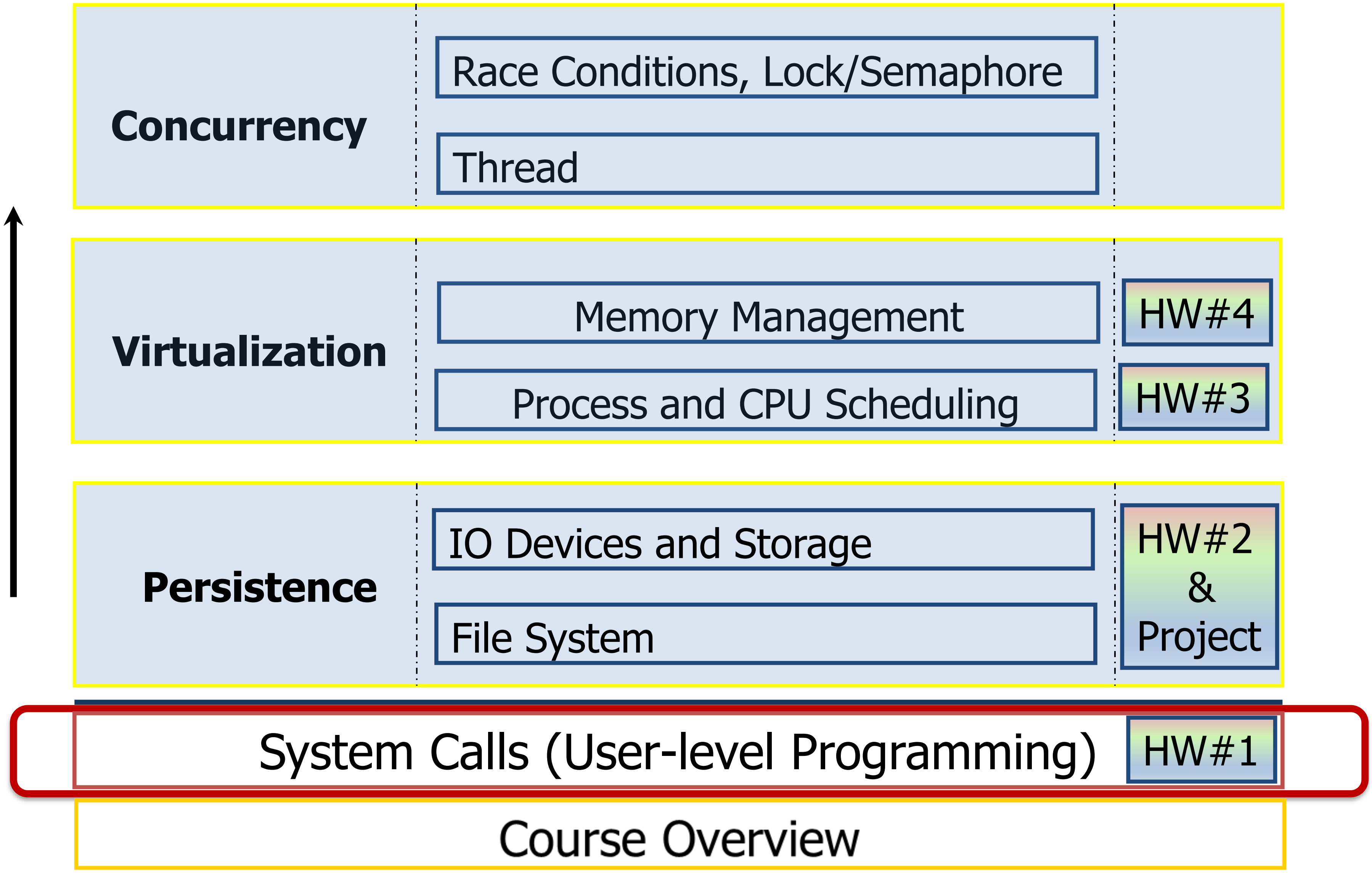
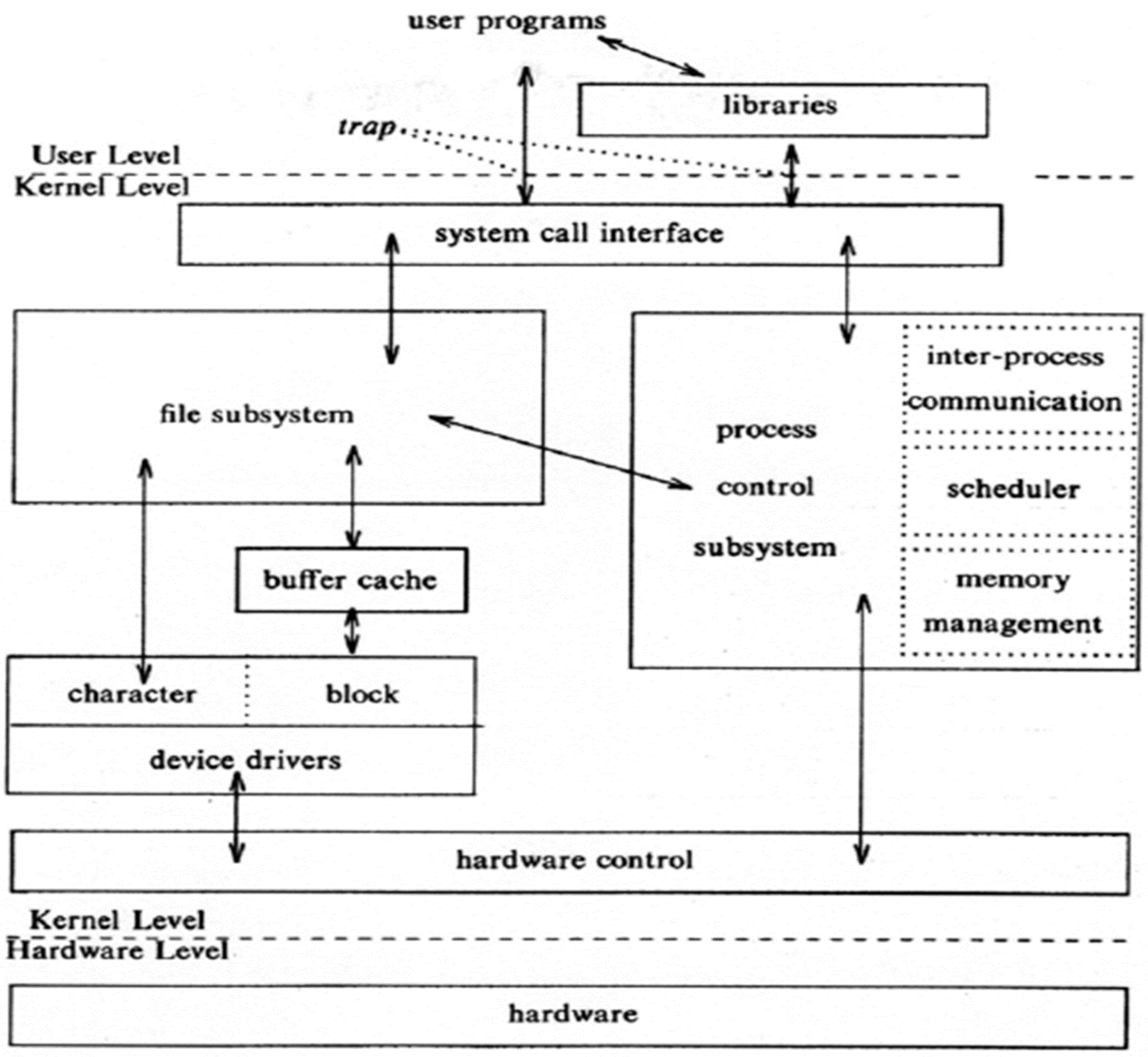
Lecture 3: User-level Programming via System Calls (Memory)

The Course Organization (Bottom-up)



System call

OS provides services via **System Call** (typically a few hundred) to run **process**, access memory/devices/files, etc.



The Design Of The Unix Operating System (Maurice Bach, 1986)

System Calls Related to Memory Allocation

- Several system calls

```
#include <unistd.h>

int brk(void *addr)
void *sbrk(intptr_t increment);
```

```
#include <sys/mman.h>

void *mmap(void *ptr, size_t length, int port, int flags,
int fd, off_t offset)
```

- User programming should not directly call them but use memory API (library functions).

Memory API: malloc()

```
#include <stdlib.h>

void* malloc(size_t size)
```

- Allocate a memory region on the heap.
  - Argument
    - `size_t size` : size of the memory block(in bytes)
    - `size_t` is an unsigned integer type.
  - Return
    - Success : a void type pointer to the memory block allocated by `malloc`
    - Fail : a null pointer

sizeof()

- Routines and macros are utilized for `size` in `malloc` instead typing in a number directly.
- Two types of results of `sizeof` with variables
  - The actual size of `'x'` is known at run-time.

```
int *x = malloc(10 * sizeof(int));
printf("%d\n", sizeof(x));
```

```
4
```

  - The actual size of `'x'` is known at compile-time.

```
int x[10];
printf("%d\n", sizeof(x));
```

```
40
```

Memory API: free()

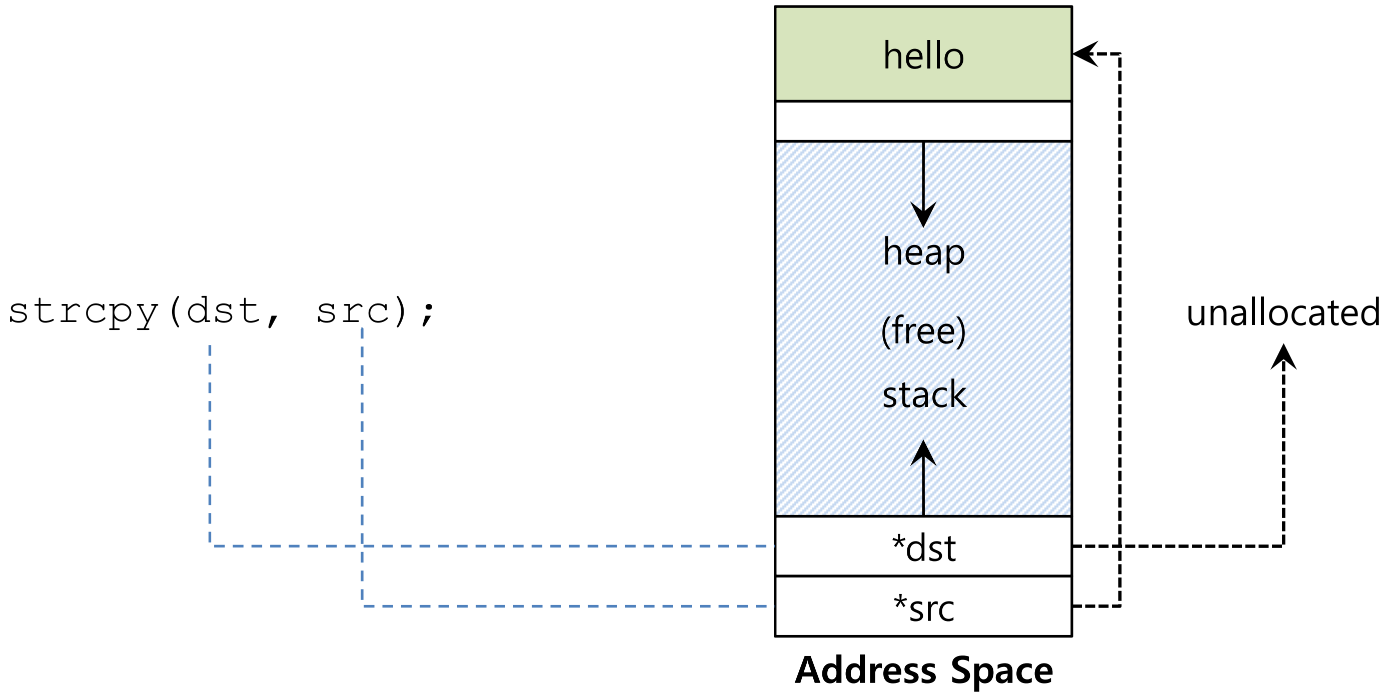
```
#include <stdlib.h>

void free(void* ptr)
```

- Free a memory region allocated by a call to `malloc`.
  - Argument
    - `void *ptr` : a pointer to a memory block allocated with `malloc`
  - Return
    - none

Common Error: Forgetting To Allocate Memory

- Incorrect code

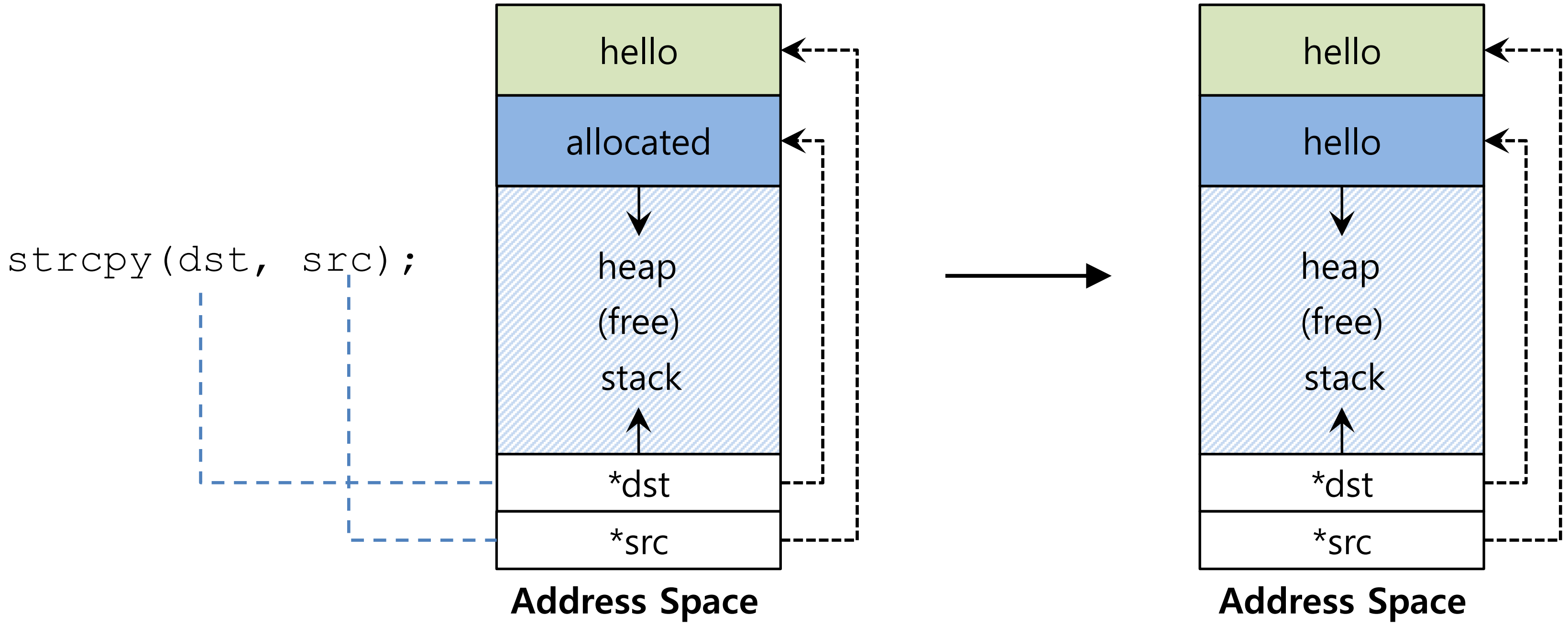
```
char *src = "hello"; //character string constant
char *dst;           //unallocated
strcpy(dst, src);    //segfault and die
```



Common Error: Forgetting To Allocate Memory(Cont.)

- Correct code

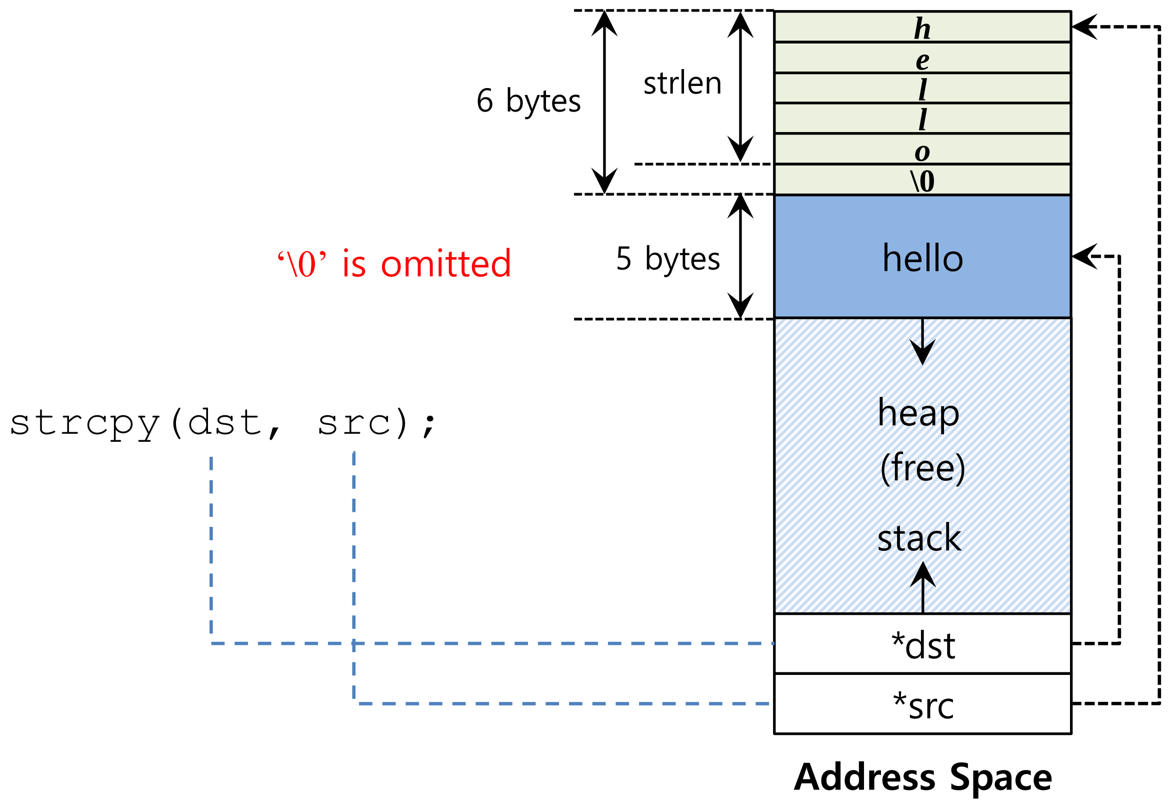
```
char *src = "hello";   //character string constant
char *dst (char *)malloc(strlen(src) + 1 ); // allocated
strcpy(dst, src);      //work properly
```



Common Error: Not Allocating Enough Memory

- Incorrect code, but work properly

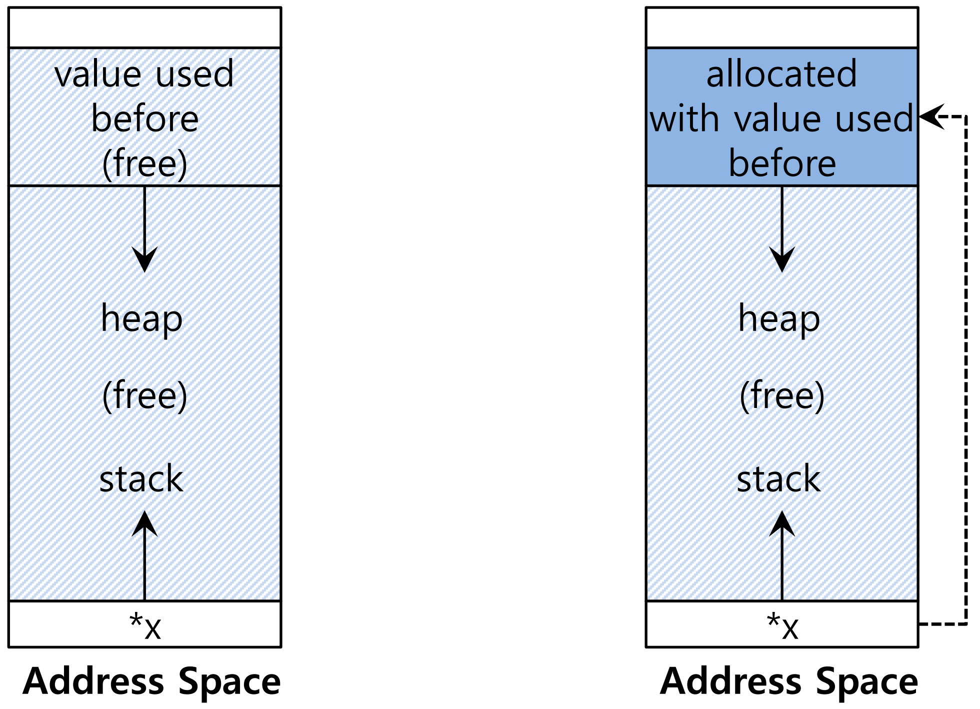
```
char *src = "hello"; //character string constant
char *dst (char *)malloc(strlen(src)); // too small
strcpy(dst, src);    //work properly
```



Common Error: Forgetting to Initialize

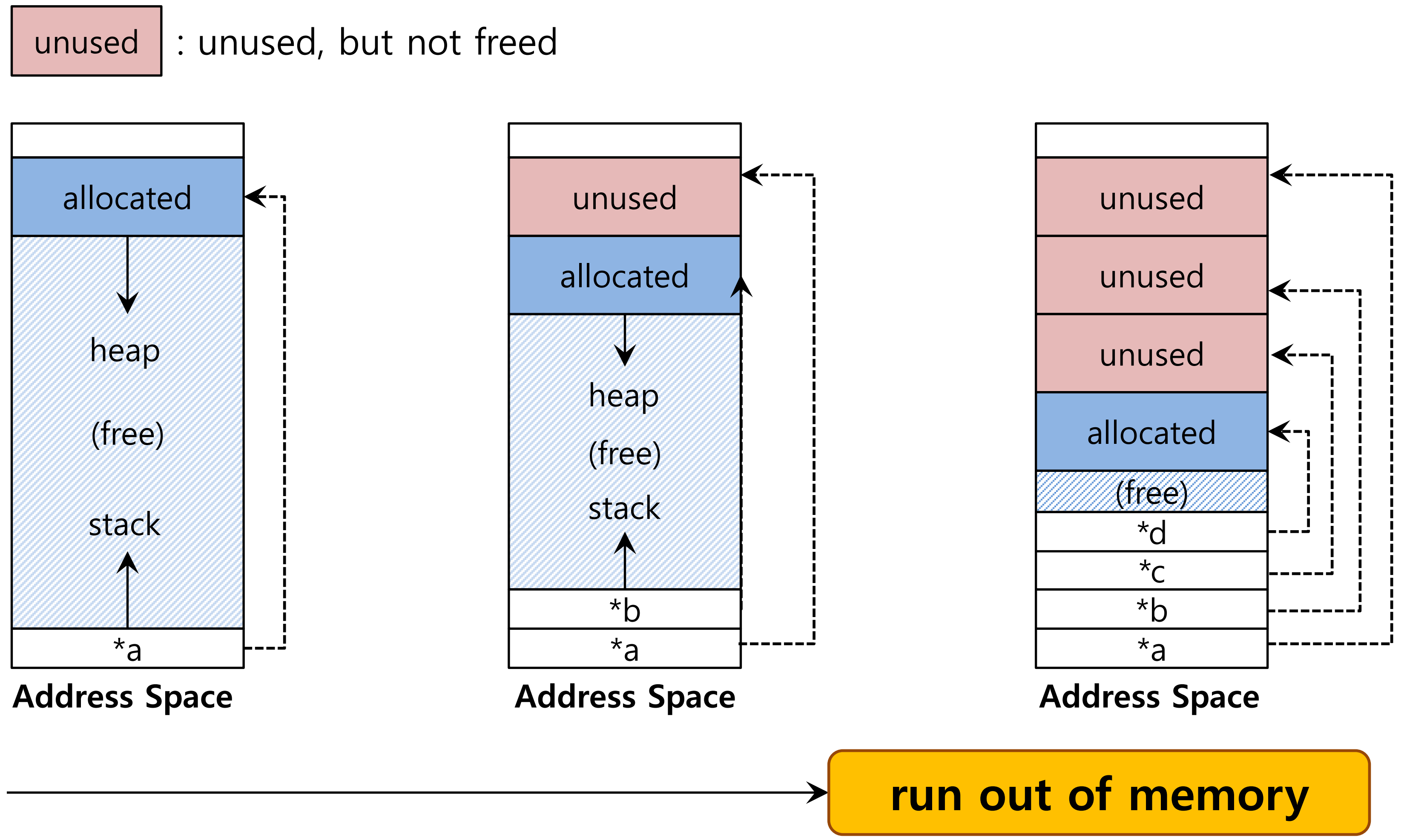
- Encounter an uninitialized read

```
int *x = (int *)malloc(sizeof(int)); // allocated
printf("*x = %d\n", *x); // uninitialized memory access
```



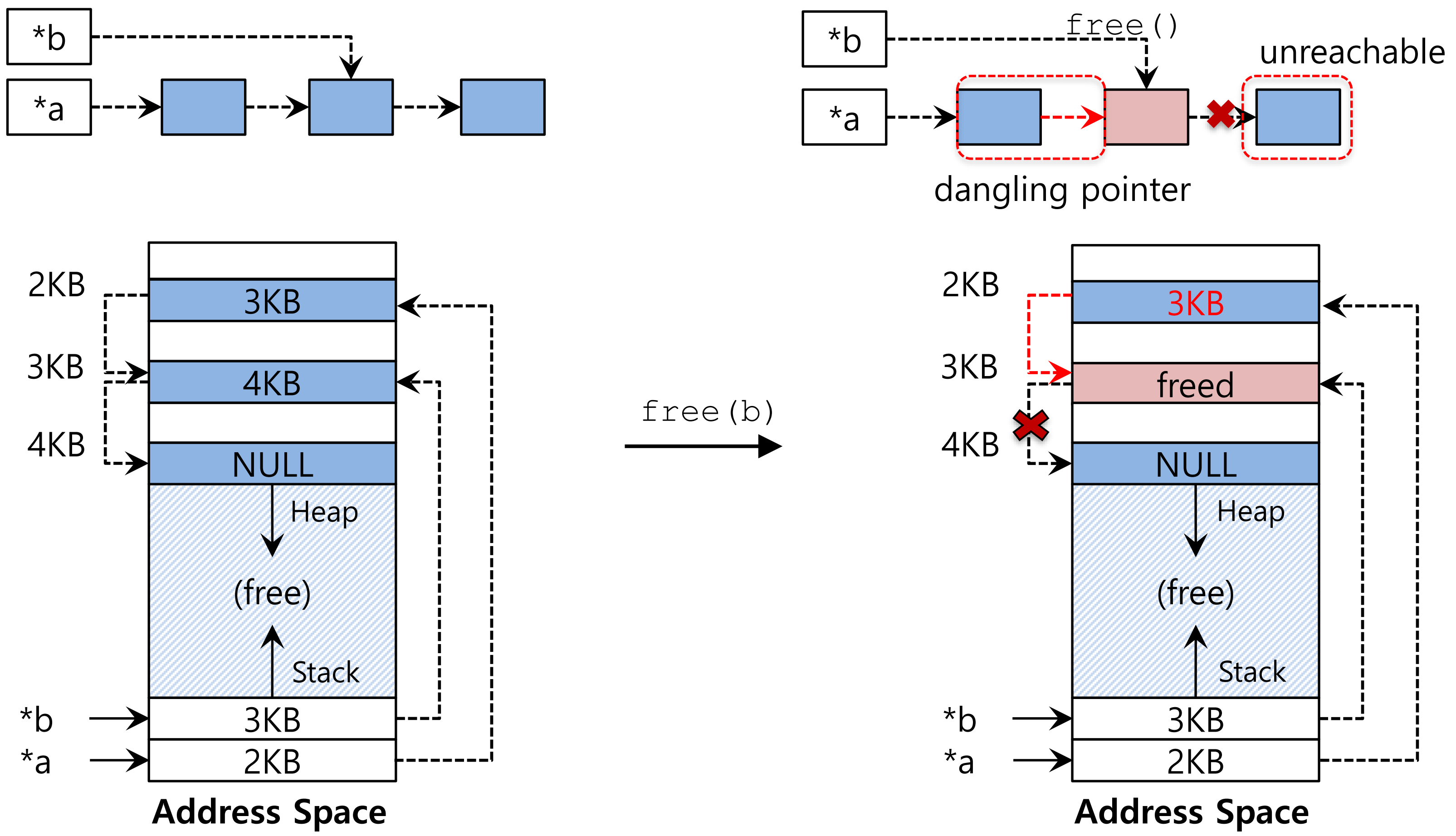
Common Error: Memory Leak

- A program runs out of memory and eventually dies.



Common Error: Dangling Pointer

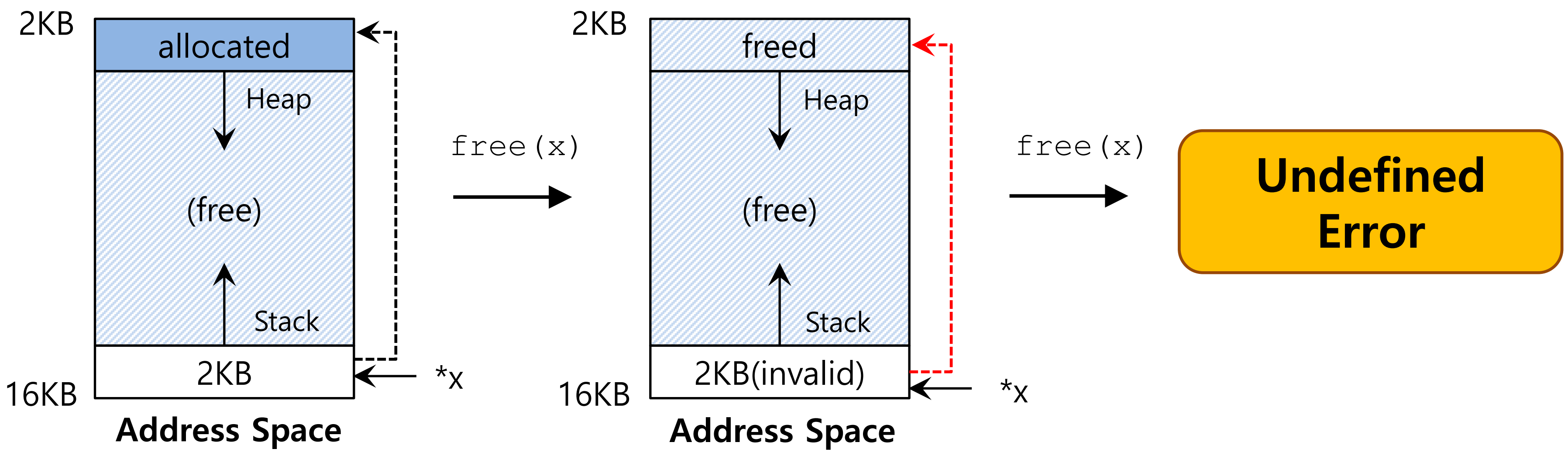
- Freeing memory before it is finished using
  - A program accesses to memory with an invalid pointer



Common Error: Double Free

- Free memory that was freed already.

```
int *x = (int *)malloc(sizeof(int)); // allocated
free(x); // free memory
free(x); // free repeatedly
```



Other Memory APIs: calloc()

```
#include <stdlib.h>

void *calloc(size_t num, size_t size)
```

- Allocate memory on the heap and set with zero before returning.
  - Argument
    - `size_t num` : number of blocks to allocate
    - `size_t size` : size of each block(in bytes)
  - Return
    - Success : a void type pointer to the memory block allocated by `calloc`
    - Fail : a null pointer

Other Memory APIs: realloc()

```
#include <stdlib.h>

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

- Change the size of memory block.
  - A pointer returned by `realloc` may be either the same as `ptr` or a new.
  - Argument
    - `void *ptr`: Pointer to memory block allocated with `malloc`, `calloc` or `realloc`
    - `size_t size`: New size for the memory block(in bytes)
  - Return
    - Success: Void type pointer to the memory block
    - Fail : Null pointer

Summary

- Functions related to memory allocation
  - Allocation: malloc()
  - Free: free()
  - Allocation & Initialization: calloc()
  - Re-allocation: realloc()
- Next: User-level programming using system calls (file & Directory)
  - Chapter 39 (Files)