Faculty of Computer Science & Engineering

Operating Systems

Contact:
Trần Ngọc Anh Tú
51304672@hcmut.edu.vn

Ho Chi Minh City
University of Technology

Lab 4 - Process

Objective

- Know how data are layout inside a process
- Understand dynamic memory allocation mechanism

Inside a process

- Each process has its own address space which is a set of memory addresses that a process can access.
- Address space is a large array of bytes starting at 0 and going up to some large number $2^{32} - 1$ or $2^{64} - 1$
- The address space is split into multiple parts, each part holds different parts of the process
  - Text
  - Data
  - Heap
  - Stack

Inside a process

- Exercise: Given following program, identify the memory segments which variables (or function) belongs to.

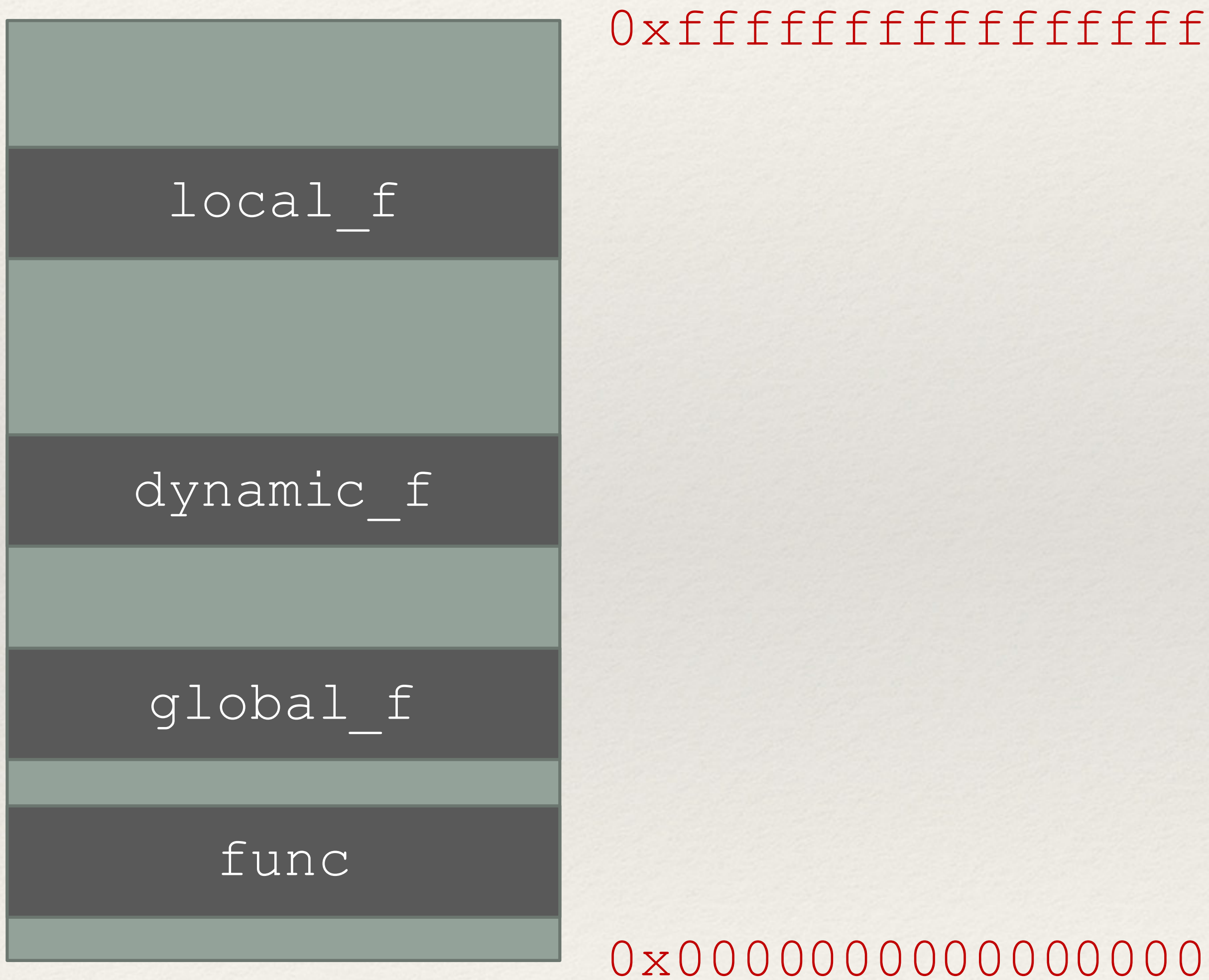
```
•global_f
•local_f
•func
•dynamic_f
```

```
#include <stdio.h>
#include <stdlib.h>
int global_f = 10;
int func() {
  return 0;
}
int main() {
  int local_f = 100;
  int *dynamic_f = (int*)malloc(sizeof(int));
  printf(“global_f:  %p\n”, &global_f);
  printf(“local_f:   %p\n”, &local_f);
  printf(“func:      %p\n”, &func);
  printf(“dynamic_f: %p\n”, dynamic_f);
}
```

Inside a process

- Exercise: Given following program, identify the memory segments which variables (or function) belongs to.

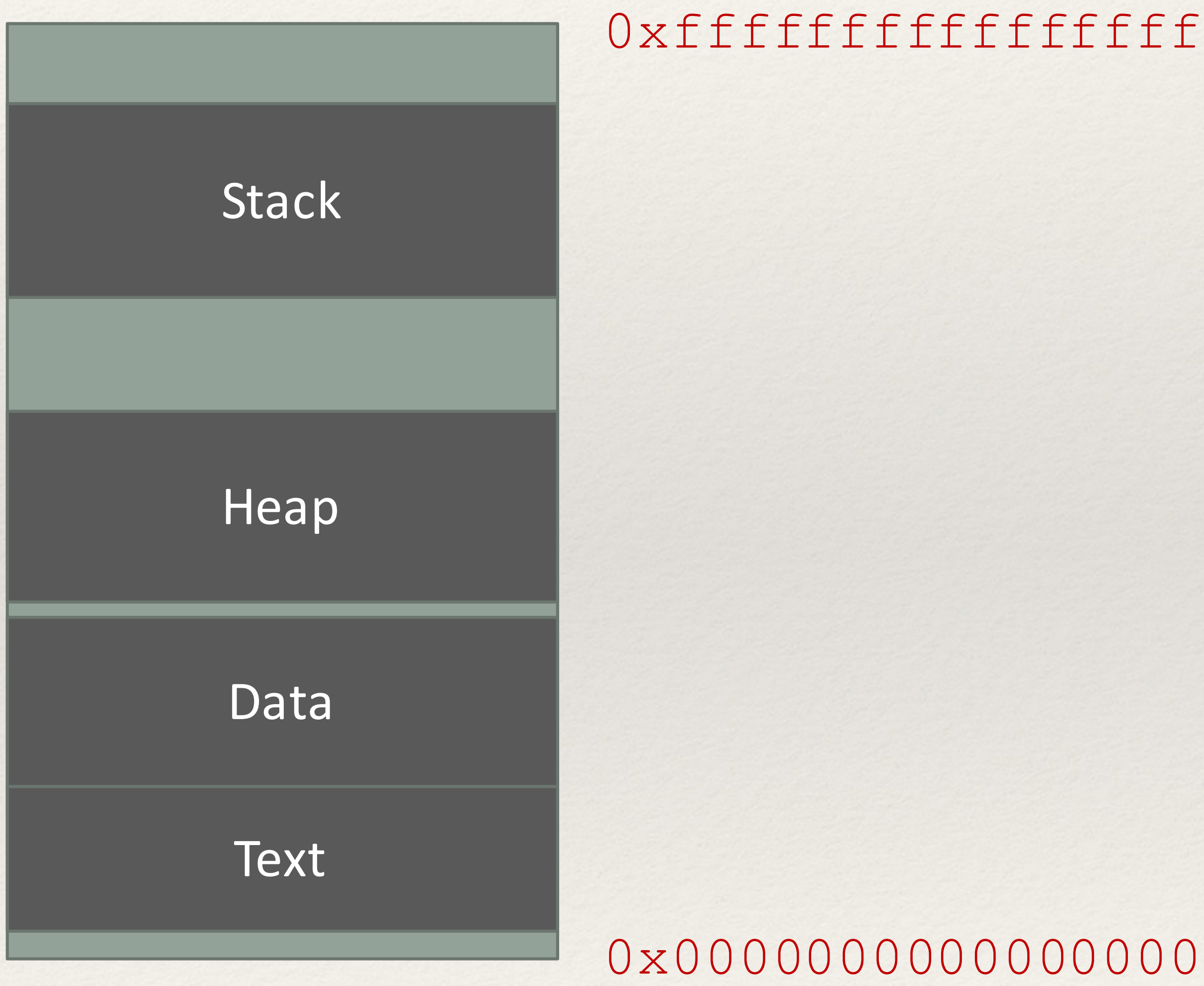
- global_f
- local_f
- func
- dynamic_f



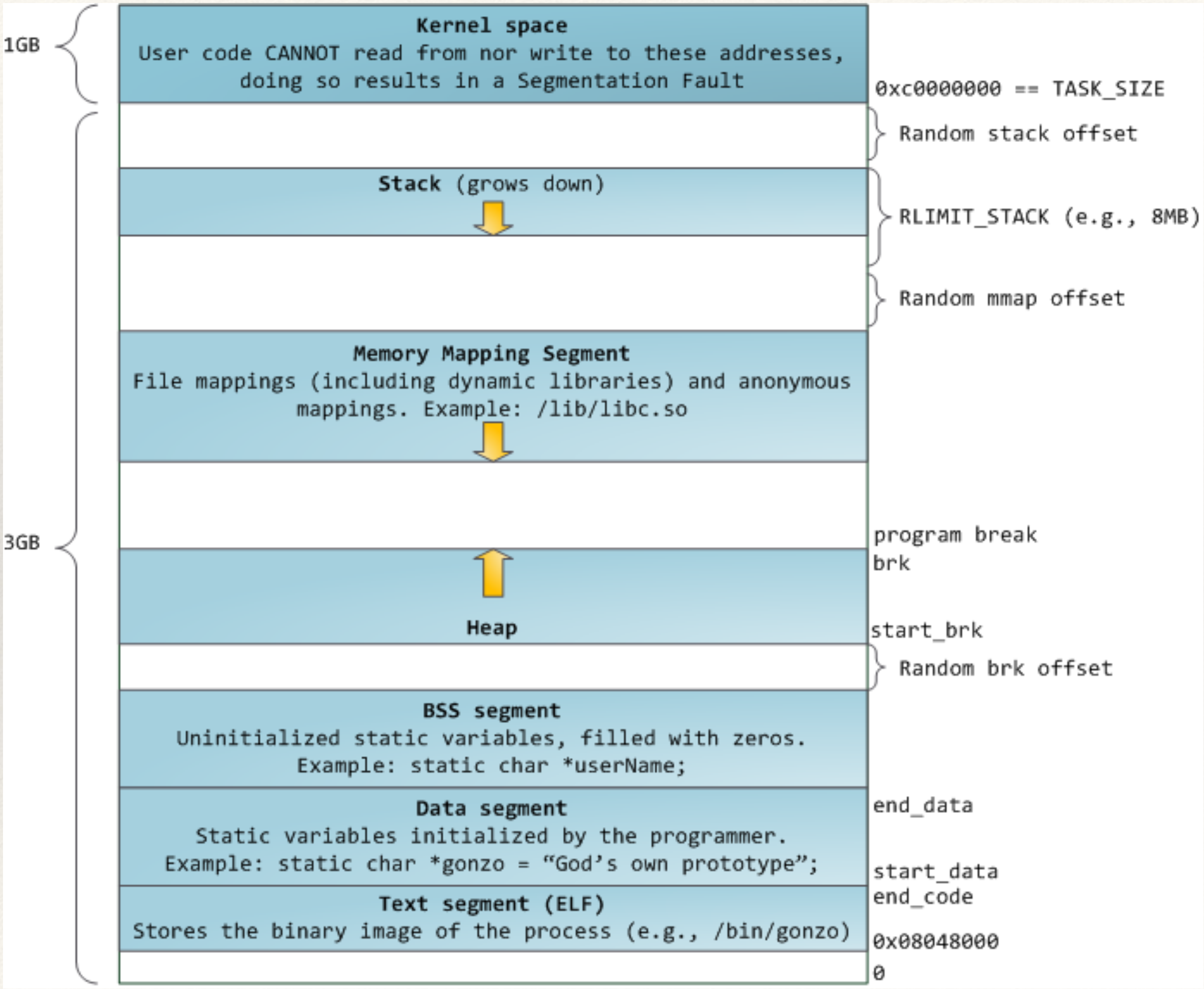
Inside a process

- Exercise: Given following program, identify the memory segments which variables (or function) belongs to.

- `global_f`
- `local_f`
- `func`
- `dynamic_f`

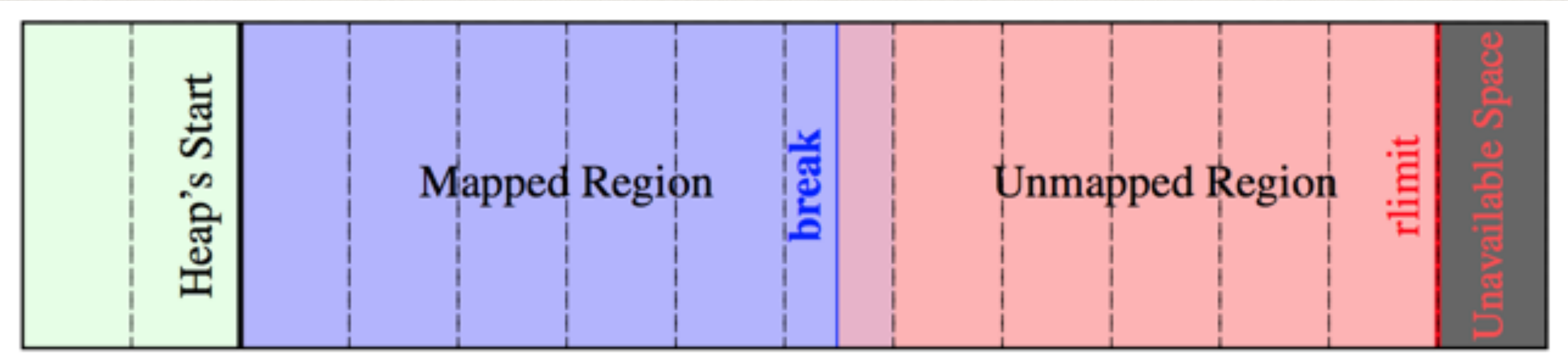


Process address space in Linux


1GB
Kernel space
User code CANNOT read from nor write to these addresses, doing so results in a Segmentation Fault
0xc0000000 == TASK_SIZE
Random stack offset
Stack (grows down)
RLIMIT_STACK (e.g., 8MB)
Random mmap offset
Memory Mapping Segment
File mappings (including dynamic libraries) and anonymous mappings. Example: /lib/libc.so
3GB
program break
brk
Heap
start_brk
Random brk offset
BSS segment
Uninitialized static variables, filled with zeros.
Example: static char *userName;
Data segment
Static variables initialized by the programmer.
Example: static char *gonzo = "God's own prototype";
end_data
start_data
end_code
Text segment (ELF)
Stores the binary image of the process (e.g., /bin/gonzo)
0x08048000
0

Heap region

- The heap is placed right after data region in address space.
- break (program break) separates used regions and unused regions.
- Heap increases upward but the break cannot go through rlimit



Heap region

- We could increase or decrease the size of heap region by using two system calls:
    - brk ->changes the value of break pointer.
    - sbrk -> increases the value of break by a given number of bytes.
- We could use those system calls to implement malloc and free.

Heap region

- A simple implementation of malloc

```
int *ptr = (int *) malloc(10 * sizeof (int));
if (ptr == NULL) {
} else {
    /* Allocation succeeded.  Do something.  */
    free(ptr);
    ptr = NULL;
}
```

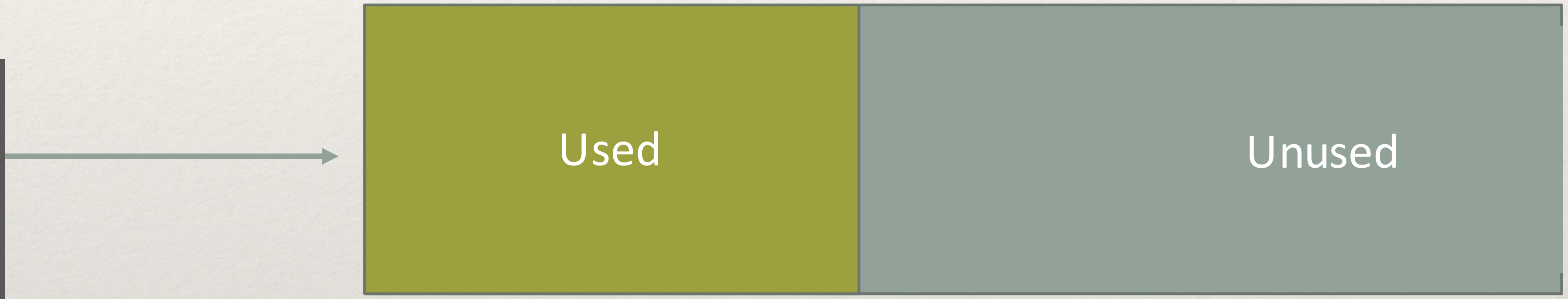
Dynamic allocation

- Actually, malloc does not allocate a new memory region whose size exactly equals to the size given by user. It allocates a few extra bytes to hold needed information (including the size of allocated region) so that free function could easily clean this region.

Dynamic allocation

- How to allocate and clean memory region on heap?

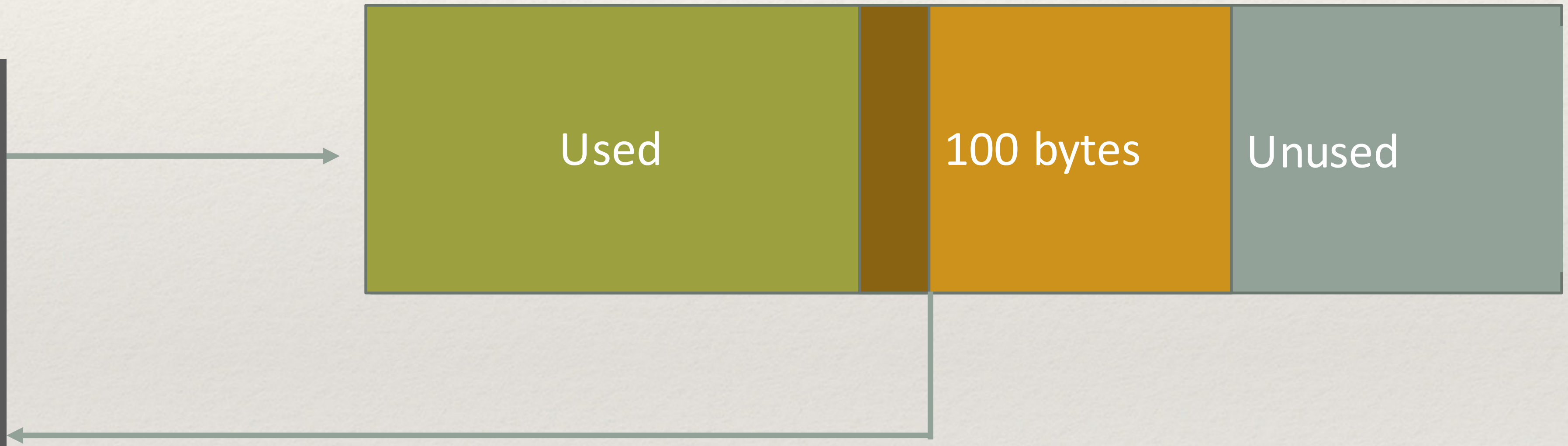
```
void *p;
p = malloc(100);

// do something



free(p);
```



Dynamic allocation

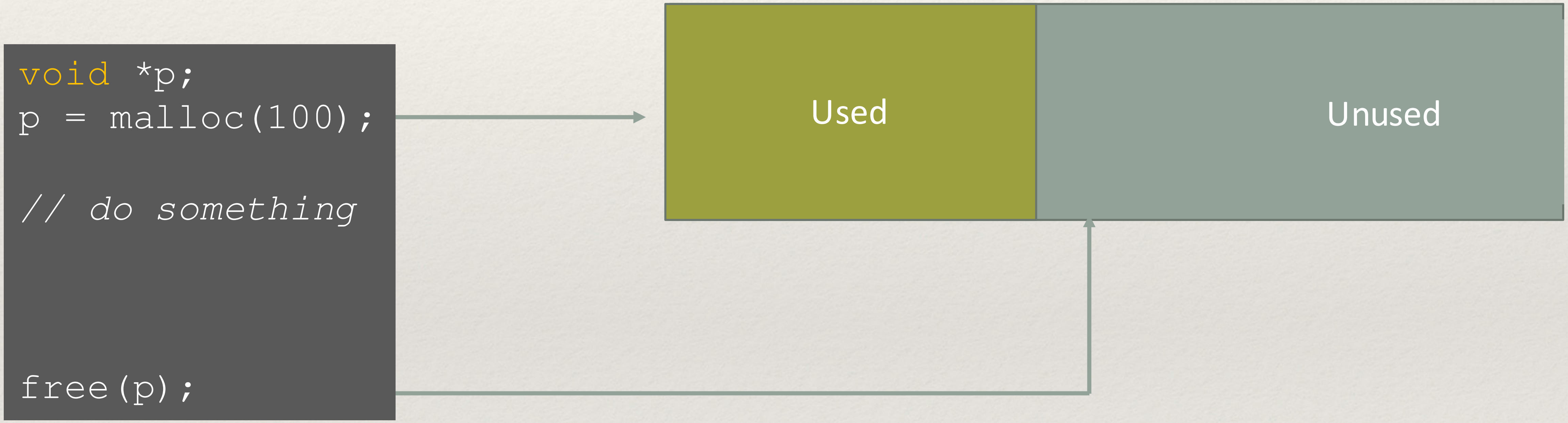
- How to allocate and clean memory region on heap?

```
void *p;
p = malloc(100);

// do something



free(p);
```



Dynamic allocation

- How to allocate and clean memory region on heap?



```
void *p;
p = malloc(100);

// do something



free(p);
```

Dynamic allocation

- Exercise: Implement your version of malloc and free using the technique described above.

End

Thanks!