13. The Abstraction: Address Space

**Operating System: Three Easy Pieces**

Memory Virtualization

- What is **memory virtualization**?

    - OS virtualizes its physical memory.

    - OS provides an illusion memory space per each process.

    - It seems to be seen like each process uses the whole memory .

Benefit of Memory Virtualization

- Ease of use in programming

- Memory efficiency in terms of times and space

- The guarantee of isolation for processes as well as OS

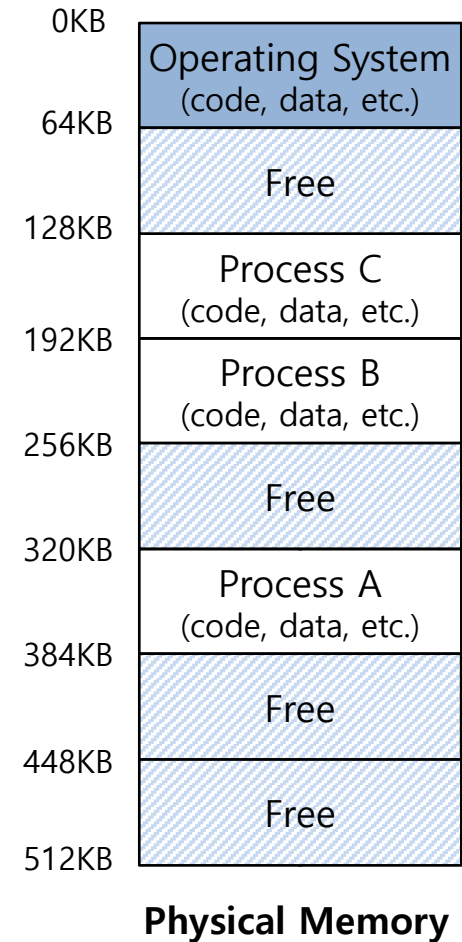   - Protection from errant accesses of other processes

- Load only one process in memory.

  - Poor utilization and efficiency
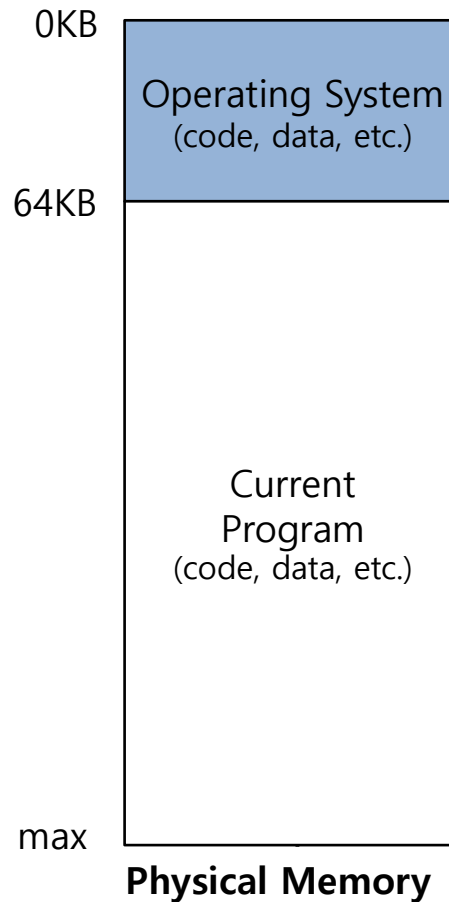
|	
0KB	
Operating System (code, data, etc.)	
64KB	
Current Program (code, data, etc.)	Perhaps in the future too: Unikernels)
max	

Physical Memory

Multiprogramming and Time Sharing

◻ **Load multiple processes** in memory.

- ◆ Execute one for a short while.

- ◆ Switch processes between them in memory.

- ◆ Increase utilization and efficiency.

◻ Cause an important **protection issue**.

- ◆ Errant memory accesses from other processes

0KB	Operating System (code, data, etc.)
64KB	Free
128KB	Process C (code, data, etc.)
192KB	Process B (code, data, etc.)
256KB	Free
320KB	Process A (code, data, etc.)
384KB	Free
448KB	Free
512KB	

Physical Memory

- OS creates an **abstraction** of physical memory.

  - The address space contains all about a running process.

  - That is consist of program code, heap, stack and etc.

|	
0KB	Program Code
1KB	Heap
2KB	↓
	(free)
	↑
15KB	Stack
16KB	
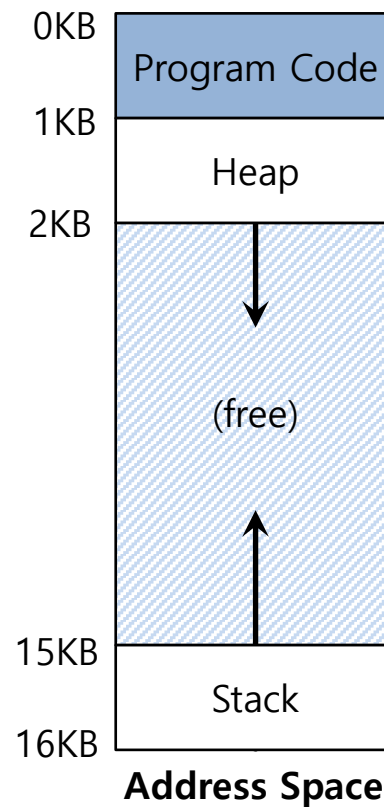
Address Space

- ## Program

  - ◆ Where instructions and static data live

- ## Heap

  - ◆ Dynamically allocate memory.

    - ○ `malloc/free` in C language

    - ○ `new/delete` in object-oriented language

  - ◆ Implicitly handled in memory managed languages

- ## Stack

  - ◆ Store return addresses or values.

  - ◆ Contain local variables arguments to routines.

  - ◆ Implicitly handled in HLL

Program
Heap
(free)
Stack
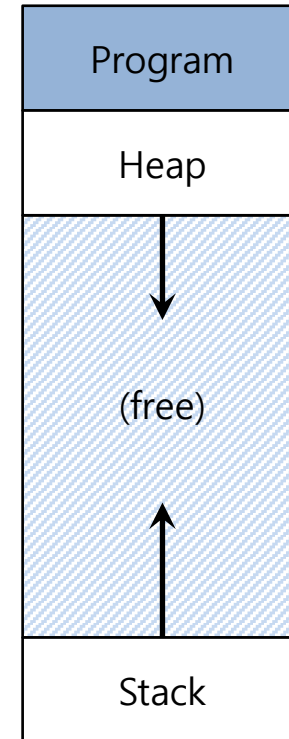
Address Space

Virtual Address

□ **Every address** in a running program is virtual.

 ◆ OS provides the "mechanims" to translate the virtual address to physical address

 ◆ HW assists to make such translation painless

```c
#include <stdio.h>
#include <stdlib.h>
int global=1;
int main(int argc, char *argv[])
{
    int x = 3;
    printf("location of code  : %p\n", (void *) main);
    printf("location of data  : %p\n", (void *) &global);
    printf("location of heap  : %p\n", (void *) malloc(1));
    printf("location of stack : %p\n", (void *) &x);

    return x;
}
```
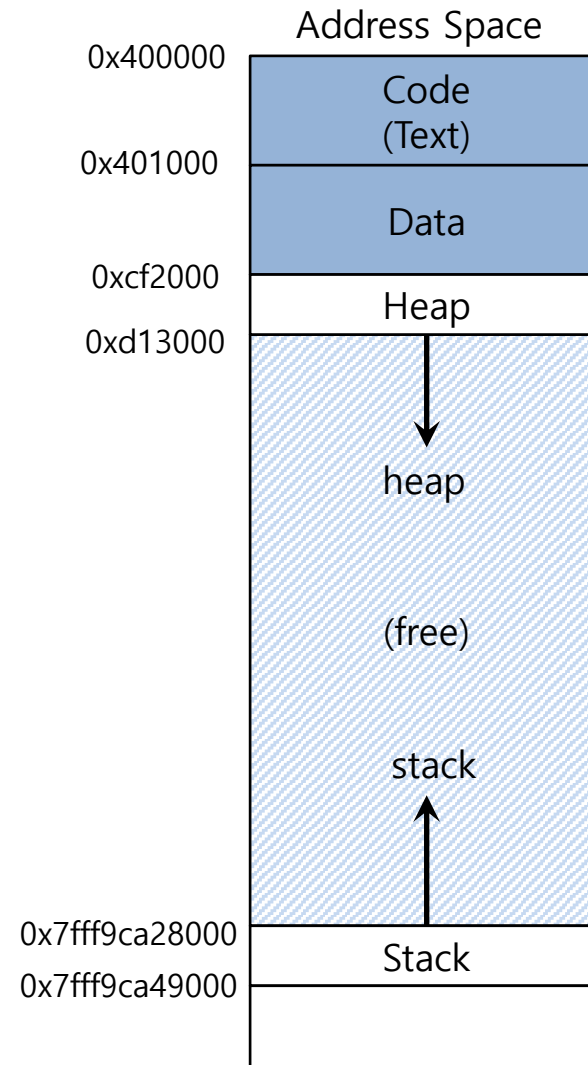
A simple program that prints out addresses

□ The output in 64-bit Linux machine

```
location of code  : 0x40057d
location of data  : 0x401010
location of heap  : 0xcf2010
location of stack : 0x7fff9ca45fcc
```

Address Space

0x400000	Code (Text)
0x401000	Data
0xcf2000	Heap
0xd13000	
	heap
	(free)
	stack
0x7fff9ca28000	Stack
0x7fff9ca49000	

□ Disclaimer: Disclaimer: This lecture slide set is used in AOS course at University of Cantabria. Was initially developed for Operating System course in Computer Science Dept. at Hanyang University. This lecture slide set is for OSTEP book written by Remzi and Andrea Arpaci-Dusseau (at University of Wisconsin)