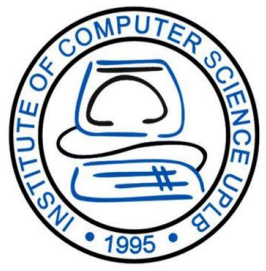
CMSC 125: Operating Systems

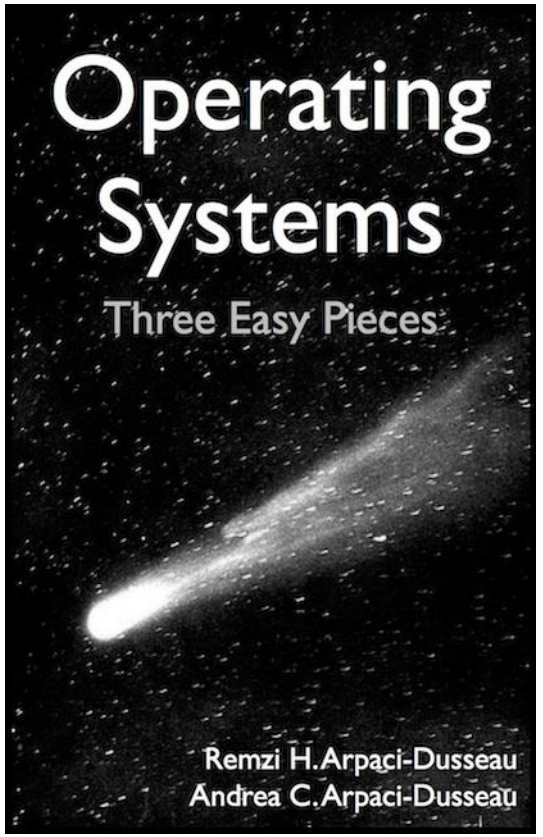
- Instructor: **Joseph Anthony C. Hermocilla**
- Email: jchermocilla@up.edu.ph
- Web: https://jachermocilla.org



Resources

Book: https://pages.cs.wisc.edu/~remzi/OSTEP/

Slides Template:
https://pages.cs.wisc.edu/~remzi/OSTEP/Educators-Slides/Youjip/



Acknowledgement

- This lecture slide set 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 at University of Wisconsin.

13. The Abstraction: Address Space

Operating System: Three Easy Pieces

Memory Virtualization

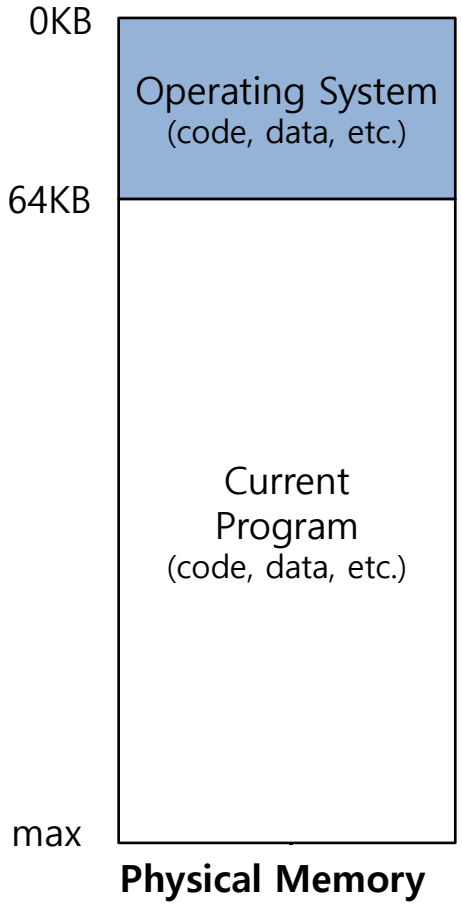
- What is **memory virtualization**?
  - OS virtualizes its physical memory
  - OS provides an illusion memory space per process
  - It seems to be seen like each process uses the whole memory

Benefits of Memory Virtualization

- Ease of use in programming
- Memory efficiency in terms of time and space
- The guarantee of isolation for processes as well as OS
  - Protection from errant accesses of other processes

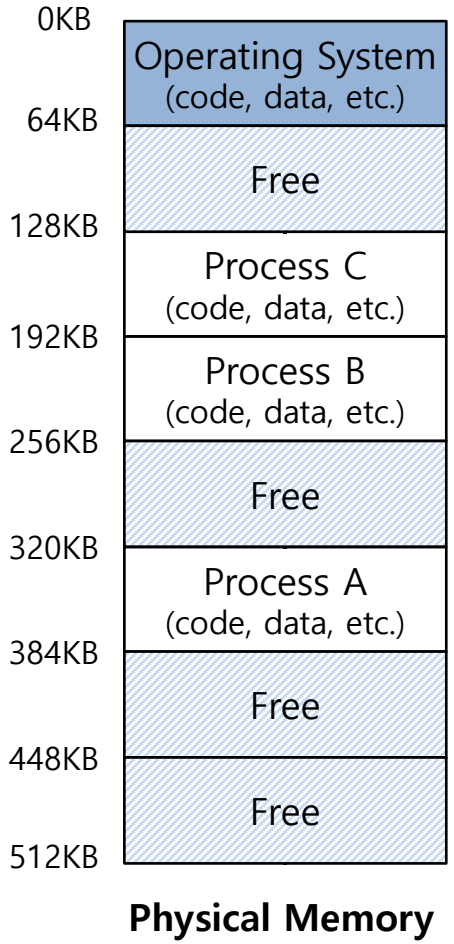
OS in early systems

- OS is just a library
- Load only one process in memory
  - Executes until completion
  - Poor CPU utilization and memory efficiency
    - CPU is idle when process is doing I/O
    - Other processes can fit in memory



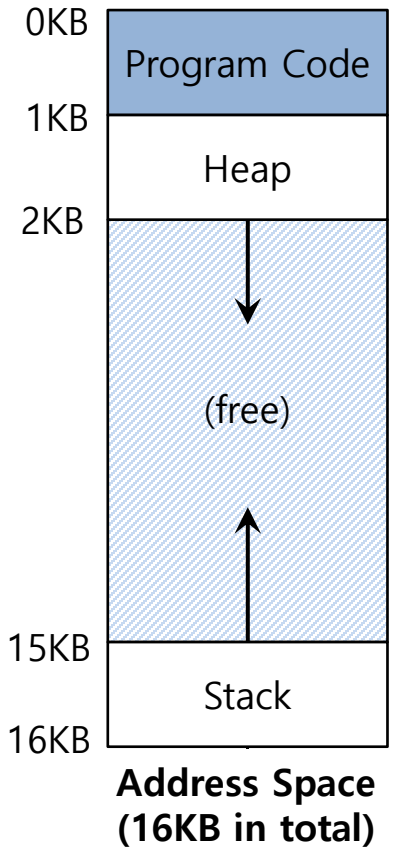
Multiprogramming and Time Sharing

- **Load multiple processes** in memory
  - Execute one for a short while
  - Switch between processes in memory
  - Result: Increases CPU utilization and memory efficiency
- Systems became more interactive compared to doing just batch-processing
- Introduces an important **protection issue**
  - What if a user process accesses another user process, or the kernel??!!



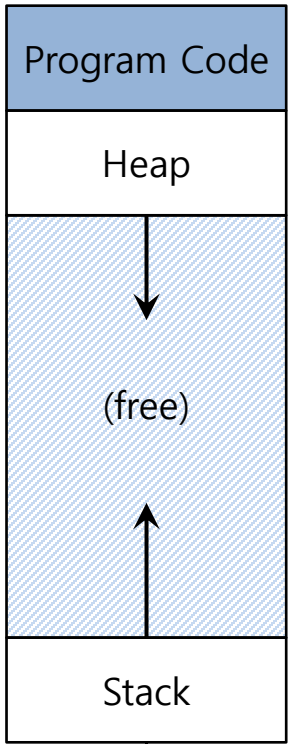
Abstraction: Address Space

- OS creates an **abstraction** of physical memory
  - The address space/mapping contains information all about a running process
  - A running program's(process) own view of memory, its memory state
    - Important: may not be actual locations in physical memory!!
  - Typically consists of program code, heap, stack, etc.



Abstraction: Address Space(Cont.)

- Code
  - Where instructions live
- Heap
  - Dynamically allocate memory
    - `malloc()` in C language
    - `new` in object-oriented language
- Stack
  - Stores return addresses during function calls or return values
  - Contains local variables, actual parameters to functions



Address Space

The Virtual Address

- **Every address** in a running program(a process) is virtual!!
  - OS translates the virtual address to physical address

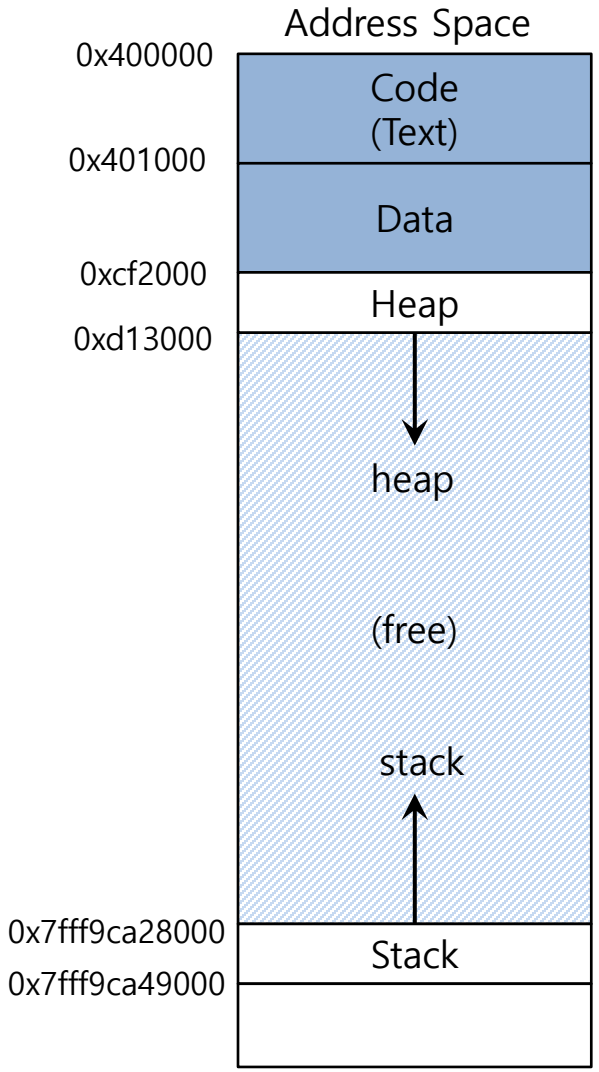
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){

    printf("location of code  : %p\n", (void *) main);
    printf("location of heap  : %p\n", (void *) malloc(1));
    int x = 3;
    printf("location of stack : %p\n", (void *) &x);

    return x;
}
```

A simple program that prints out virtual addresses

Virtual Address(Cont.)

- The output in 64-bit Linux machine

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



Goals for a Virtual Memory system

1. **Transparency**
   - implementation should be invisible to a process
   - Process behaves as if it has its own private physical memory
2. **Efficiency**
   - In terms of time(not make processes run slowly) and space(not much overhead in memory use)
   - Will need to rely on hardware support (TLBs, etc)
3. **Protection and Isolation**
   - Prevent a user process from accessing other user processes, or the kernel