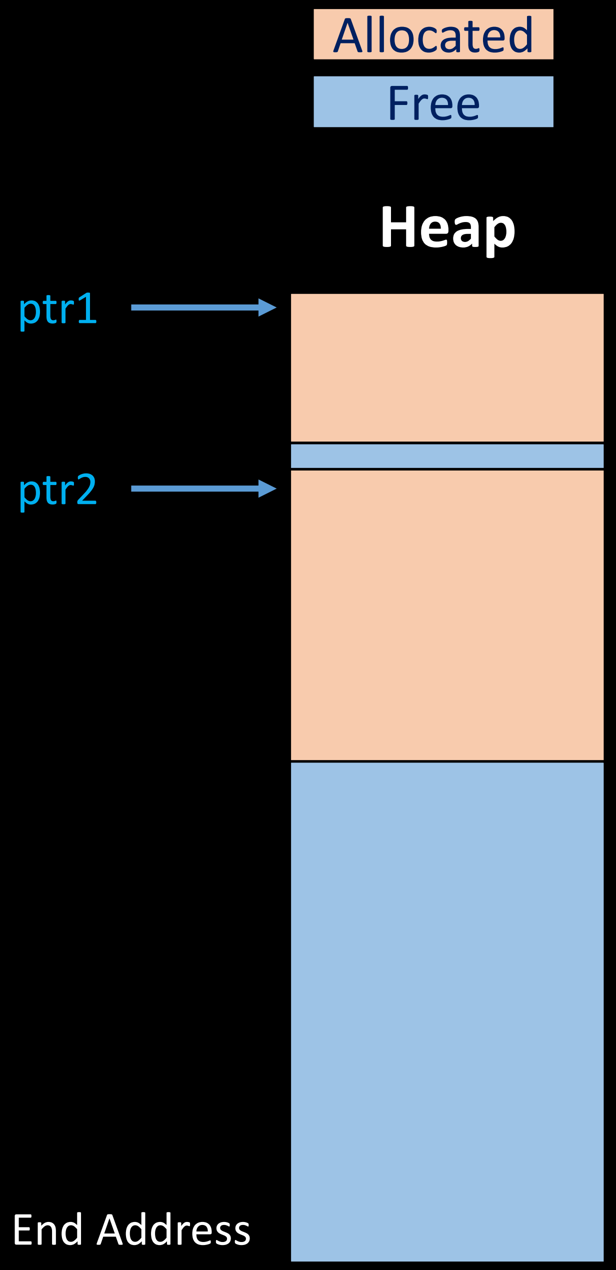


Heap Functions

- Malloc
- Calloc
- Realloc
- Free

Allocates ‘N‘ Contiguous bytes in Heap Space

Pointers identify and track location in Heap



```
char * ptr_TO_HEAP;
ptr1 = (char *)malloc(8);
```

8 Bytes Not Initialized

```
char * ptr2_TO_HEAP;
ptr2 = (char *)calloc(16, 1);
```

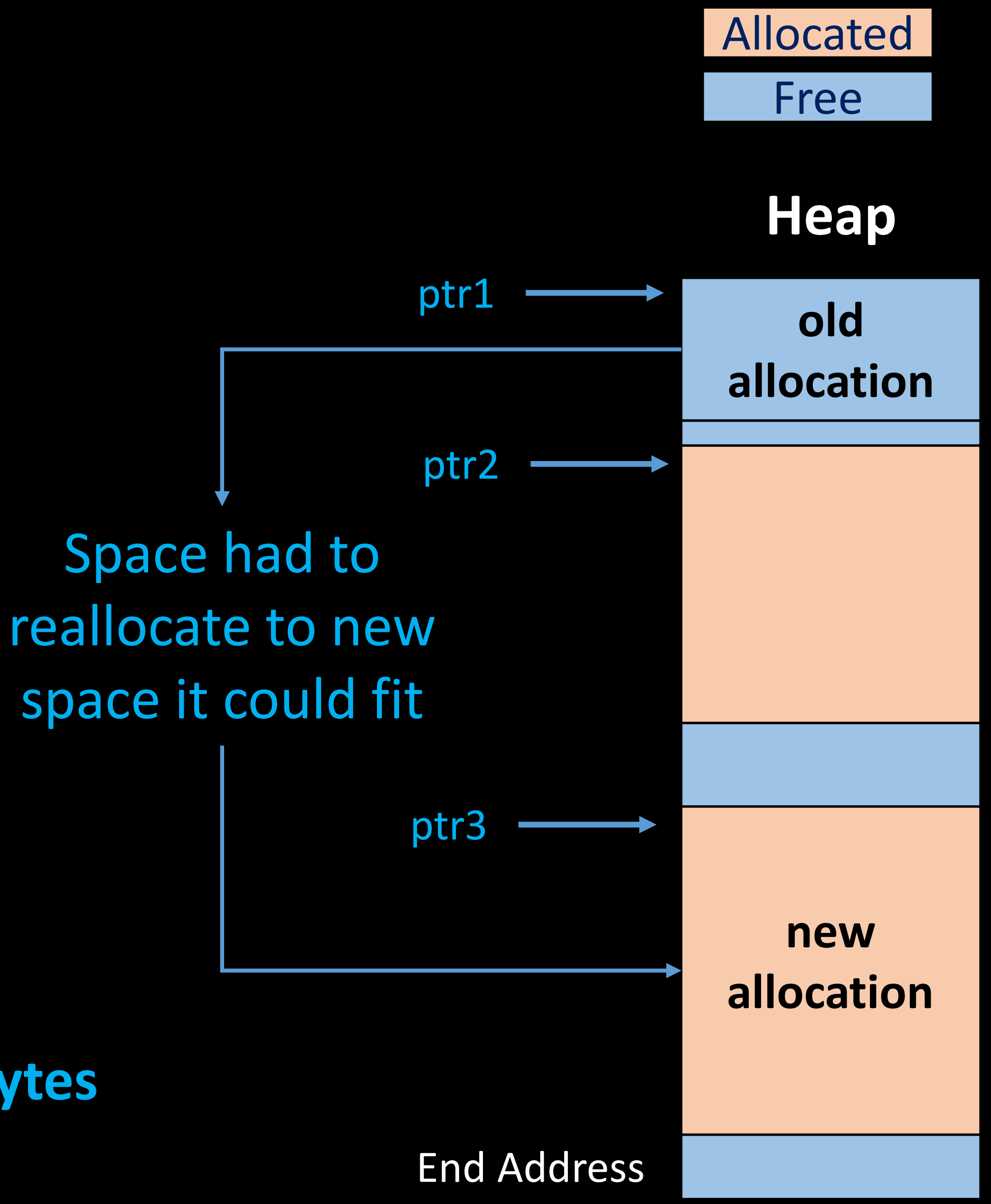
16 Bytes Initialized to Zero

Heap Functions

- Malloc
- Calloc
- **Realloc** → Reallocates region to new size, frees old space
- Free

```
char * ptr1;
ptr1 = (char *)malloc(8);
char * ptr2;
ptr2 = (char *)malloc(16);

char * ptr3;
ptr3 = (char *)realloc((void*)ptr1, 24);
```

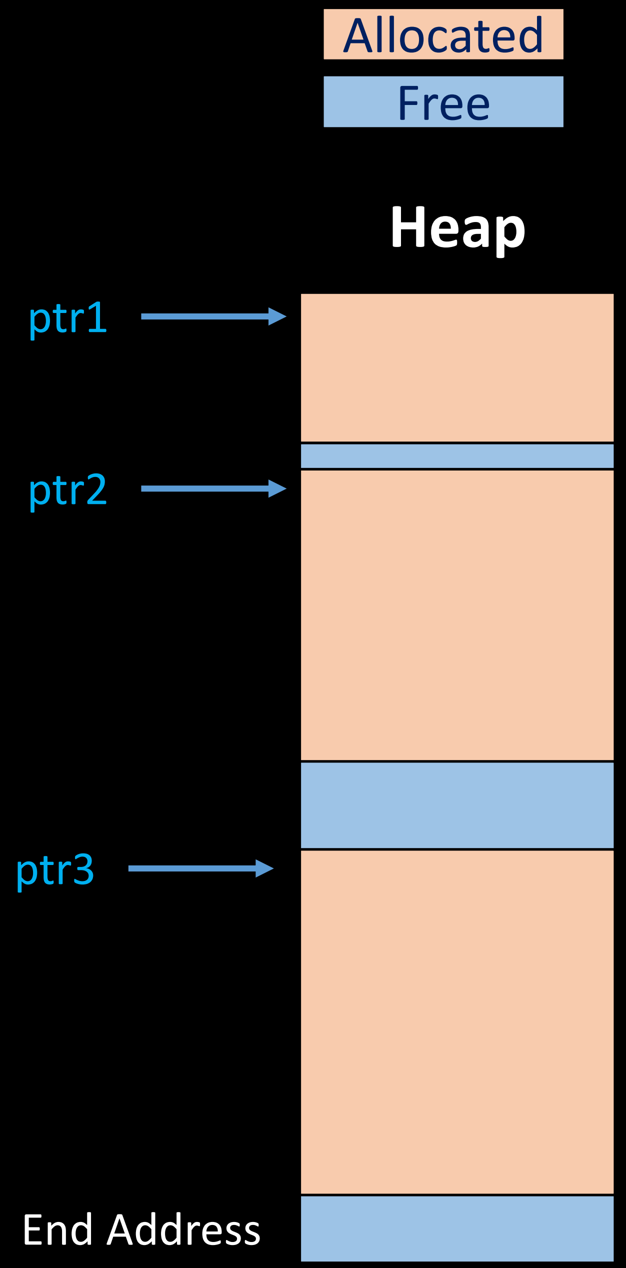
Resized to 24 Bytes



Heap Allocation

- Any Heap Allocation/Reallocation requires raw byte count and returns a pointer the beginning of the piece of memory requested

```
ptr1 = (char *) malloc(8);                        /* 8 Bytes */
ptr2 = (int *)  malloc(16);                       /* 4 Ints */
ptr3 = (float *)malloc(16);                       /* 4 Floats */
```



Heap Allocation

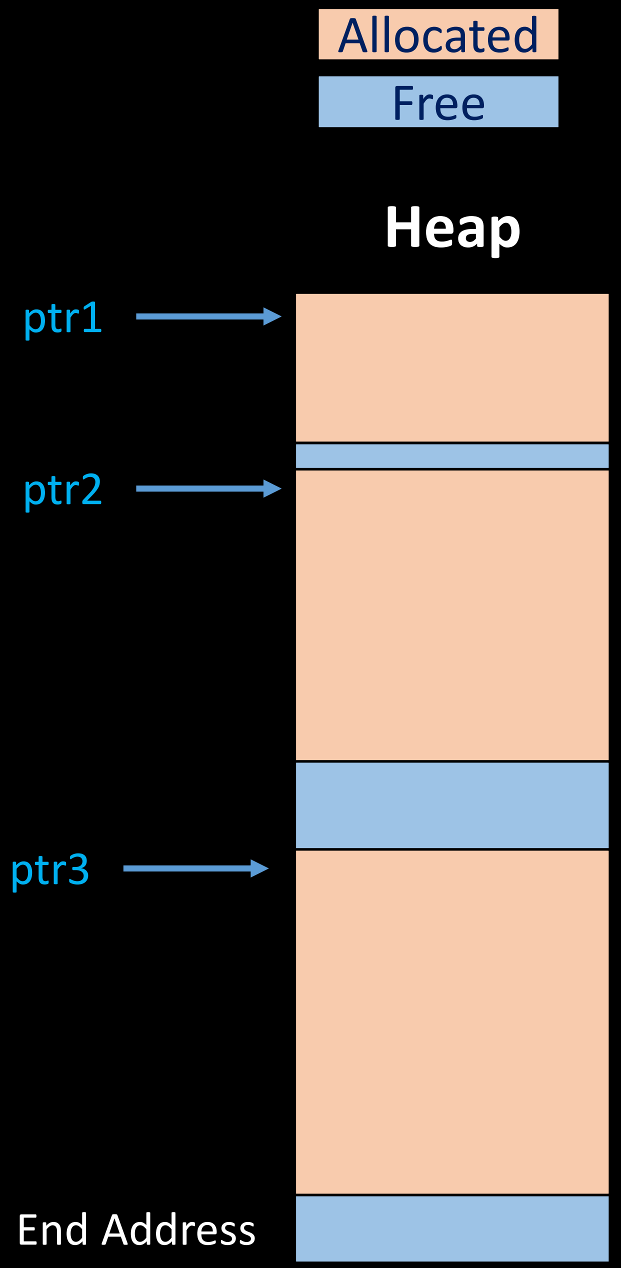
- Any Heap Allocation/Reallocation requires raw byte count and returns a pointer the beginning of the piece of memory requested

Pointer to location in Heap

Different Data Types

Raw # of Bytes

```
ptr1 = (char *)  malloc(8);                    /*  8 Bytes */
ptr2 = (int *)   malloc(16);                   /*  4 Ints */
ptr3 = (float *) malloc(16);                   /*  4 Floats */
```

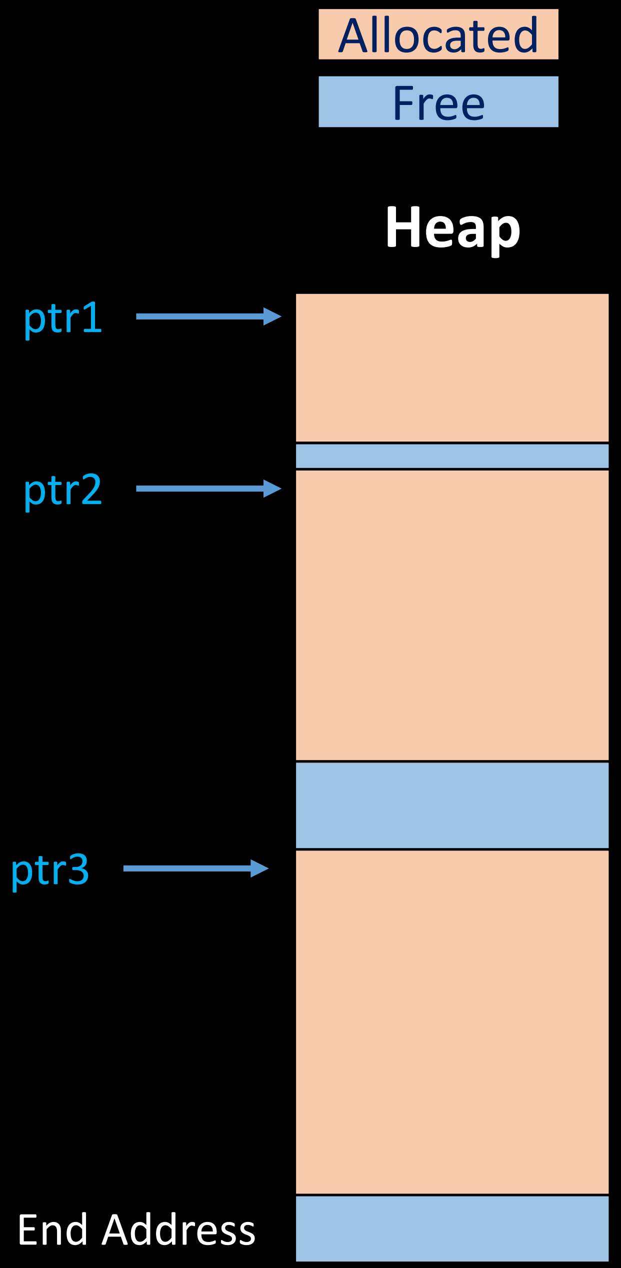


Heap Allocation

- Any Heap Allocation/Reallocation requires raw byte count and returns a pointer the beginning of the piece of memory requested

Sizeof allows to get the size of a type or structure

```
ptr1 = (char *) malloc(8*sizeof(char));   /* 8 Bytes */
ptr2 = (int *)  malloc(4*sizeof(int));    /* 4 Ints */
ptr3 = (float *)malloc(4*sizeof(float));  /* 4 Floats */
```



Heap Functions

- Malloc
- Calloc
- Realloc
- Free → Deallocates data back to Heap Free Space

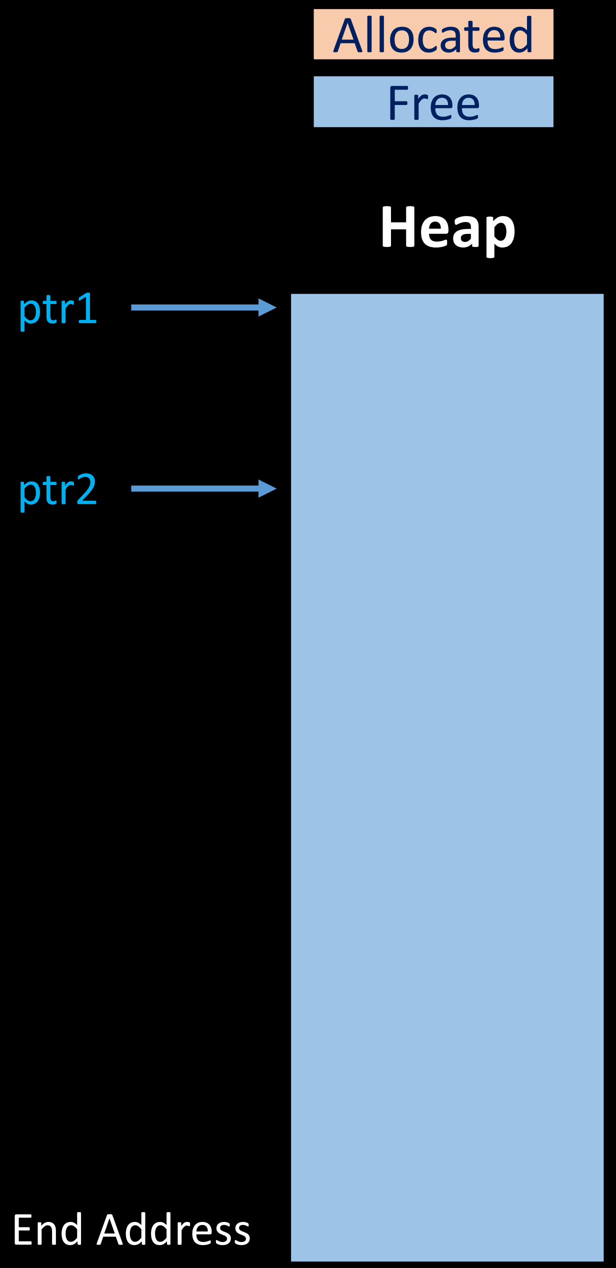
```
char * ptr1;
ptr1 = (char *)malloc(8);
free((void *)ptr1);
```

Frees 8 Bytes

```
char * ptr2;
ptr2 = (char *)calloc(16);
free((void *)ptr2);
```

Frees 16 Bytes

Pointers still point to location, but heap freed it



Failed Allocation

Allocated

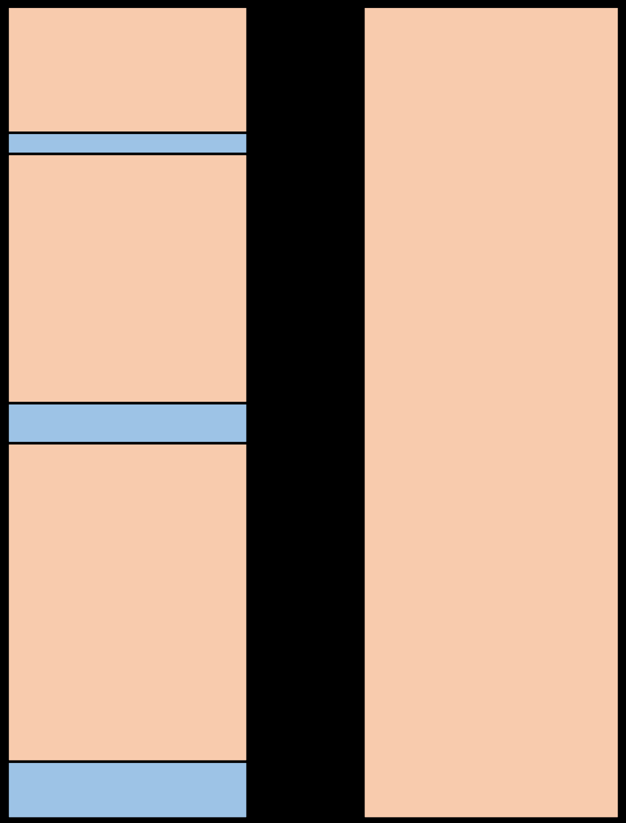
Free

- When dynamic memory allocaiton fails, routines return a NULL pointer (Pointer to nothing/address 0x0)

```
char * ptr1;
ptr1 = (char *)malloc(24);

if (ptr1 == NULL) {
   /* Allocation Failed!!! */
   /* …Handle Failure */
}
```

CANNOT assume that your allocation will work, need to check it succeeded!

Heap Examples



Not Enough Contiguous Space

Full

Malloc Example

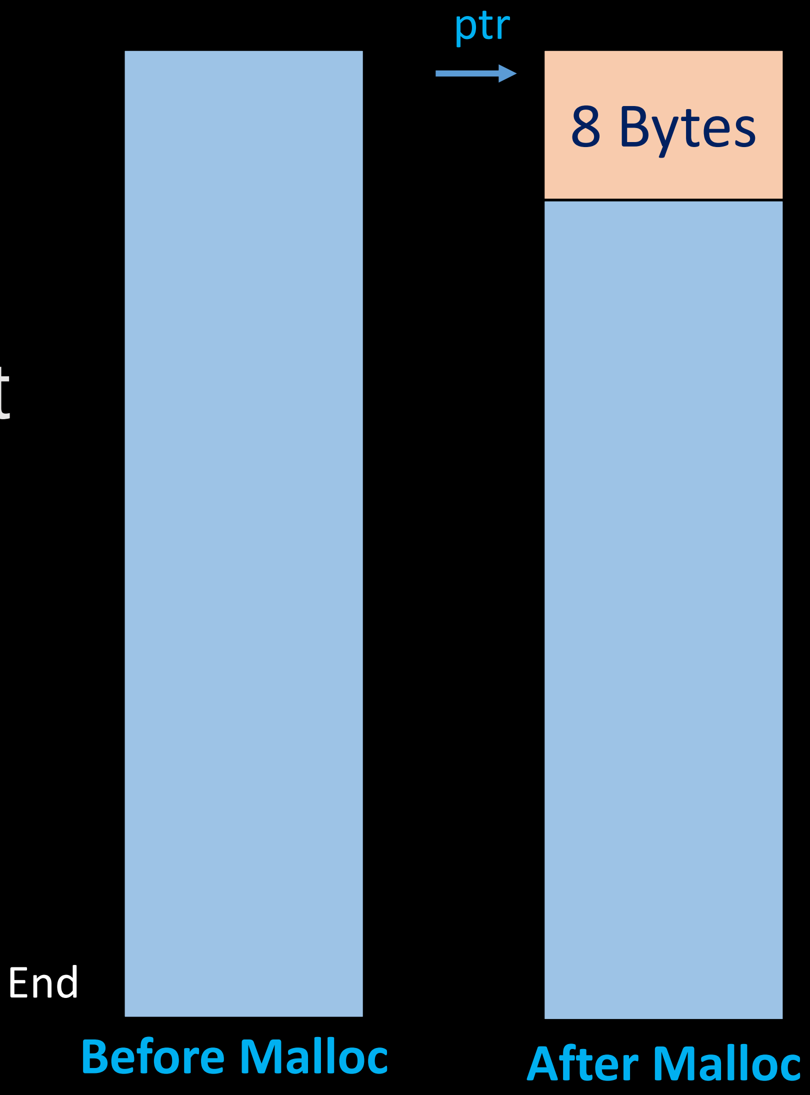
- Void Pointer = Generic Pointer, an address without a data type

```
void * ptr;    /* Generic Address Pointer */
```

- Cast generic pointer to type of data you want in heap region

```
char * ptr;
ptr = (char *)malloc(8*sizeof(char));

if (ptr == NULL) {
   /* Allocation Failed!!! */
   /* …Handle Failure */
}
```

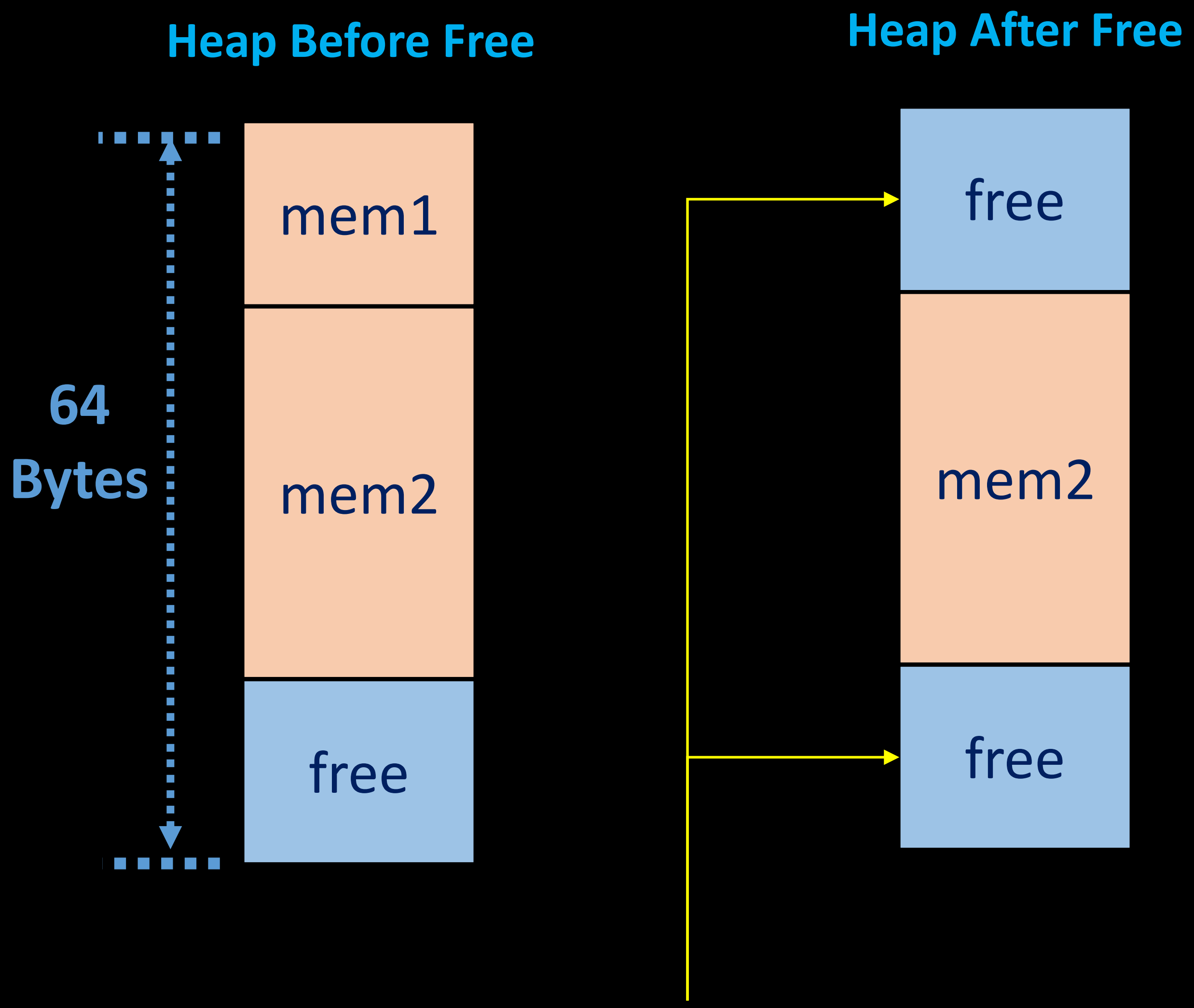
Heap States



Fragmentation

- Heap Size: 64 Bytes
- Memory Sizes:
  - mem1 – 16 Bytes
  - mem2 – 32 Bytes
  - Free space – 16 bytes

```
char * mem3;

free((char *) mem1); //deallocates

mem3 = (char *)malloc(32); //Fails!!!
```



32 Bytes are available, just not contiguously!

A Whole Heap of Issues

- Direct Software Management
- Potential Memory Leaks
  - Loss of heap tracking pointer
- Memory Fragmentation
- Performance Hit (extra CPU overhead)
  - Runtime allocated
  - Calling Heap functions to allocate memory