Memory Virtualization: Address Spaces

Prof. Patrick G. Bridges

Memory Virtualization

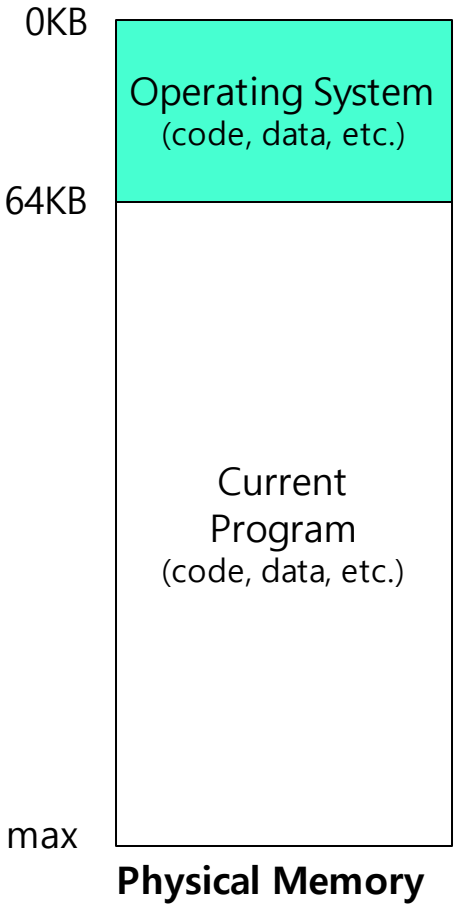
- **What is memory virtualization**?
  - OS virtualizes its physical memory.
  - OS provides an illusion memory space per each process.
  - It seems to be seen like each process uses the whole memory .

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

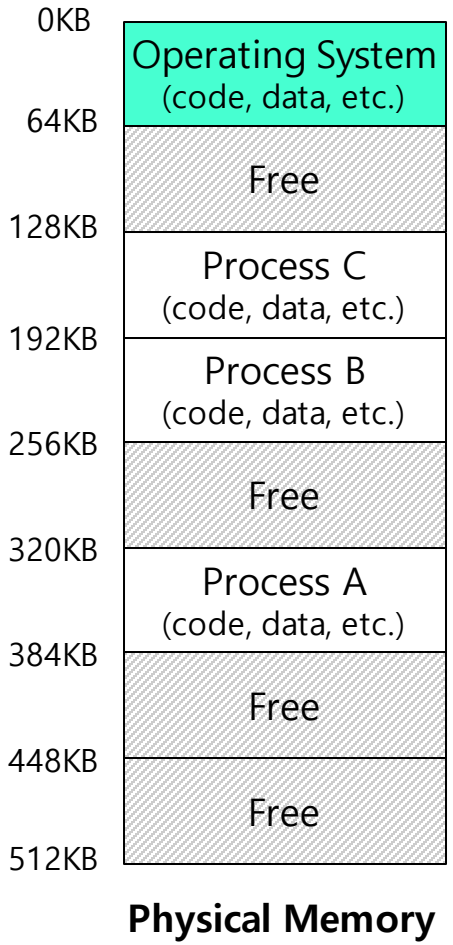
OS in early eystems

- **Load only one process in memory.**
  - Poor utilization and efficiency



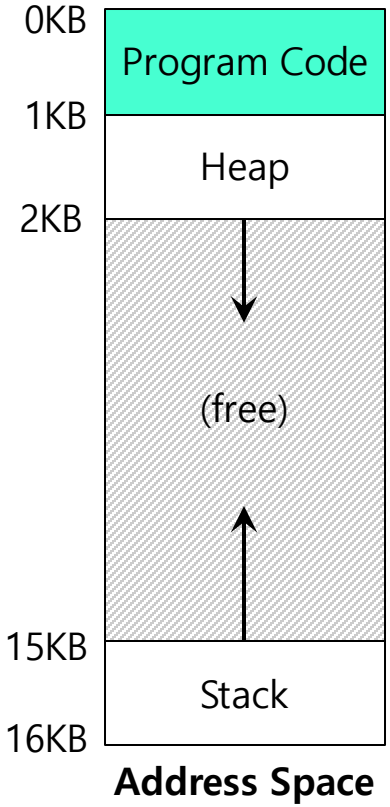
Multiprogramming and Time Sharing

- **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



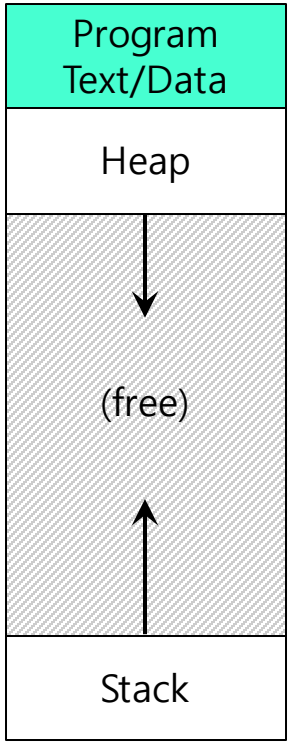
Address Space

- **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.



Address Space(Cont.)

- **Text/Data**
  - Where instructions and global variables live
- **Heap**
  - Dynamically allocate memory.
    - `malloc` in C language
    - `new` in object-oriented language
- **Stack**
  - Store return addresses or values.
  - Contain local variables arguments to routines.



Address Space

Virtual Address

- **Every address in a running program 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 addresses

Virtual Address(Cont.)

- **The output in 64-bit Linux machine**

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