## Memory

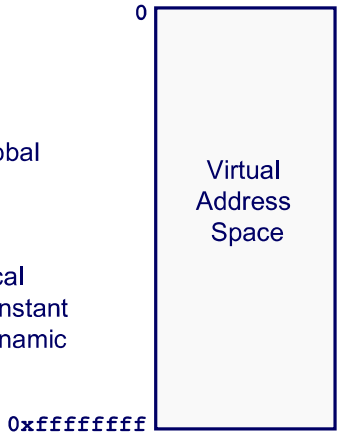
- What is stored in memory?


## Memory

- What is stored in memory?
  - Code
  - Constants
  - Global and static variables
  - Local variables
  - Dynamic memory (malloc)

```
int iSize;                  ← global

char *f(void)
{
     char *p;               ← local
     iSize = 8;             ← constant
     p = malloc(iSize);     ← dynamic
     return p;
}
```

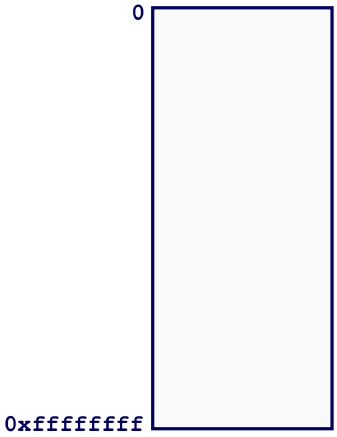


# Memory Layout

- How is memory organized?
  - Code
  - Constants
  - Global and static variables
  - Local variables
  - Dynamic memory (malloc)

```
int iSize;

char *f(void)
{
     char *p;
     iSize = 8;
     p = malloc(iSize);
     return p;
}
```

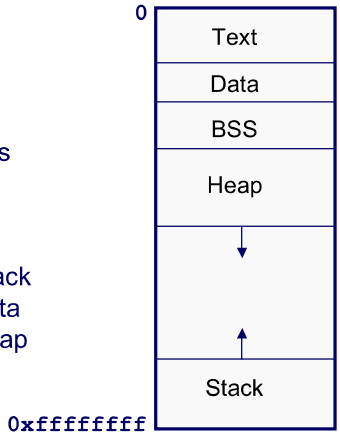



# Memory Layout

- How is memory organized?
  - Text = code
  - Data = constants
  - BSS = global and static variables
  - Stack = local variables
  - Heap = dynamic memory

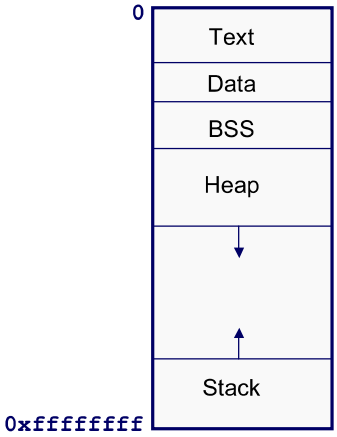
```
int iSize;                  ← bss

char *f(void)
{
     char *p;               ← stack
     iSize = 8;             ← data
     p = malloc(iSize);     ← heap
     return p;
}
```



# Memory Allocation

- How is memory allocated?
  - Global and static variables = program startup
  - Local variables = function call
  - Dynamic memory = malloc()




# Memory Allocation

```
int iSize;                  ← allocated in BSS, set to zero at startup

char *f(void)
{
    char *p;                ← allocated on stack at start of function f
    iSize = 8;
    p = malloc(iSize);      ← 8 bytes allocated in heap by malloc
    return p;
}
```

## Memory Deallocation

- How is memory deallocated?
  - Global and static variables = program finish
  - Local variables = function return
  - Dynamic memory = free()
- All memory is deallocated at program termination
  - It is good style to free allocated memory anyway

## Memory Deallocation

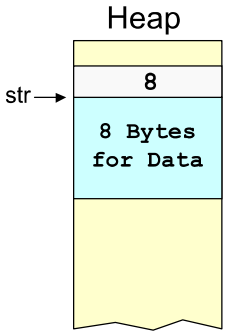
```
int iSize;                      ← available until program termination

char *f(void)
{
    char *p;                    ← deallocated by return from function f
    iSize = 8;
    p = malloc(iSize);          ← deallocate by calling free(p)
    return p;
}
```

# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

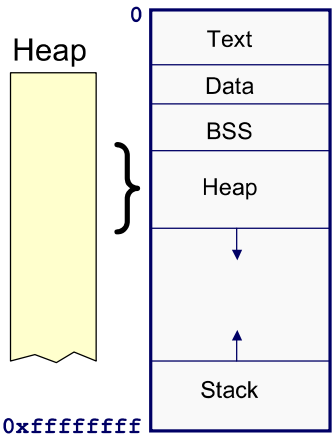
```
char *str = malloc(8);
...
free(str);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

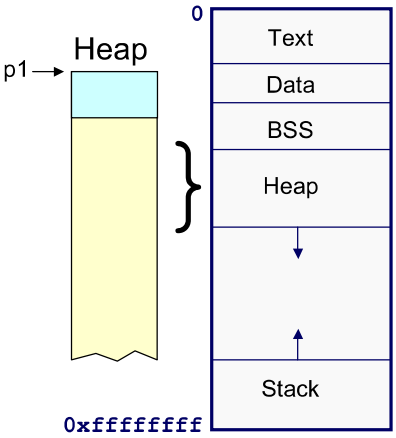
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```



# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

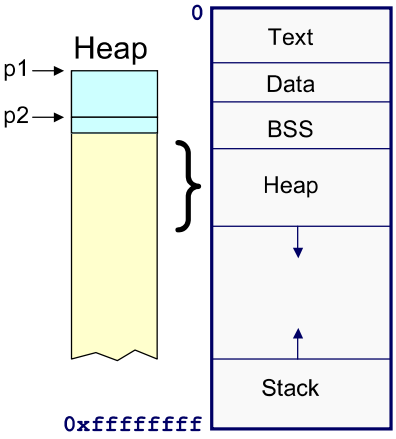
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

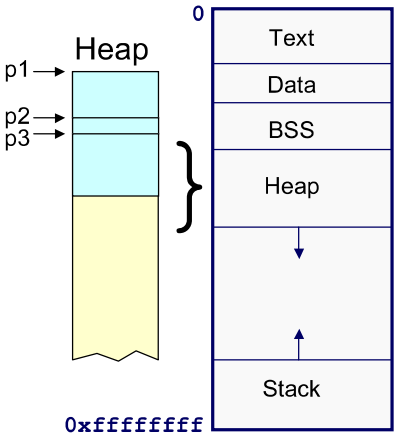
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```



# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

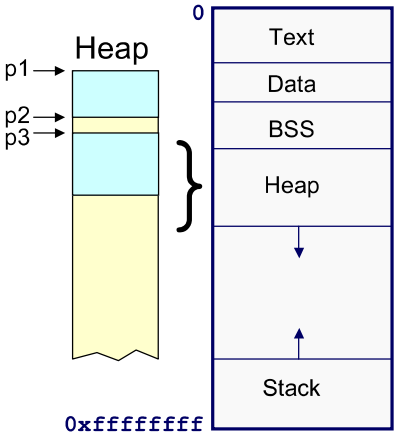
```
   char *p1 = malloc(3);
   char *p2 = malloc(1);
   char *p3 = malloc(4);
   free(p2);
   char *p4 = malloc(6);
   free(p3);
   char *p5 = malloc(2);
   free(p1);
   free(p4);
   free(p5);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

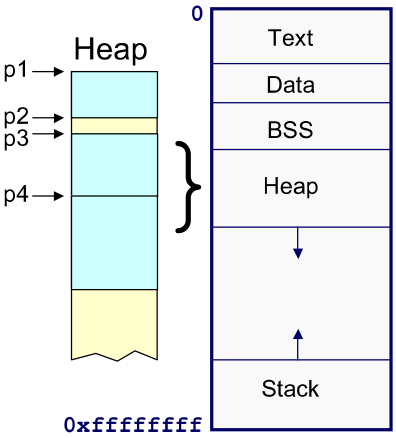
```
   char *p1 = malloc(3);
   char *p2 = malloc(1);
   char *p3 = malloc(4);
   free(p2);
   char *p4 = malloc(6);
   free(p3);
   char *p5 = malloc(2);
   free(p1);
   free(p4);
   free(p5);
```



# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

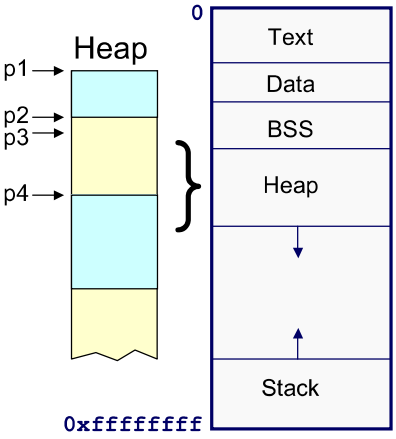
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

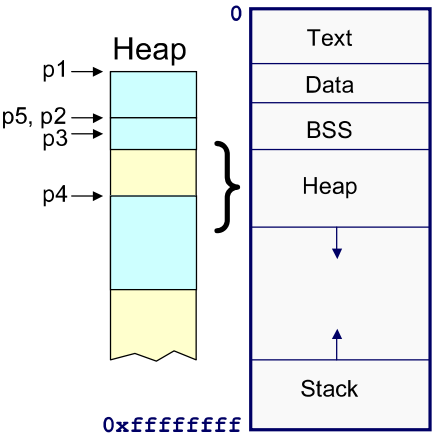
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```



# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

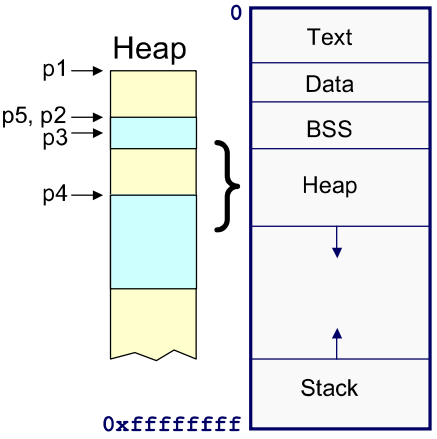
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

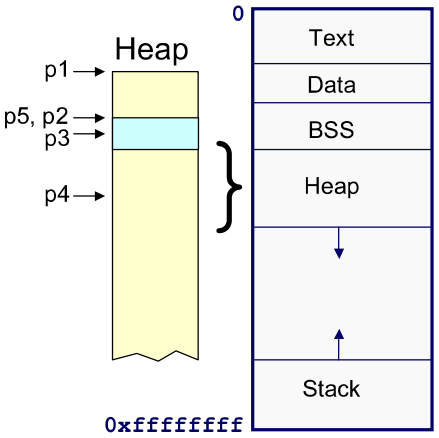
```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```



# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```




# Dynamic Memory

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
```

```
char *p1 = malloc(3);
char *p2 = malloc(1);
char *p3 = malloc(4);
free(p2);
char *p4 = malloc(6);
free(p3);
char *p5 = malloc(2);
free(p1);
free(p4);
free(p5);
```