CS 3502
Operating Systems

Page Design and Segmentation

Kun Suo

Computer Science, Kennesaw State University

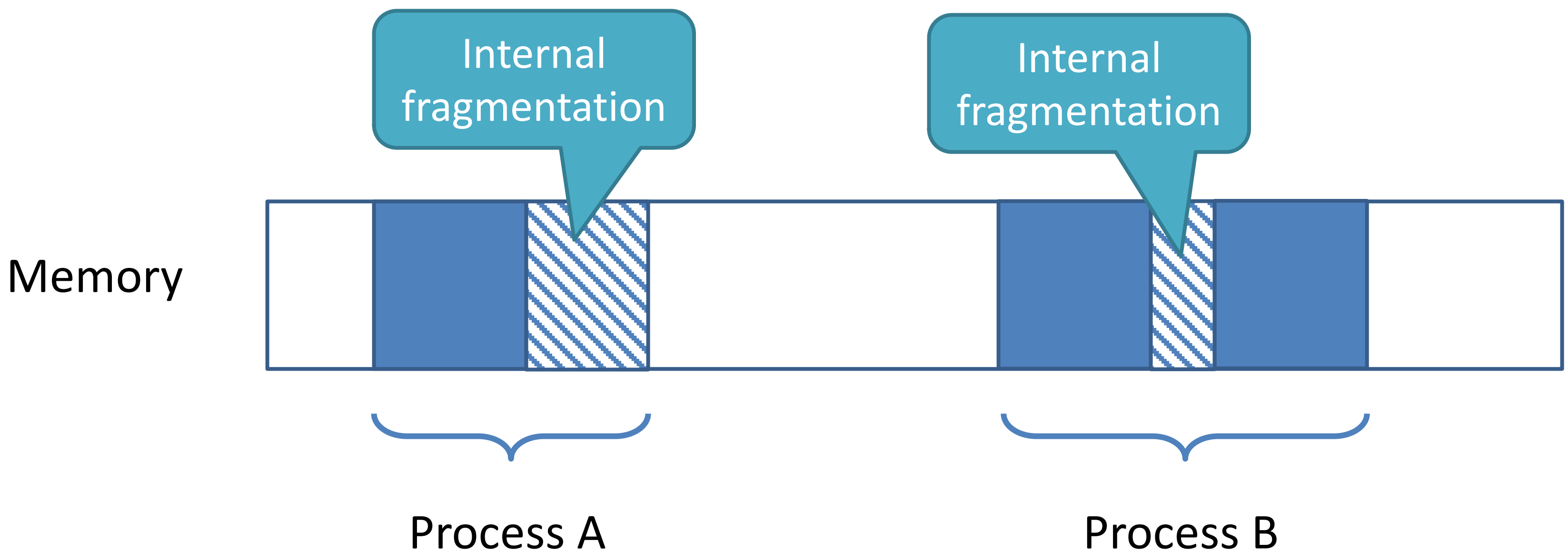
https://kevinsuo.github.io/

Outline

- Page design
  - Internal fragmentation vs. External fragmentation
  - Local page replacement vs. Global page replacement
  - Page size small vs. Large
  - Shared page
  - Paging with Process life cycle
- Segmentation
  - Page vs. Segmentation

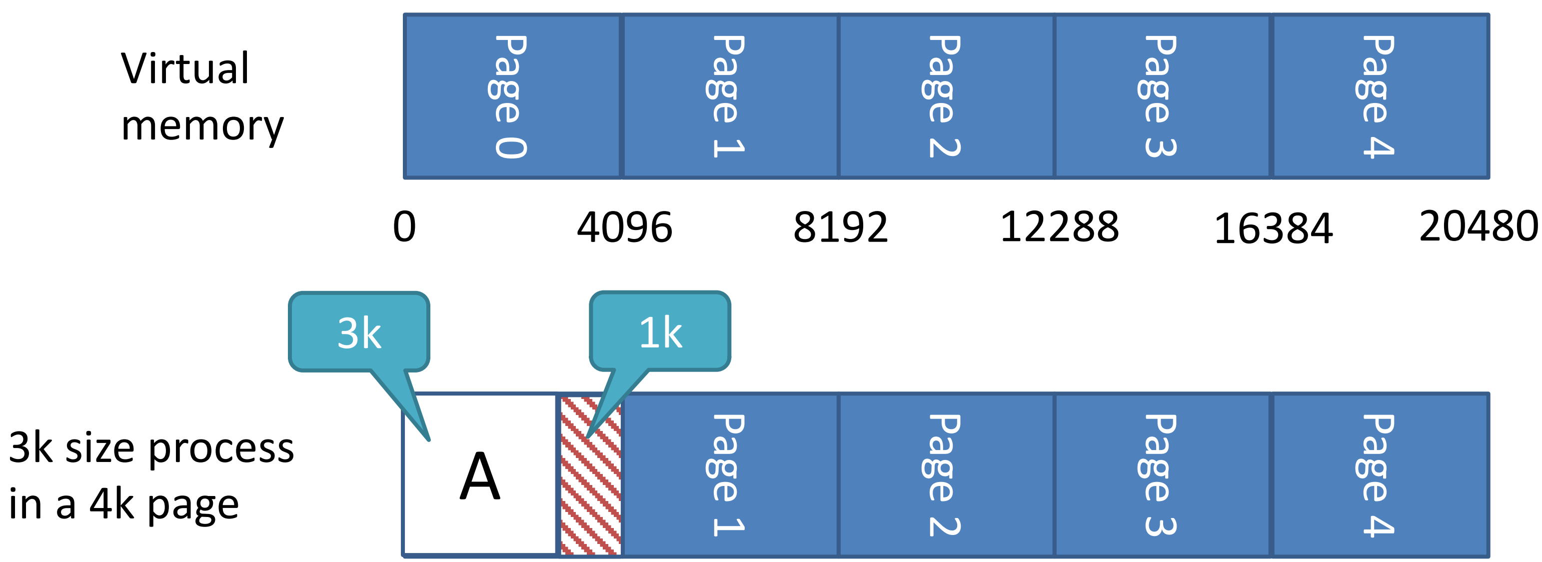
Internal fragmentation

- ***Internal fragmentation***: when memory allocated to a process is larger than requested memory, the difference between these two numbers is internal fragmentation.



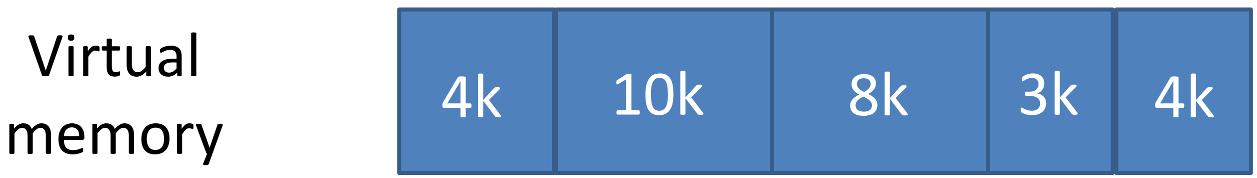
How internal fragmentation is generated?

- Many internal fragmentation is caused by **fixed-sized blocks** of memory
- Whenever a process requests for the memory, the fixed sized block is allocated to the process

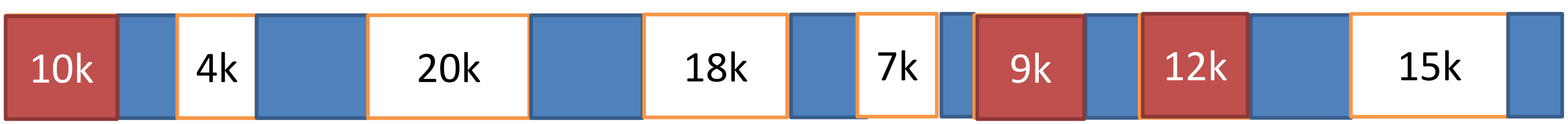


How to deal with the internal fragmentations?

- Allocate dynamic size blocks of memory based on process requirement

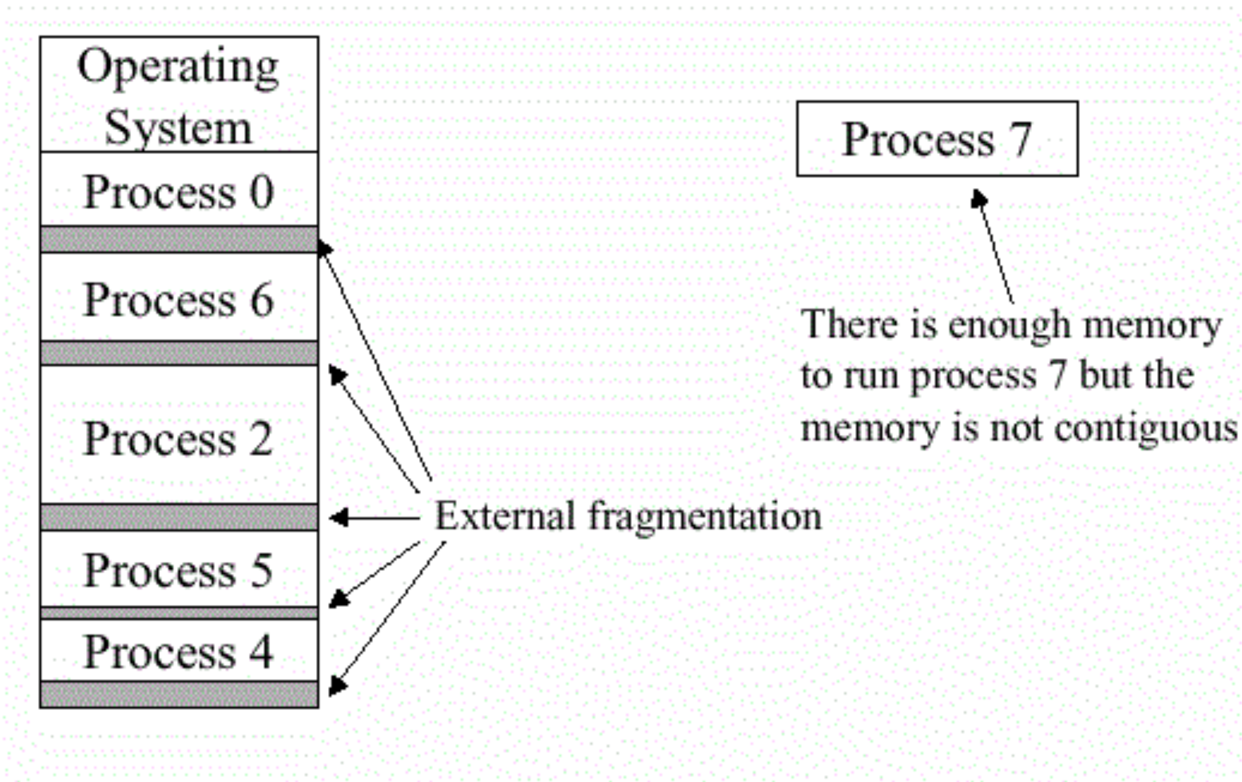


- Best-fit allocation algorithm



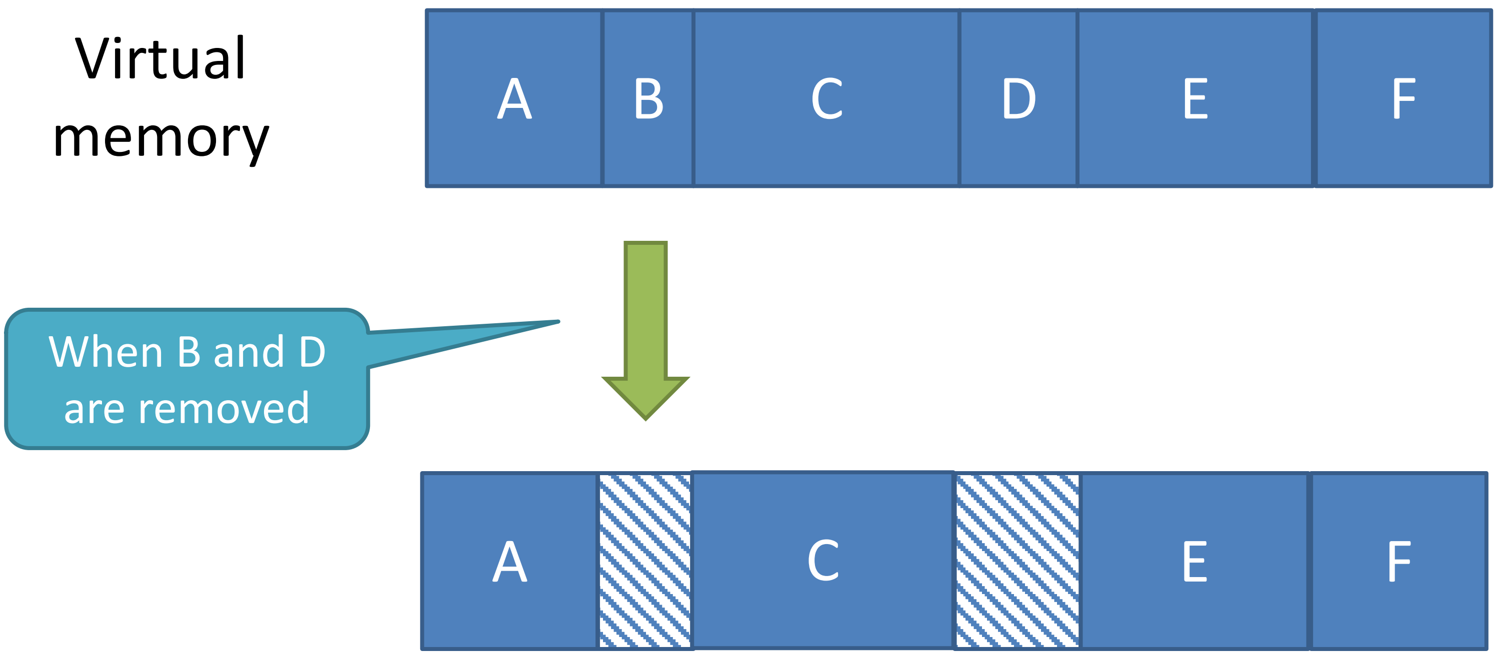
External fragmentation

- ***External fragmentation***: Total memory space is enough to satisfy a request or to reside a process in it. However, it is not contiguous and can not be used.



How external fragmentation is generated?

- When a process is removed from the memory, the free space creates the hole in the memory



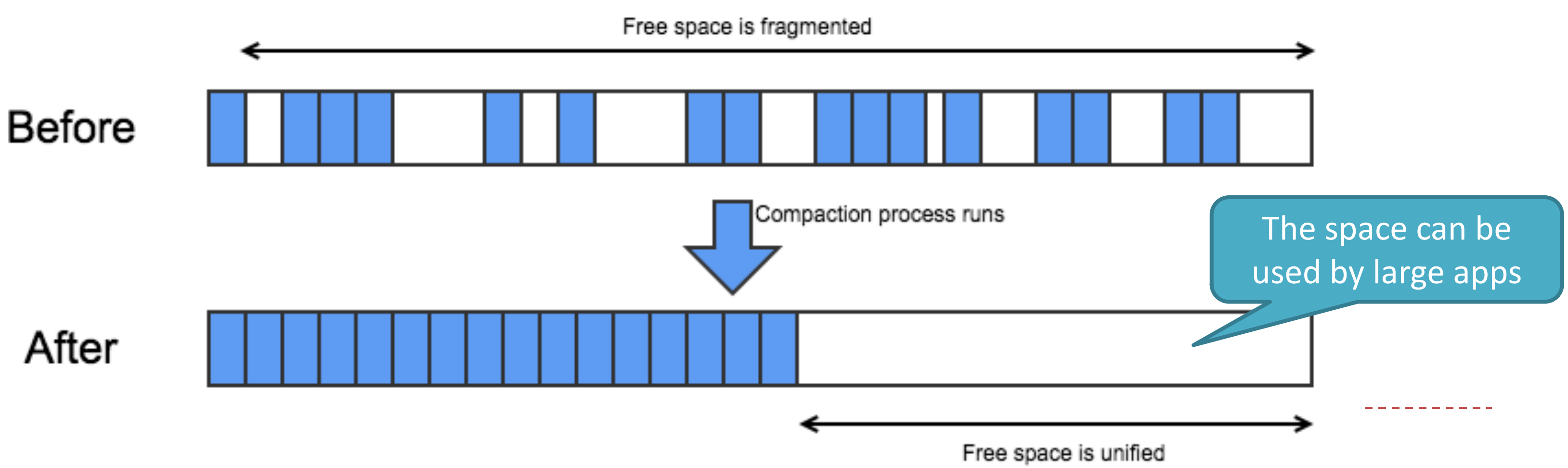
How to deal with the external fragmentations?

- Best-fit allocation algorithm

- Memory compaction

Memory compaction in MacOS

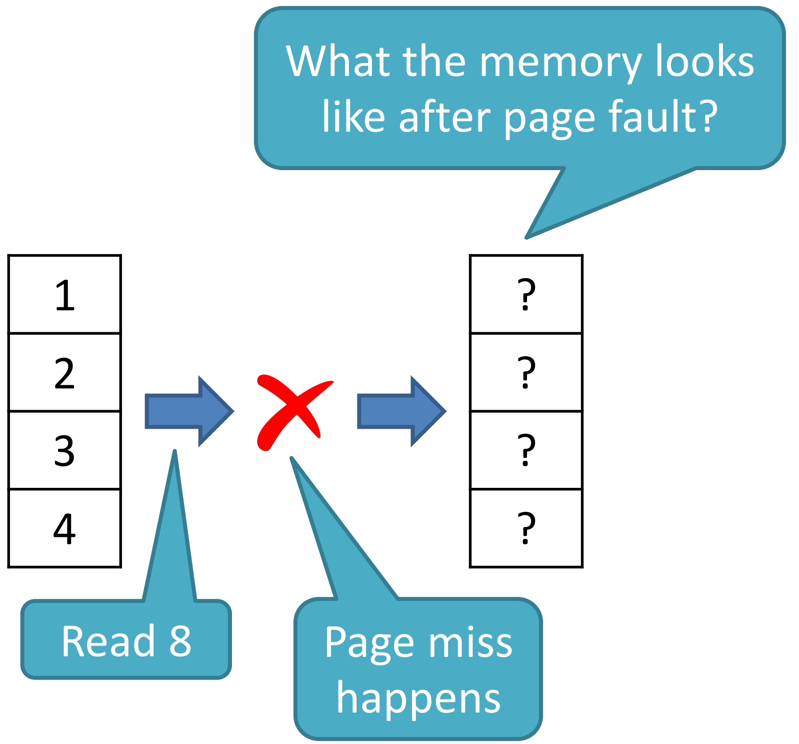
https://youtu.be/hIigp_bxUcQ?t=1763



Internal fragmentation vs. External fragmentation

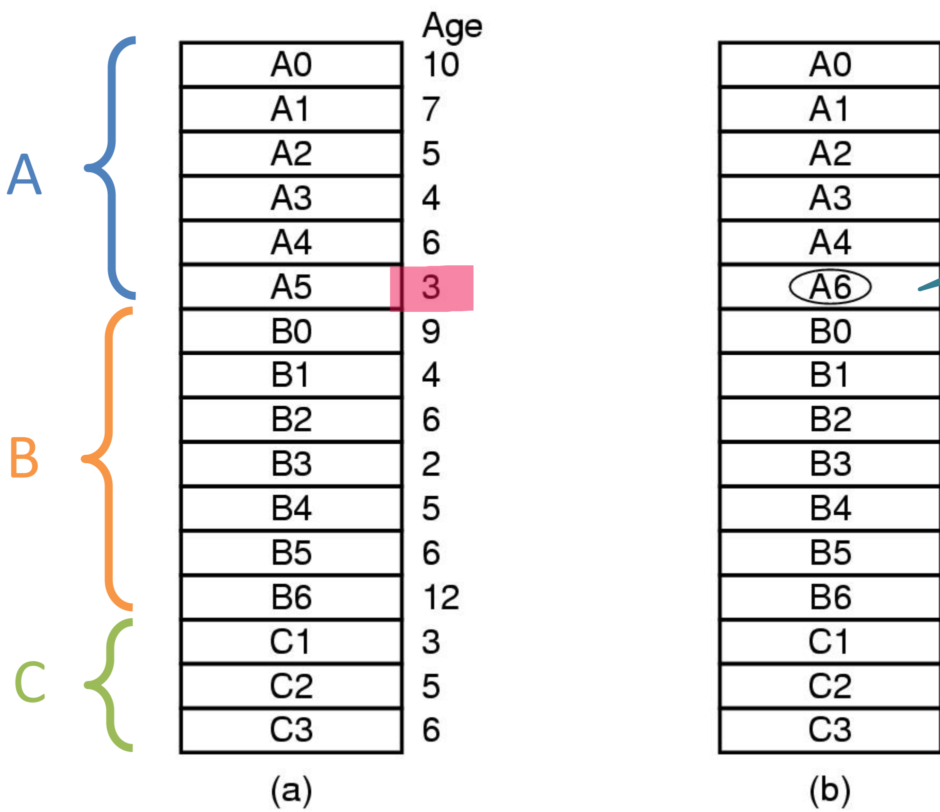
	Internal fragmentation	External fragmentation
Definition	A form of fragmentation that arises when there are sections of memory remaining because of allocating large blocks of memory for a process than required	A form of fragmentation that arises when there is enough memory available to allocate for the process, but that available memory is not contiguous
Reason	Memory block assigned to a process is large – the remaining portion is left unused as it cannot be assigned to another process	Memory space is enough to reside a process, but it is not contiguous. Therefore, that space cannot be used for allocation
Solution	Best fit Dynamic block size	Best fit Memory Compaction

Page replacement



- Page replacement algorithm
  - OPR
  - FIFO
  - LRU
  - NFU
  - NRU
  - Second chance
  - Clock
  - Aging

Local Page replacement



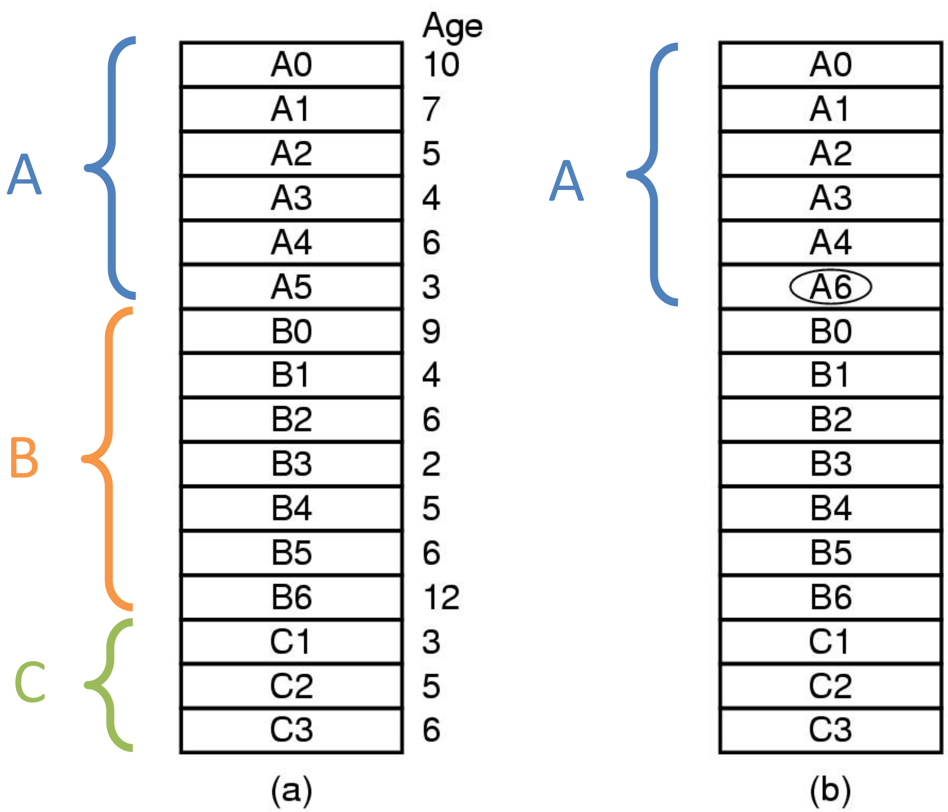
A5 will be replaced by local algorithm

Suppose the algorithm replaces the page which has *the least age*.

e.g., A6 comes

(a) Original configuration. (b) Local page replacement.

Local Page replacement



- Local page replacement requires static allocation
- The page number of one process does not change during replacement

(a) Original configuration. (b) Local page replacement.

Local Page replacement problem

- Thrashing
  - Physical memory is too small to hold the process work set
  - Large page faults happen and swap frequently
  - Slowdown the process speed

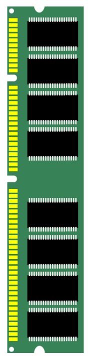
		Age
A	A0	10
	A1	7
	A2	5
	A3	4
	A4	6
	A5	3
B	B0	9
	B1	4
	B2	6
	B3	2
	B4	5
	B5	6
	B6	12
C	C1	3
	C2	5
	C3	6

Suppose A needs 9 pages during execution:

A0 A1 A2 A3 A4 A5 *A6 A7 A8 …*

Thrashing happens

Local Page replacement problem: Thrashing

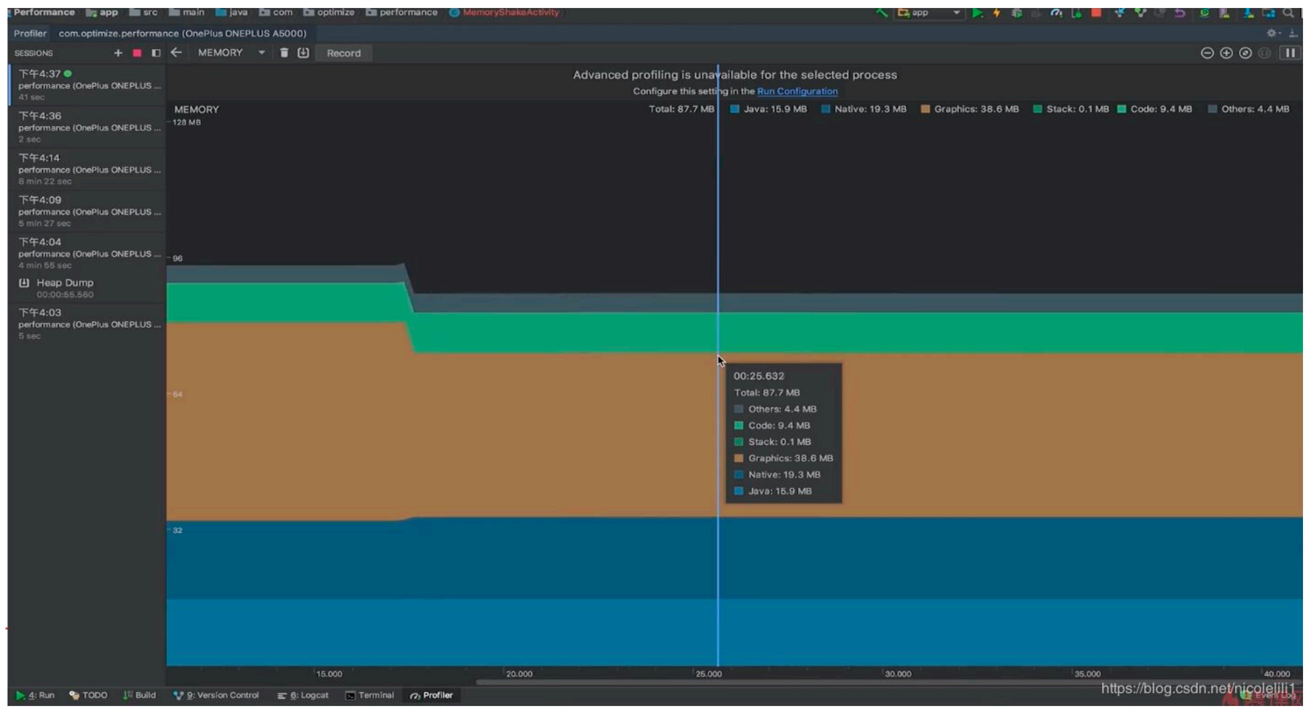


A0
A1
A2
A3
A4

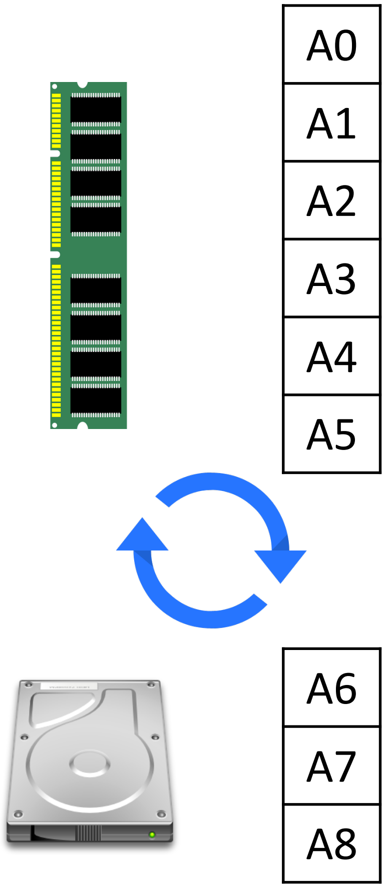
Suppose A needs ≤ 6 pages during execution

and the memory space for A is 6

Memory performance is stable

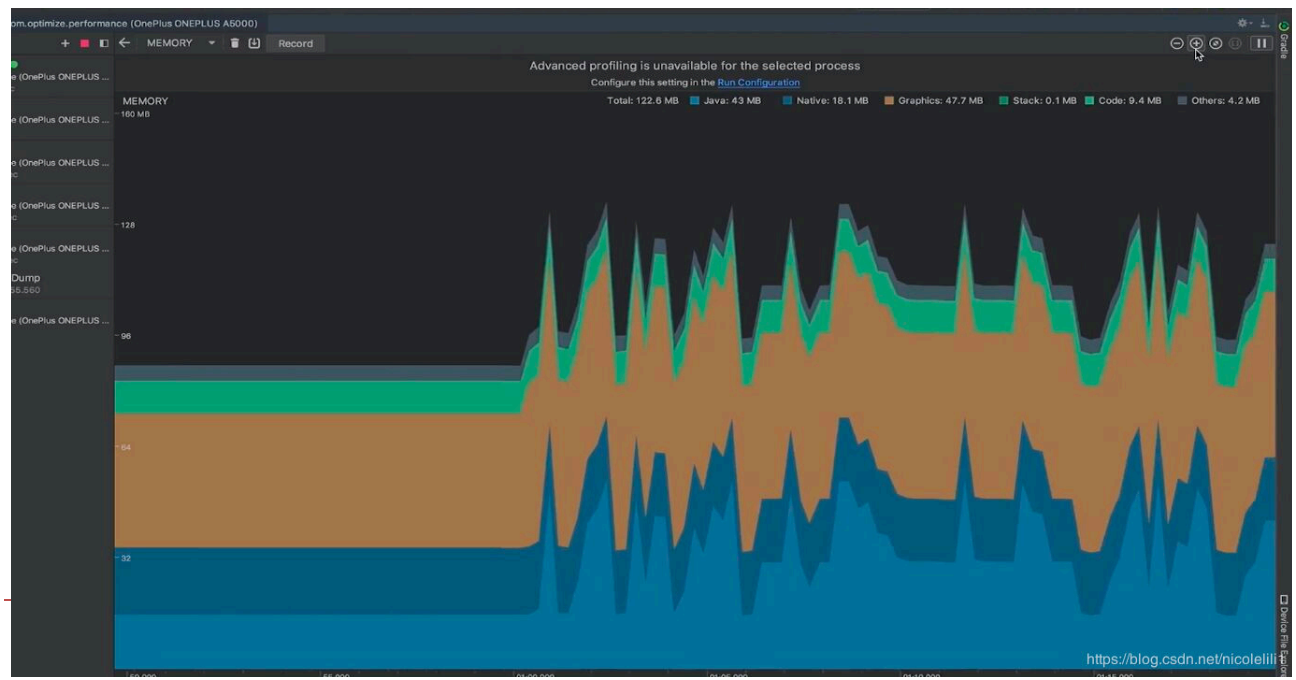


Local Page replacement problem: Thrashing



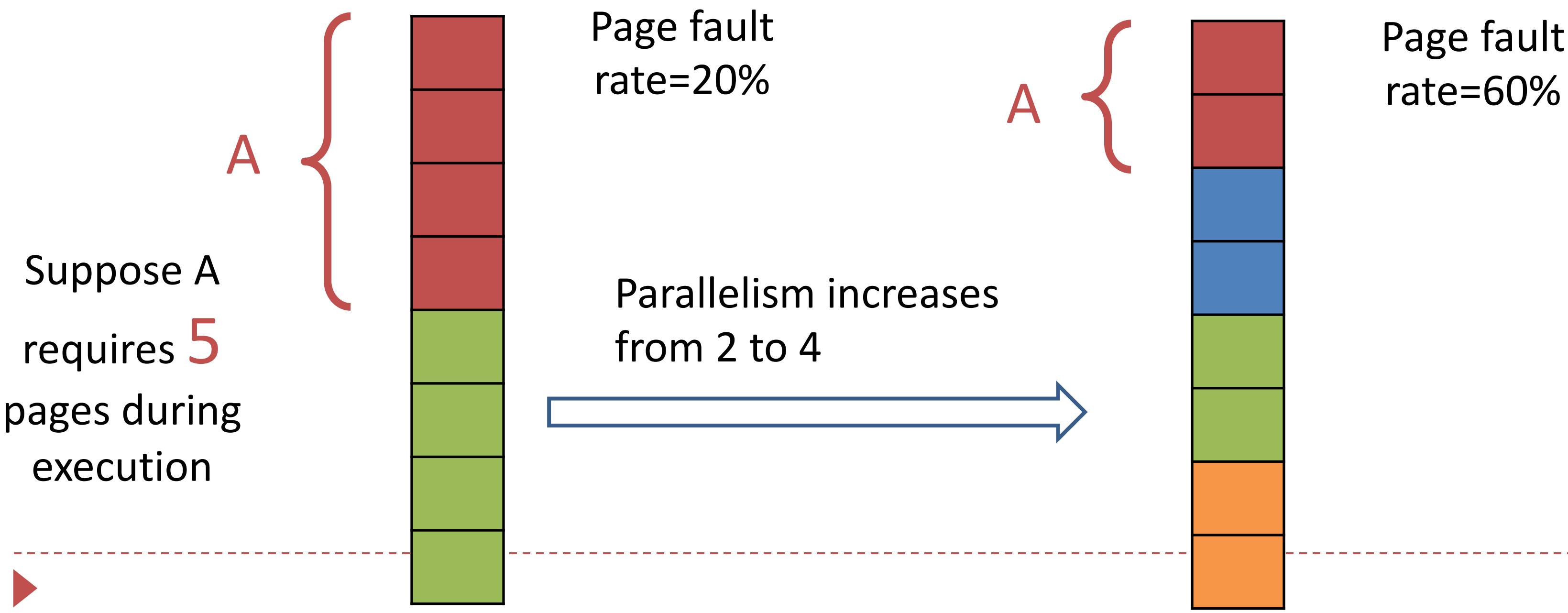
Suppose A needs 9 pages during execution and the memory space for A is only 6

Memory performance is unstable and lots of CPU resources will be wasted on swapping

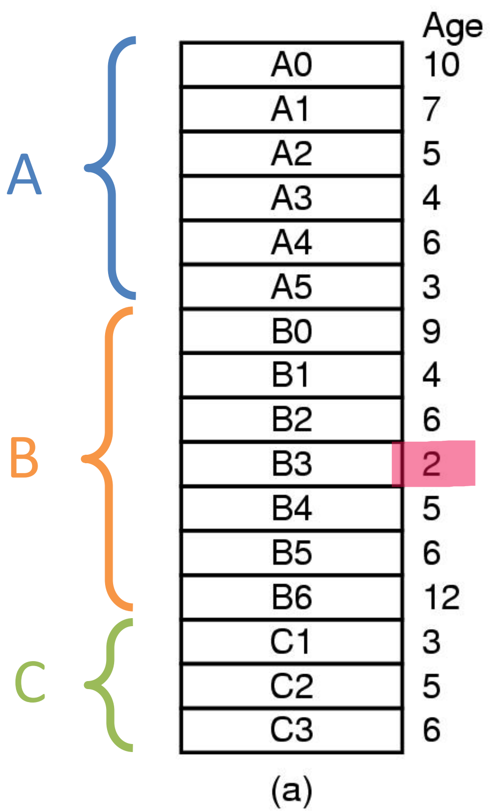


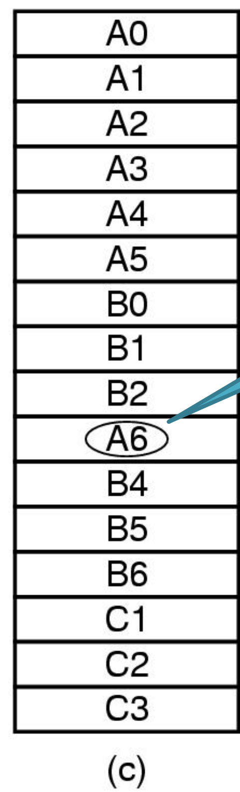
Local Page replacement problem

- Thrashing
  - As the number of process in memory increases, the memory for each process decreases and page faults could also increase
  - OS needs a ***tradeoff*** between parallelism and page fault rate



Global Page Replacement





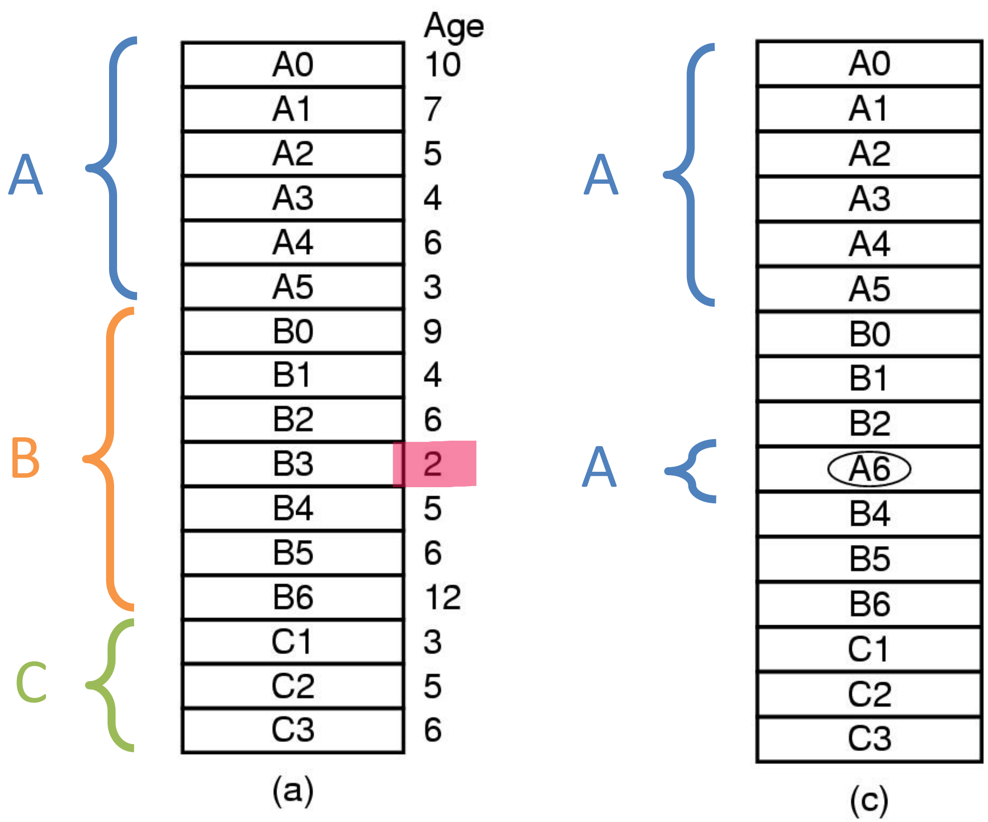
B3 will be replaced by global algorithm

Suppose the algorithm replaces the page which has the least age.

e.g., A6 comes

(a) Original configuration. (c) Global page replacement.

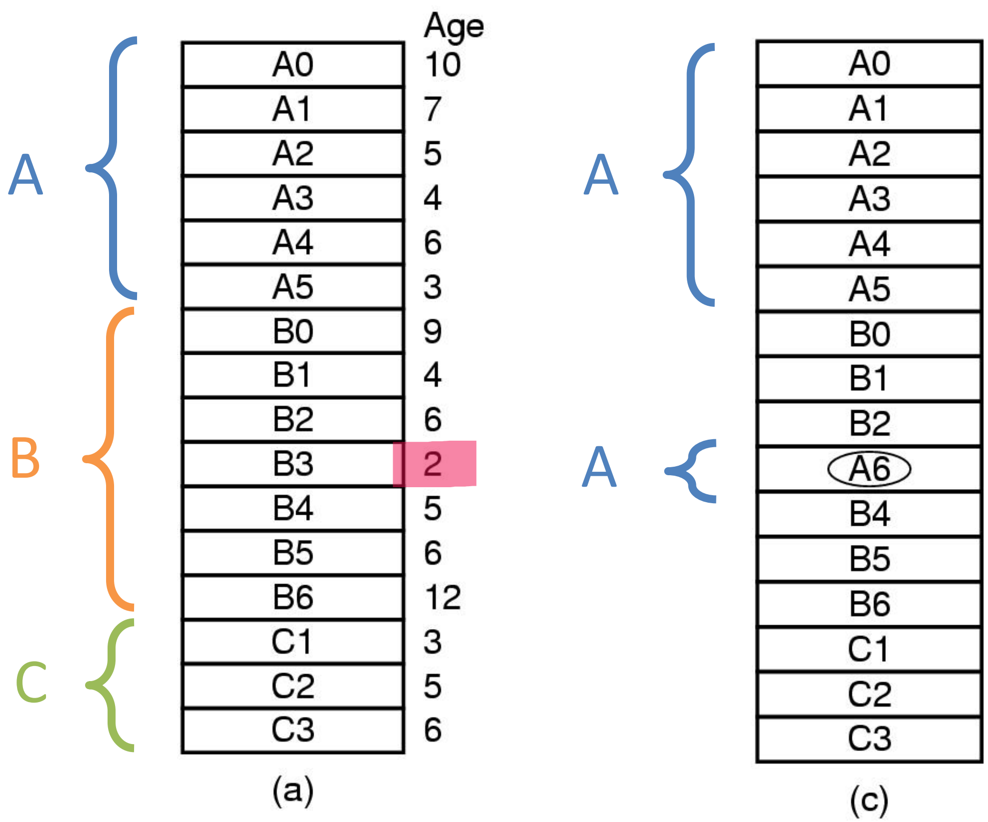
Global Page Replacement



- Global page replacement requires dynamic allocation
- The page number of one process will change during replacement (**A++** , **B--**)

(a) Original configuration. (c) Global page replacement.

Global Page Replacement Problem



- How to control the page frames assigned to each process?
- Otherwise, some processes will take much more memories than others

(a) Original configuration. (c) Global page replacement.

Page Fault Frequency (PFF) algorithm

- PFF: control the size of allocation set of a process
  - when and how much to increase or decrease a process’ page frame allocation

		Age
A	A0	10
	A1	7
	A2	5
	A3	4
	A4	6
	A5	3
B	B0	9
	B1	4
	B2	6
	B3	2
	B4	5
	B5	6
	B6	12
C	C1	3
	C2	5
	C3	6

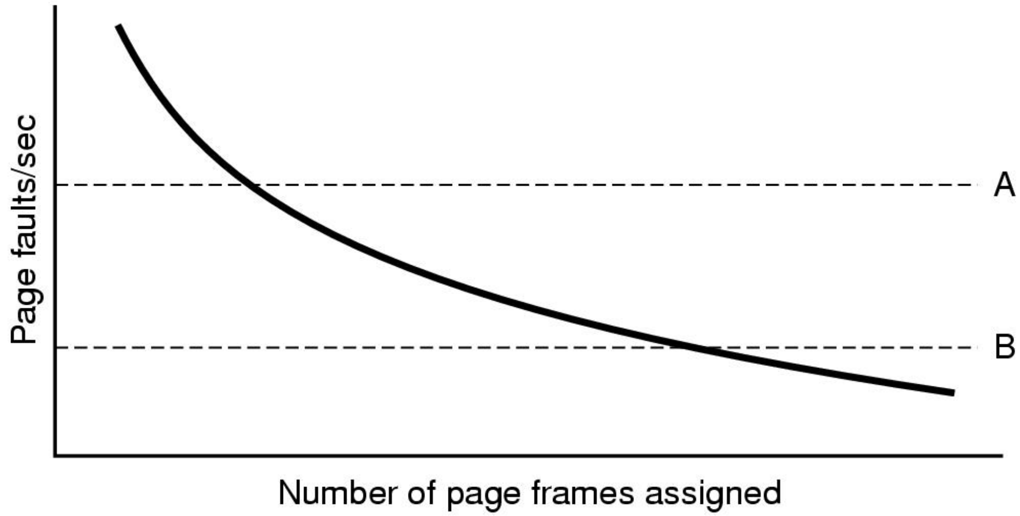
Keep track of page fault rate for each process

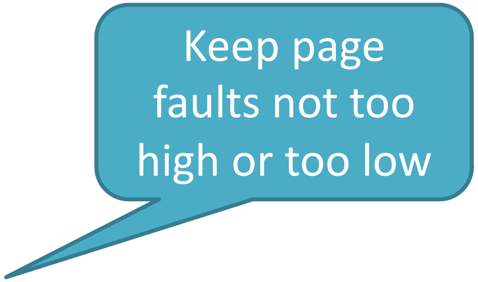
A: 20% → 25% → 30%

B: 20% → 15% → 10%

C: 50% → 50% → 50%

Page Fault Frequency (PFF) algorithm



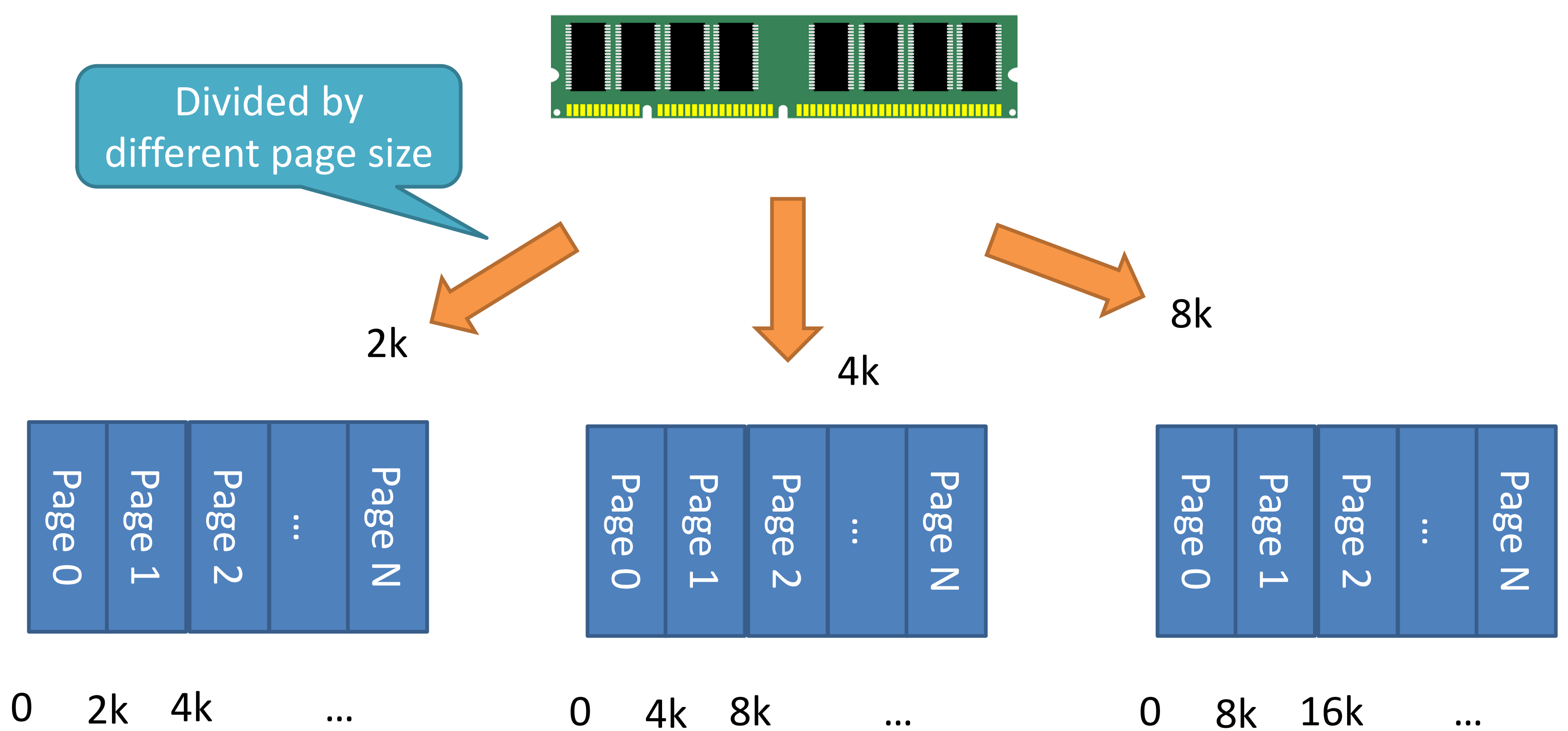


- Keep monitoring the page fault for each process and set the threshold
  - If one process page fault rate is too high, allocate more memory pages for it
  - If one process page fault rate is too low, allocate less memory pages for it

Local Page Replacement vs. Global Page Replacement

	Definition	Allocation	Potential problem	Possible solution
Local Page Replacement	Replace the page within one process	static	Thrashing	OS needs a tradeoff between parallelism and page fault rate
Global Page Replacement	Replace the page entire the system	dynamic	Page allocation imbalance among processes	Page Fault Frequency (PFF) and load control

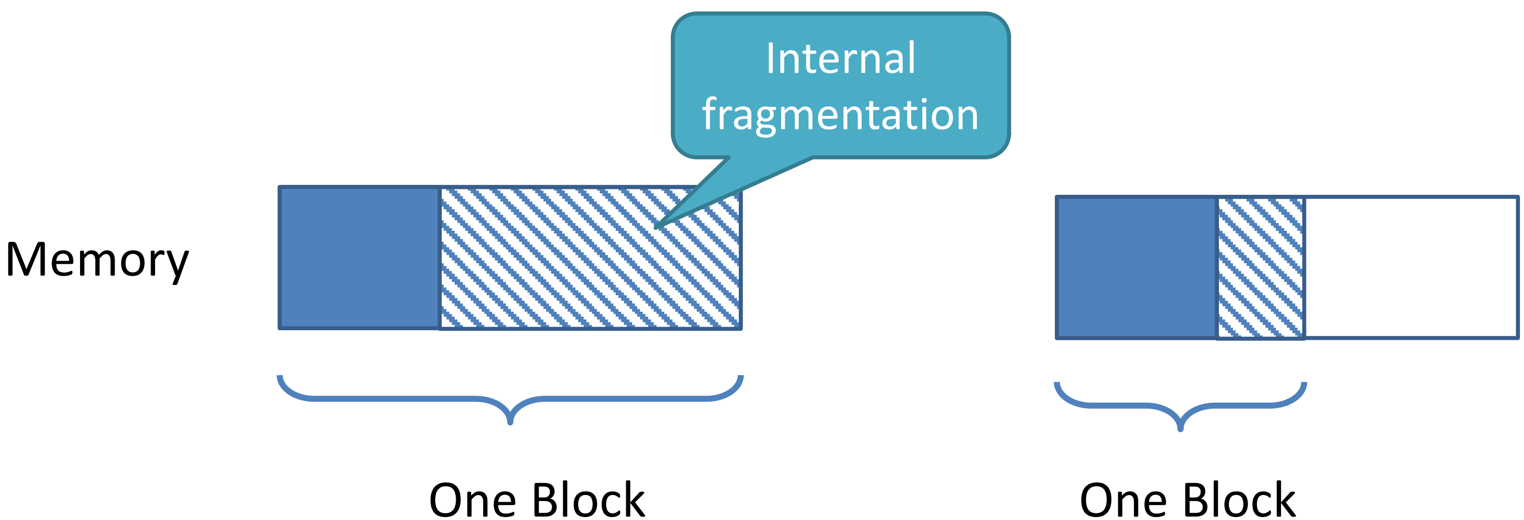
Page Size



Divided by different page size
2k
4k
8k
Page 0
Page 1
Page 2
…
Page N
0
2k
4k
…
Page 0
Page 1
Page 2
…
Page N
0
4k
8k
…
Page 0
Page 1
Page 2
…
Page N
0
8k
16k
…

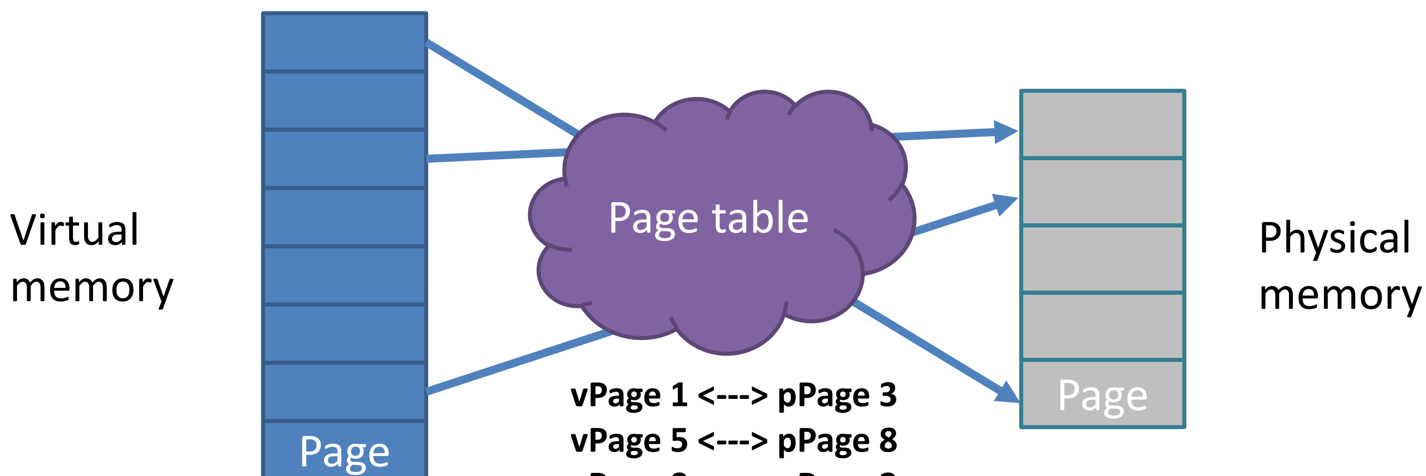
Page Size is small

- Advantages
  - less unused program in memory (due to *internal fragmentation*)
  - better fit for various data structures, code sections (e.g., 80% of the data structures or codes are small)



Page Size is small

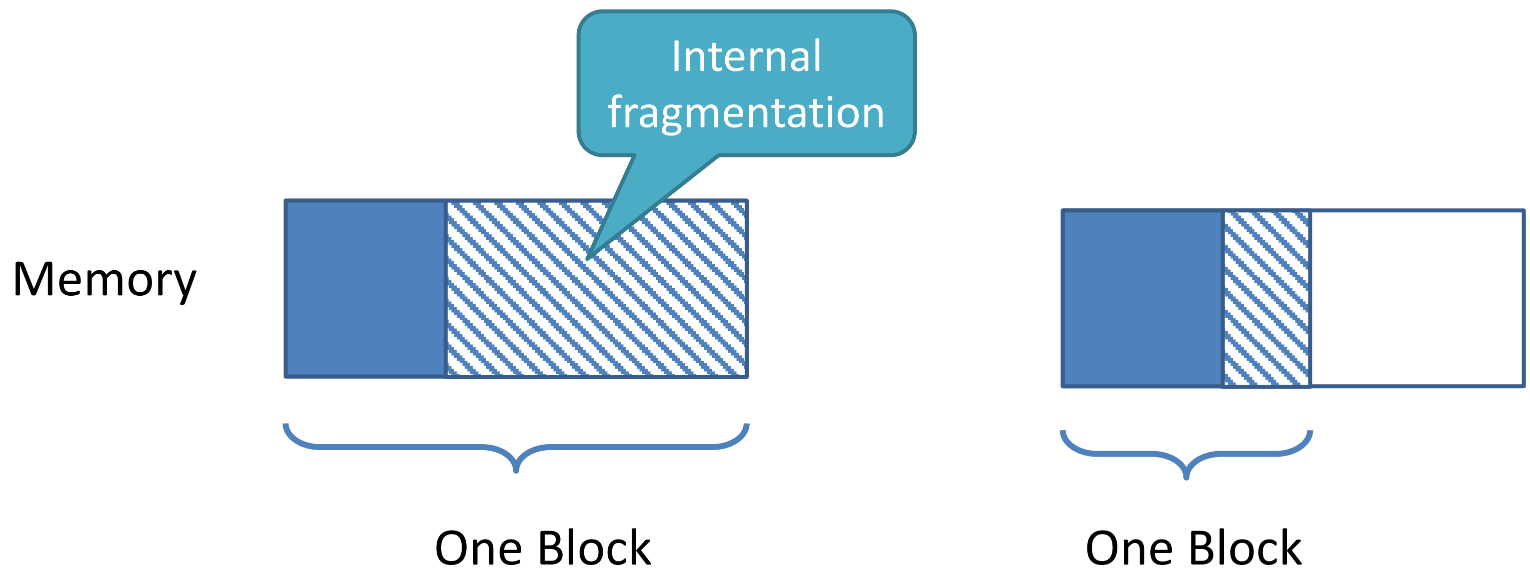
- Disadvantages
  - Programs need many pages, larger page tables
  - Longer access time of page due to more pages
  - More page faults could happen due to more pages



vPage 1 <---> pPage 3
vPage 5 <---> pPage 8
vPage 9 <---> pPage 2
…

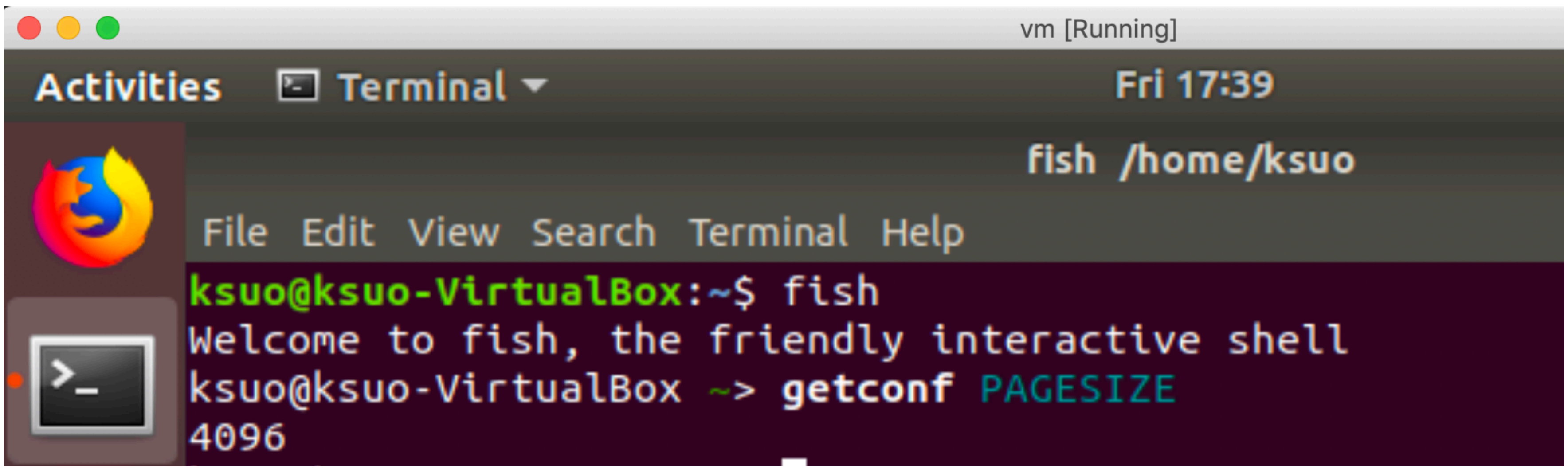
Page Size is large

- Disadvantages
  - More internal fragmentation and less efficiency
- Tradeoff between page size and memory efficiency
  - Normal we choose 4k for page size



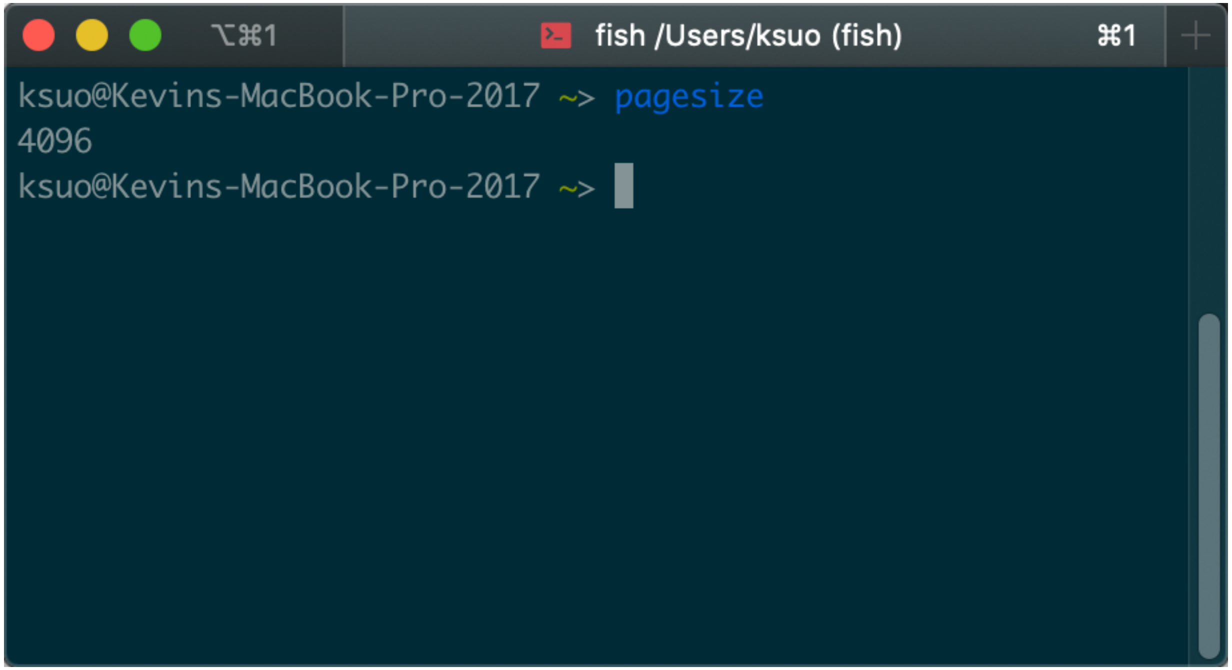
$ getconf PAGESIZE

- How to get page size in Linux



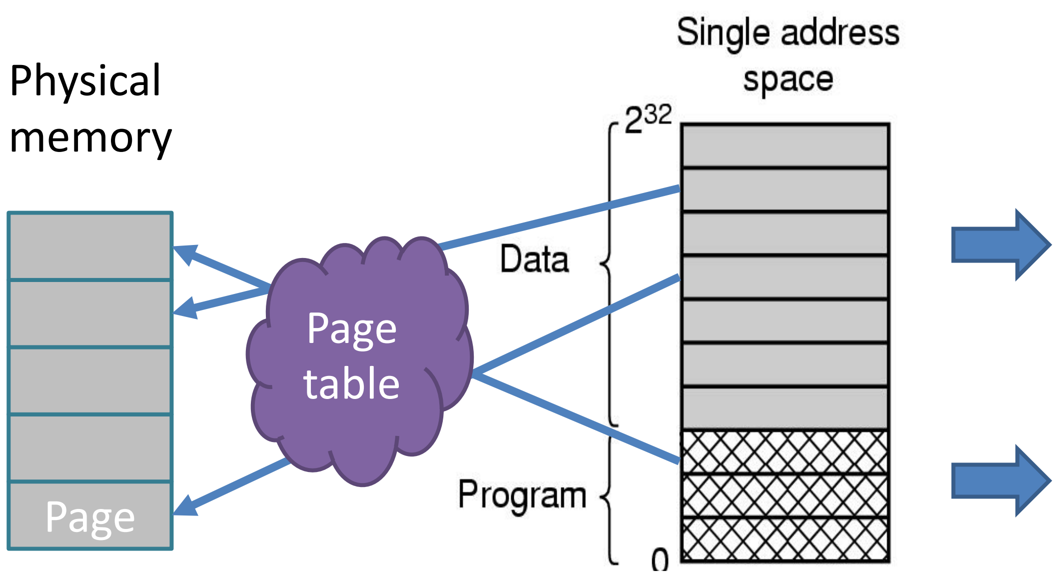
$ pagesize

- How to get page size on Mac



Separate Instruction and Data Spaces

- Normally, the memory address stores instruction and data of program together
  - Address space is limited
  - Interference between program and data memory space (e.g., security issue)



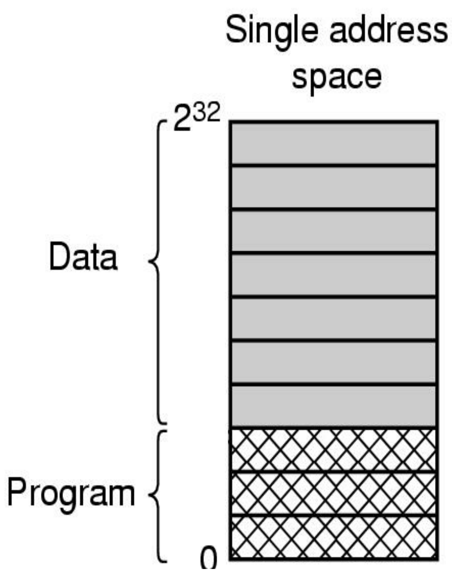
Data memory could increase: e.g., load more files, open high resolution figure/video, etc.

Program memory could increase: e.g., browser open more tabs, increase more threads, etc.

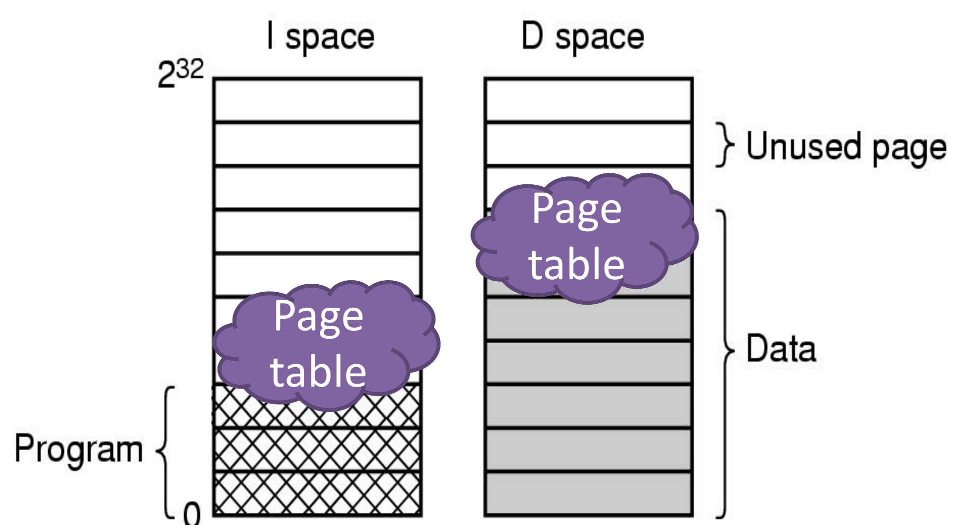
One address space

Separate Instruction and Data Spaces

- Separate memory address: I-space, D-space
  - Address space of instruction and data is independent
  - Both addresses have its own pages and page tables mapping for physical address to virtual address (sacrifice space for performance)



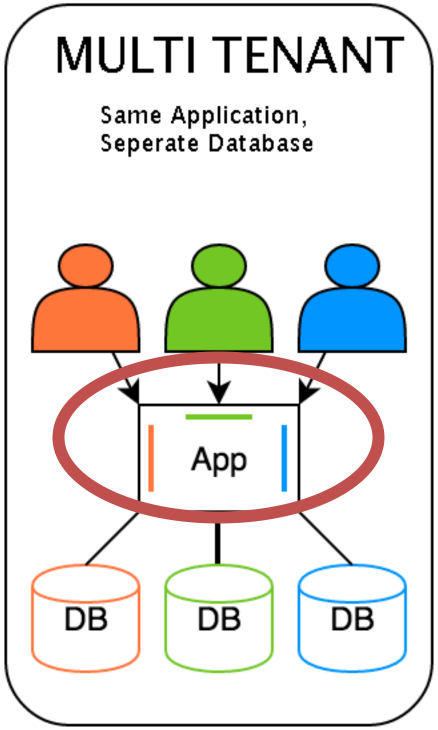
One address space

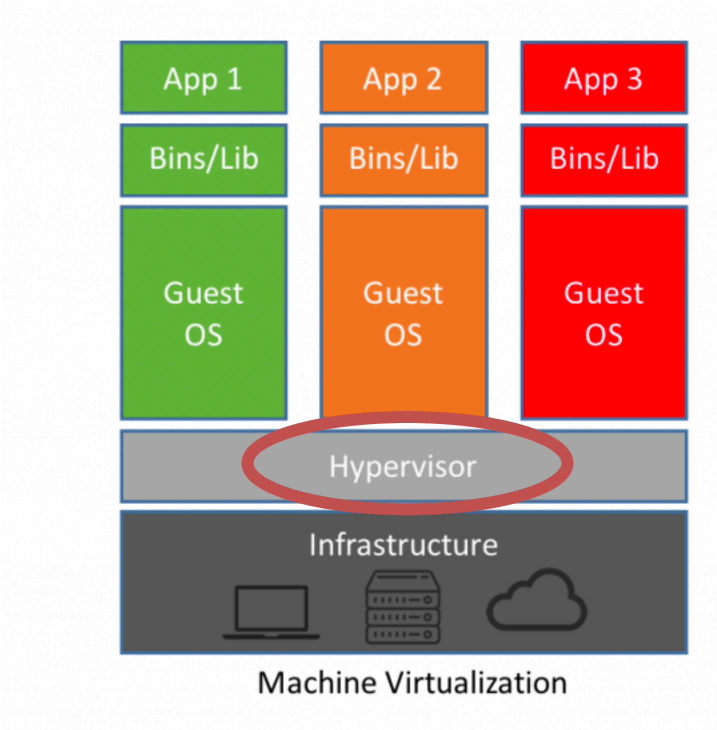


Separate I and D spaces

Shared Pages

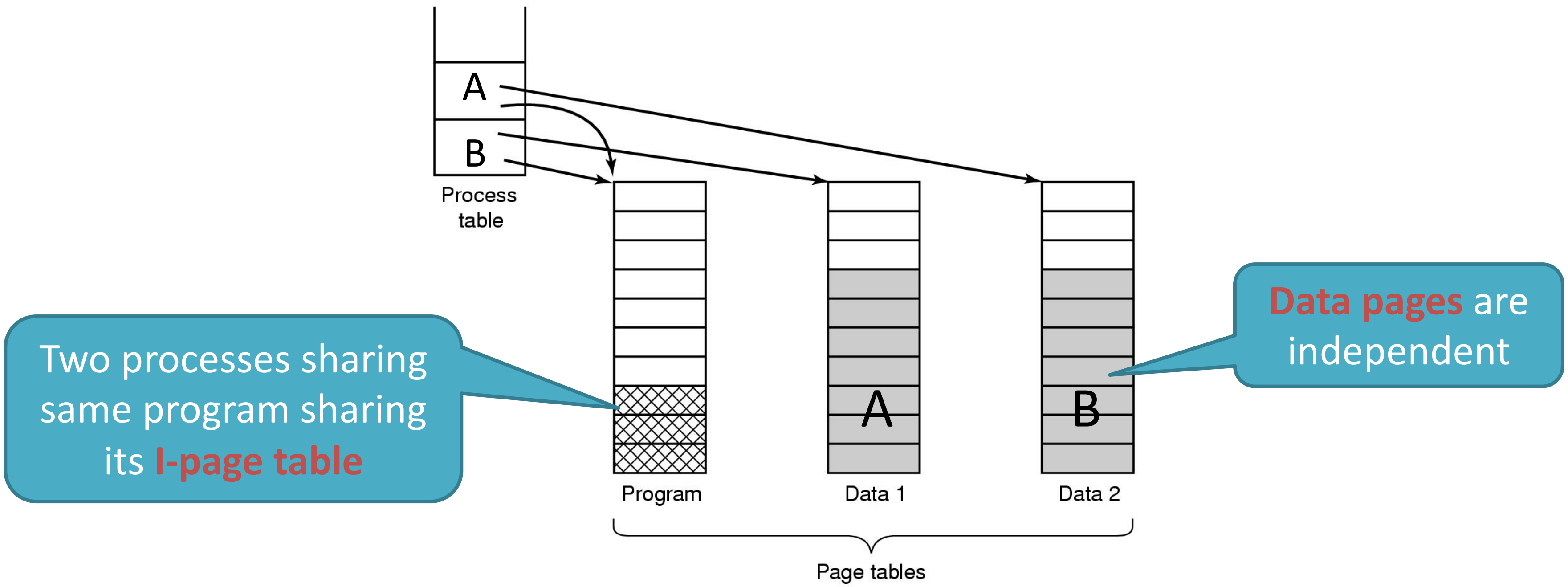
- It is common that multiple users execute a same application
- In the cloud, multiple Oses usually run on the same hypervisor





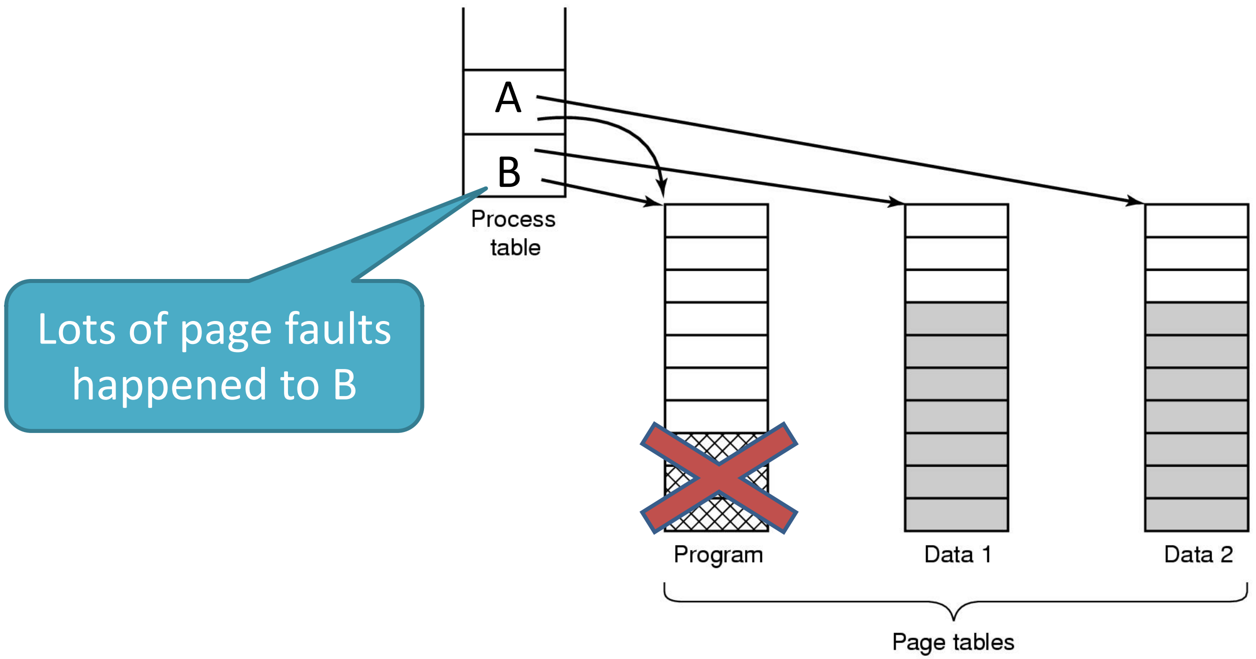
Shared Pages

- To avoid multiple duplicates in memory, shared pages have more efficiency in memory design



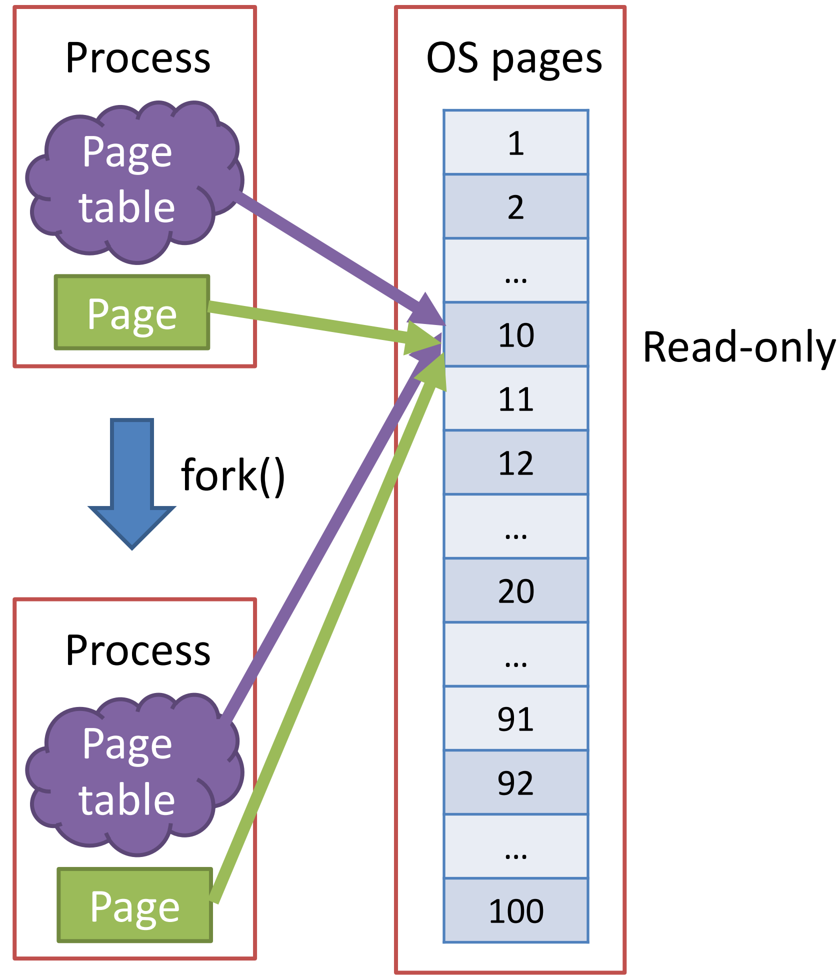
Shared Pages Problems

- Suppose Process A and B share the same I-Page. Then OS scheduler schedules out A and releases all pages of A. What will happen to B?

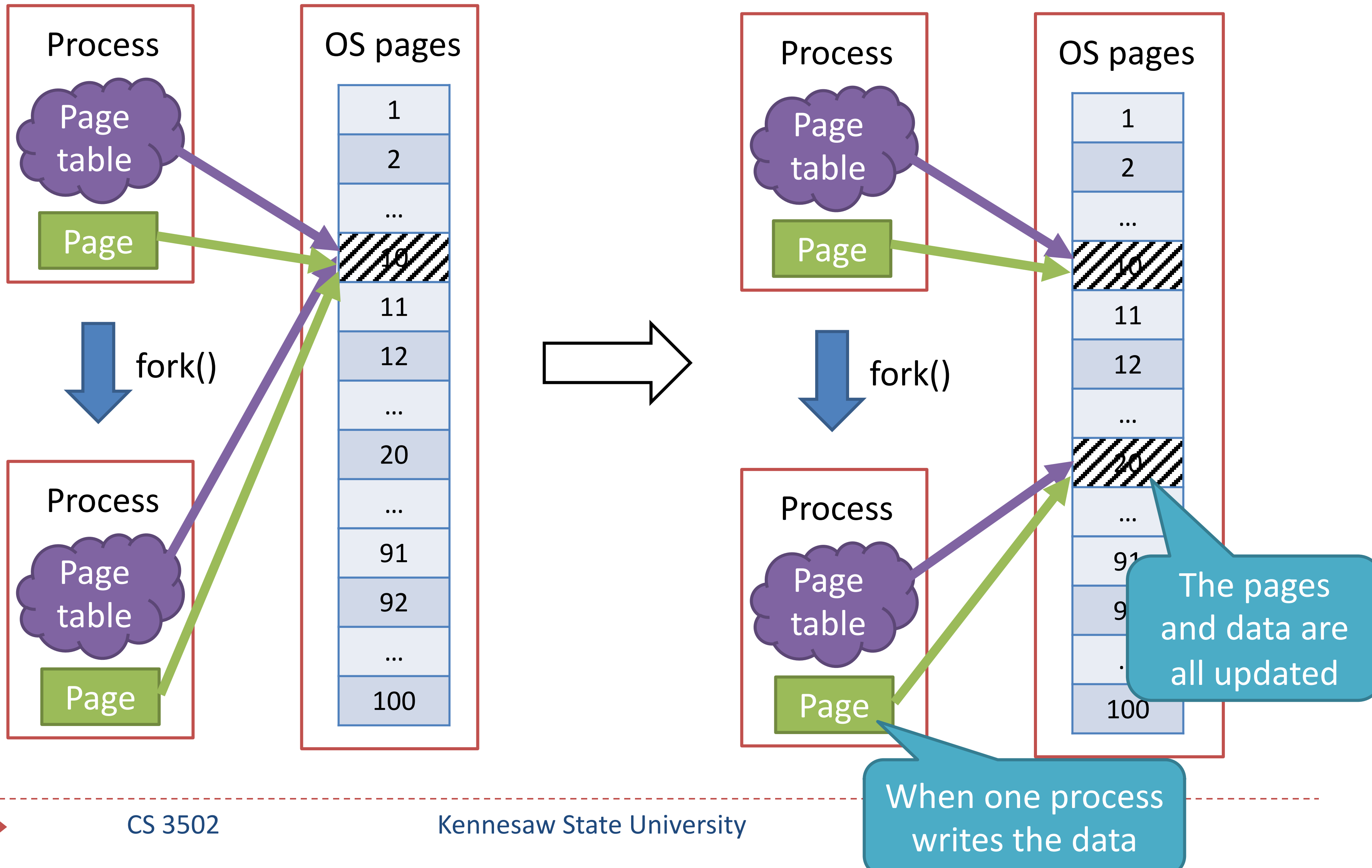


Shared Pages Example: fork()

- In folk(), two processes share program instruction and data in memory
- Two processes have their own page table but mapping to the same page
- All data pages are read-only

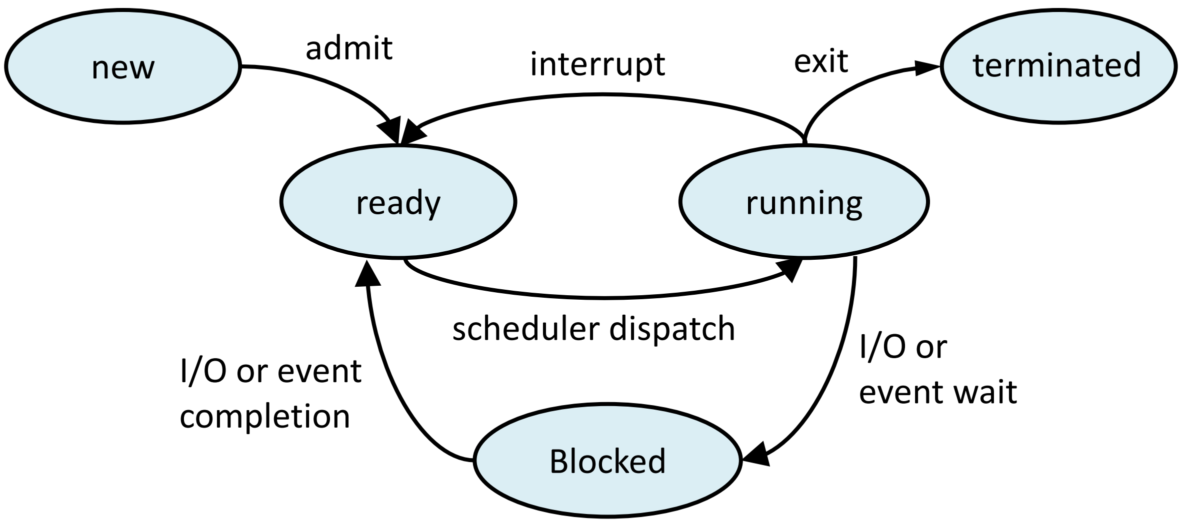


Copy-on-write(COW) for Shared Pages

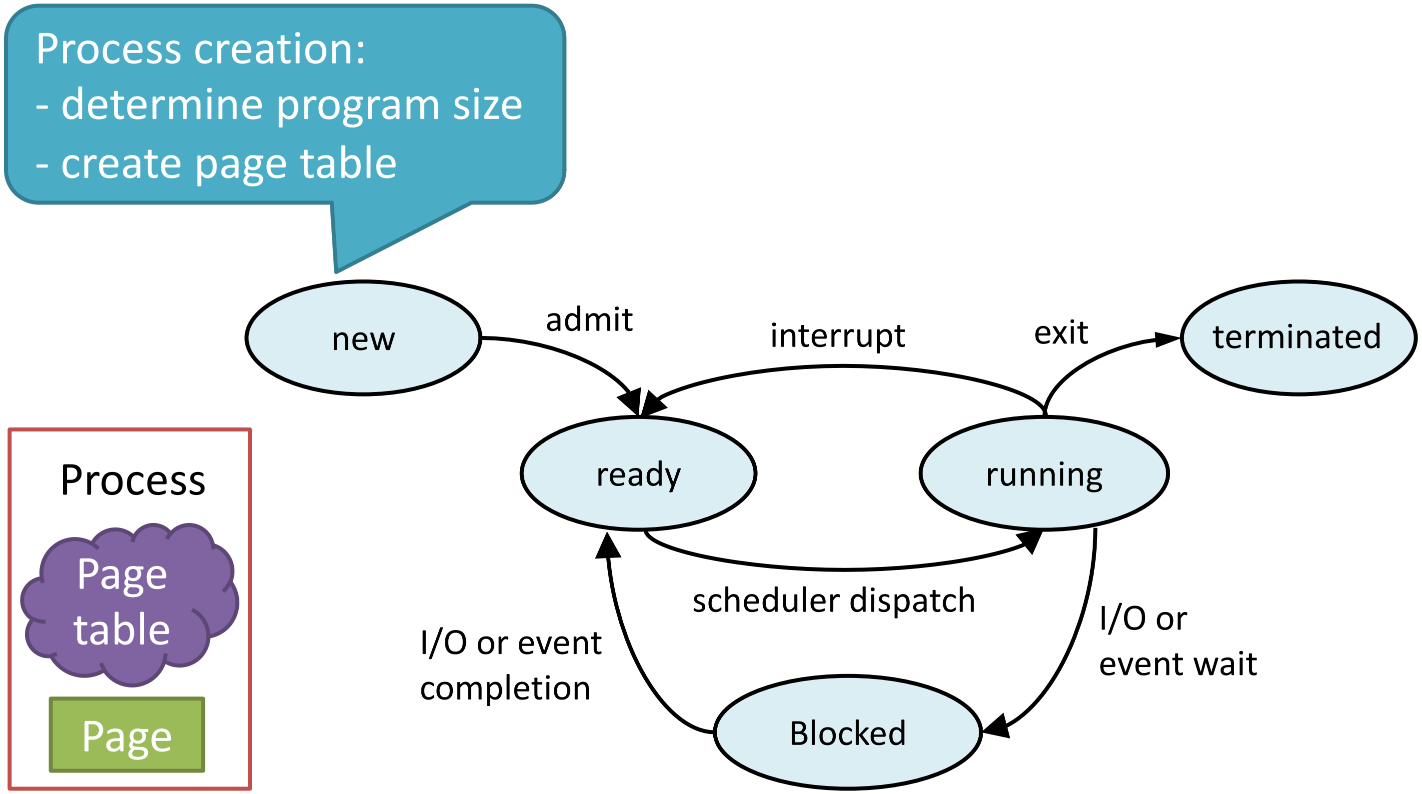


Paging with Process Life Cycle

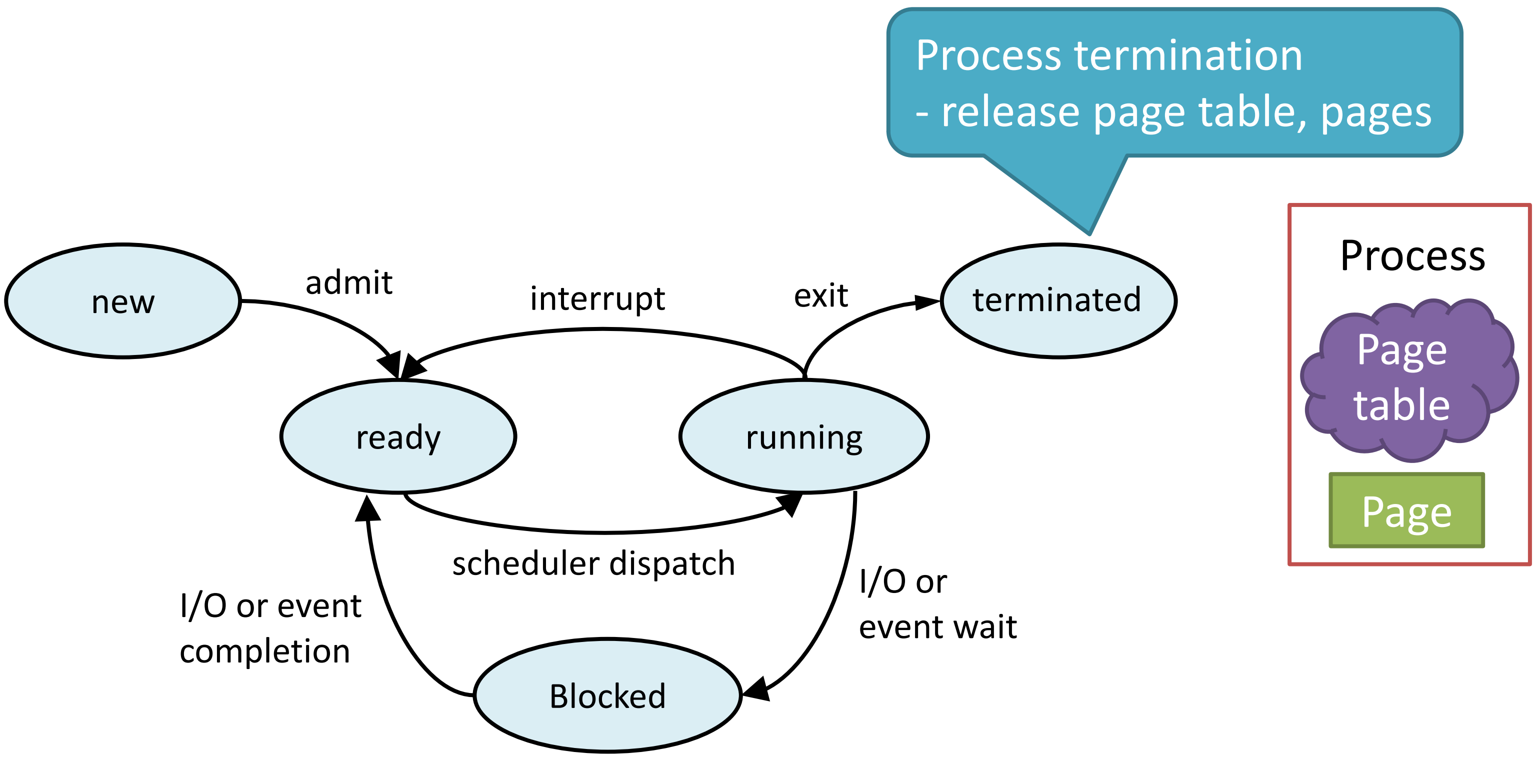
- During the process execution, how does the process interact with the page memory system?

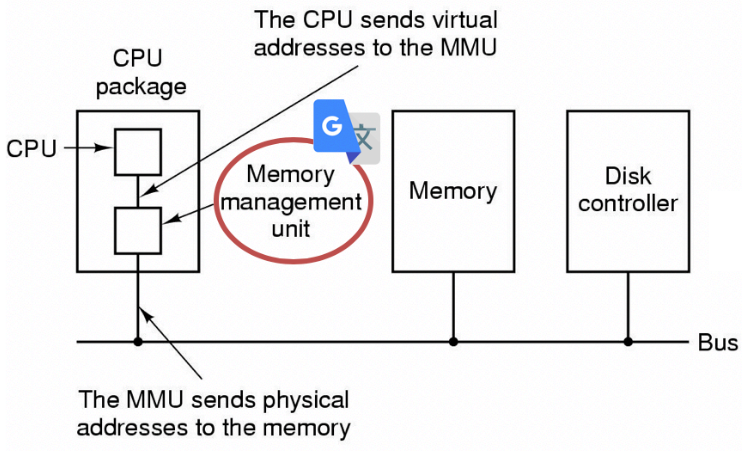


Paging with Process Life Cycle



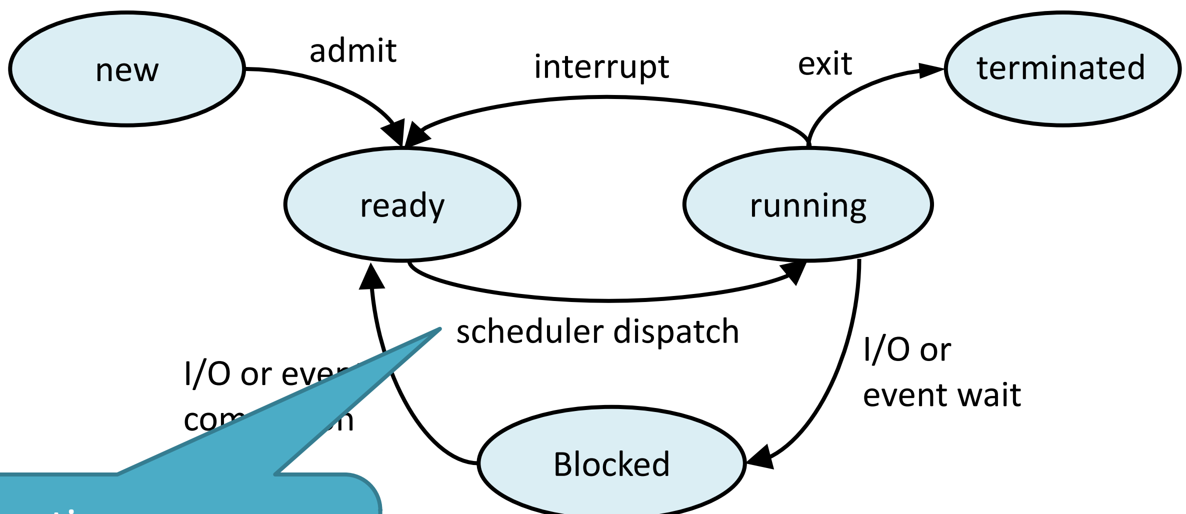
Paging with Process Life Cycle



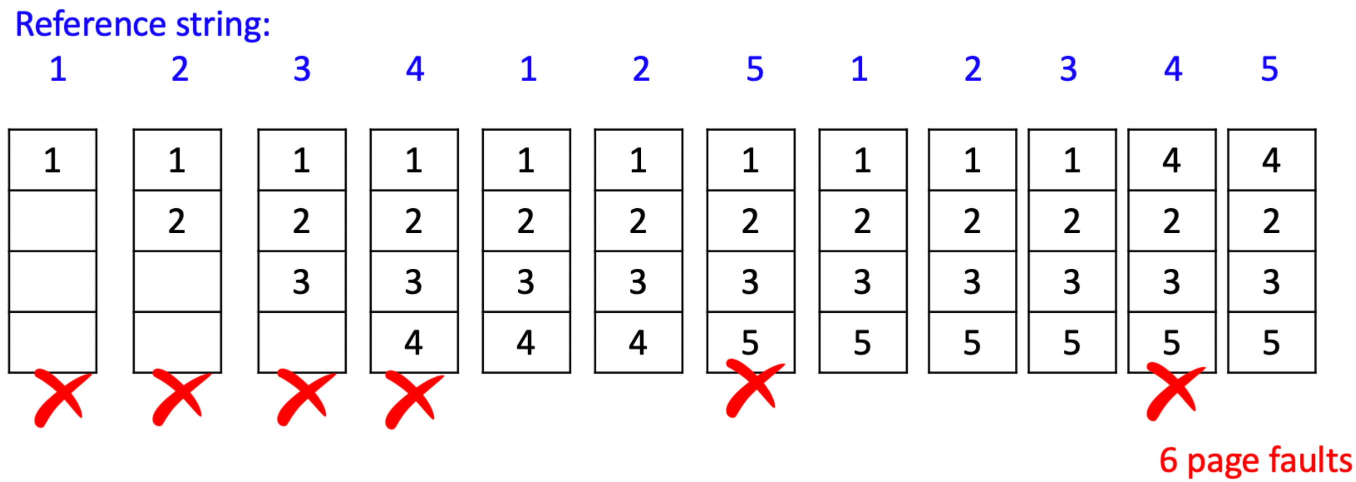


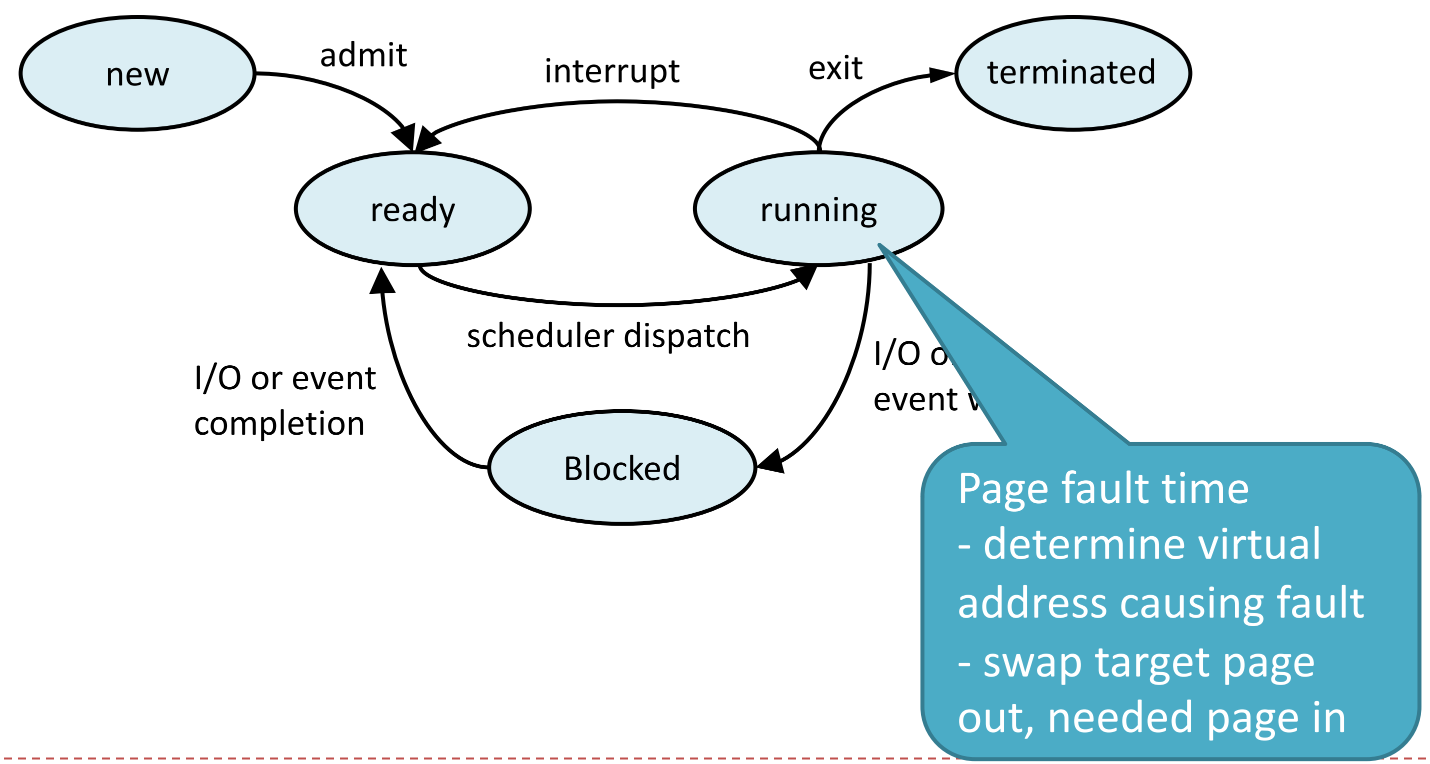
Address translation done by MMU

Keep updating as address changes



Process execution
- MMU reset for new process
- TLB flushed

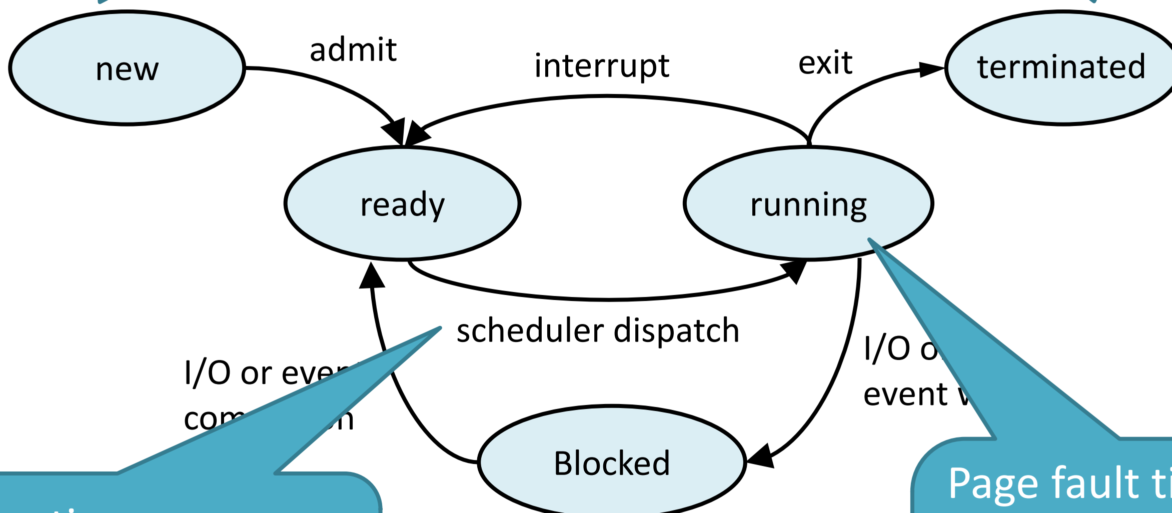

Reference string:
1 2 3 4 1 2 5 1 2 3 4 5
6 page faults


new
admit
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O o
event w
Blocked
Page fault time
- determine virtual address causing fault
- swap target page out, needed page in

Paging with Process Life Cycle

Process creation:
- determine program size
- create page table

Process termination
- release page table, pages



Process execution
- MMU reset for new process
- TLB flushed

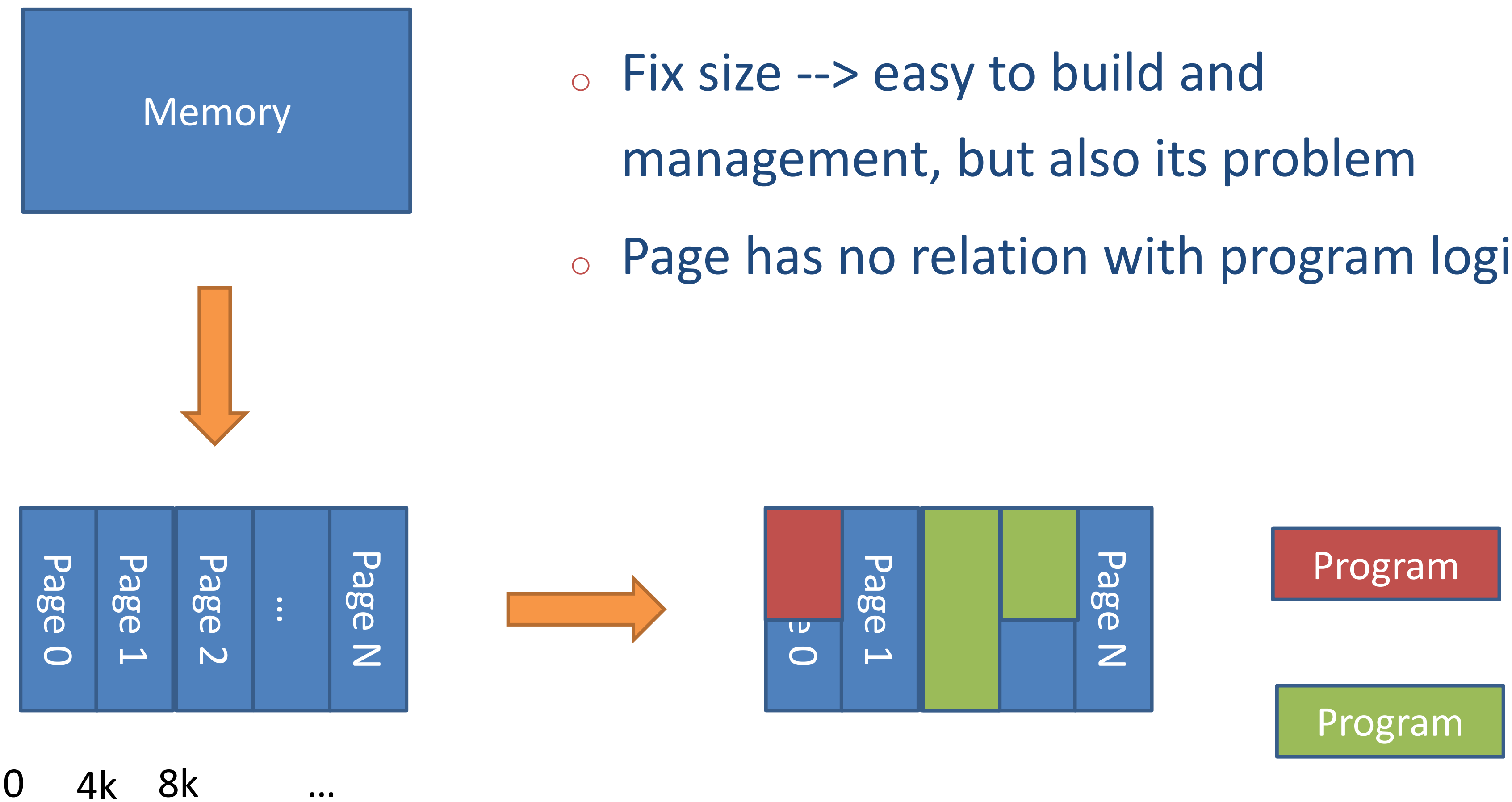
Page fault time
- determine virtual address causing fault
- swap target page out, needed page in

Outline

- Page design
  - Internal fragmentation vs. External fragmentation
  - Local page replacement vs. Global page replacement
  - Page size small vs. Large
  - Shared page
  - Paging with Process life cycle
- Segmentation
  - Page vs. Segmentation

Segmentation: Rethink Pages

- Divide memory into pages
  - Fix size --> easy to build and management, but also its problem
  - Page has no relation with program logic



Segmentation

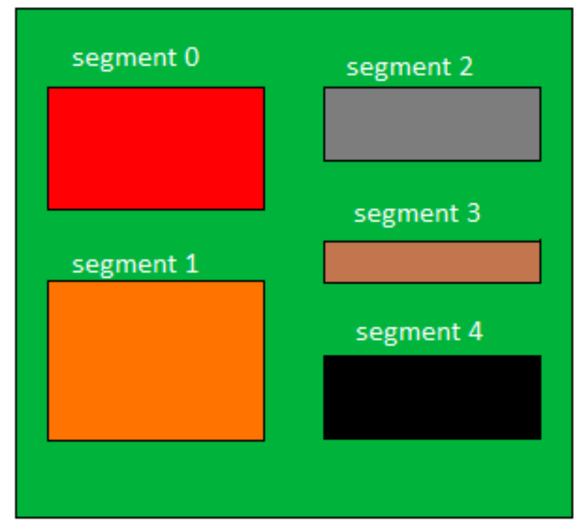
The user program address is divided into several segments of different sizes

Each segment can define a relatively complete set of logical information

Segment 0

Segment 2

Segment 4

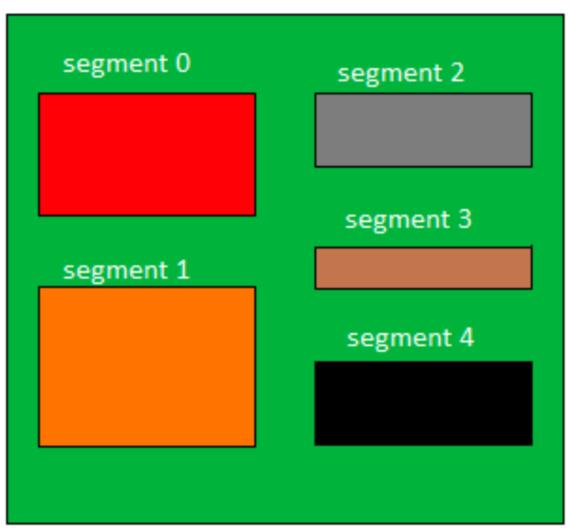


Logical Address Space

Segmentation

Allows each segment to grow or shrink, independently

- When storing allocation in segments, they can be non-contiguous in memory, and under discrete allocation



Logical Address Space

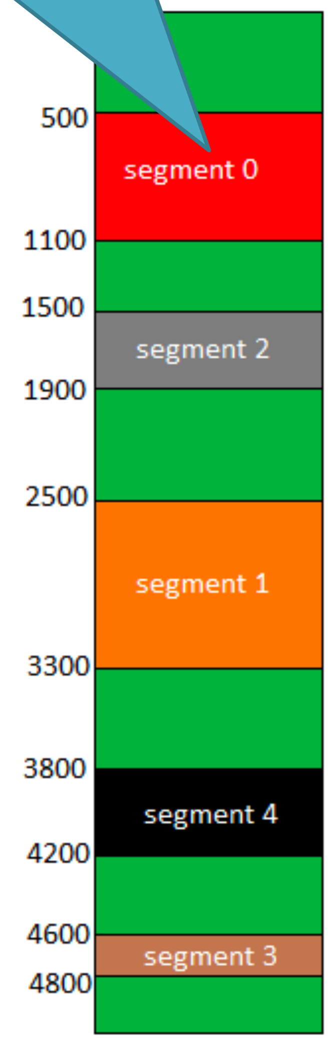
Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table

start

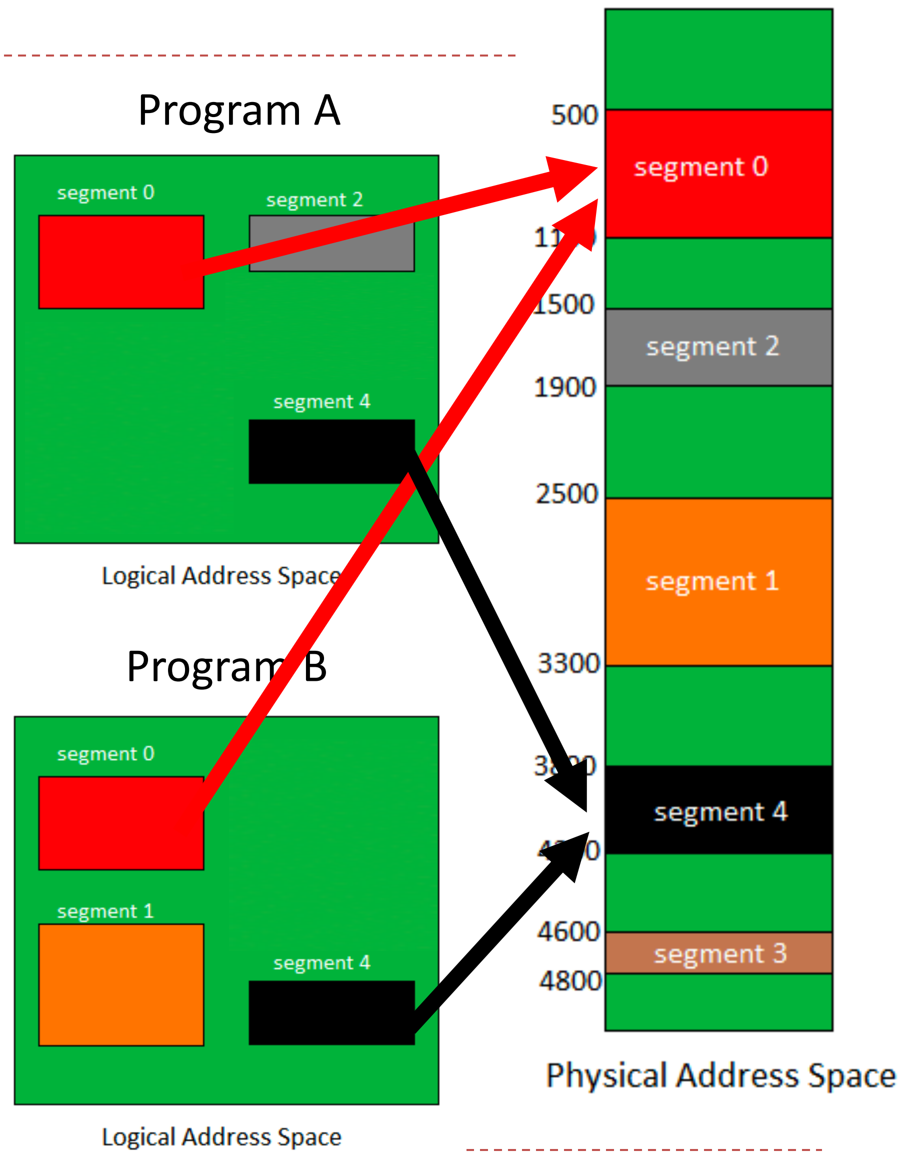
length



Physical Address Space

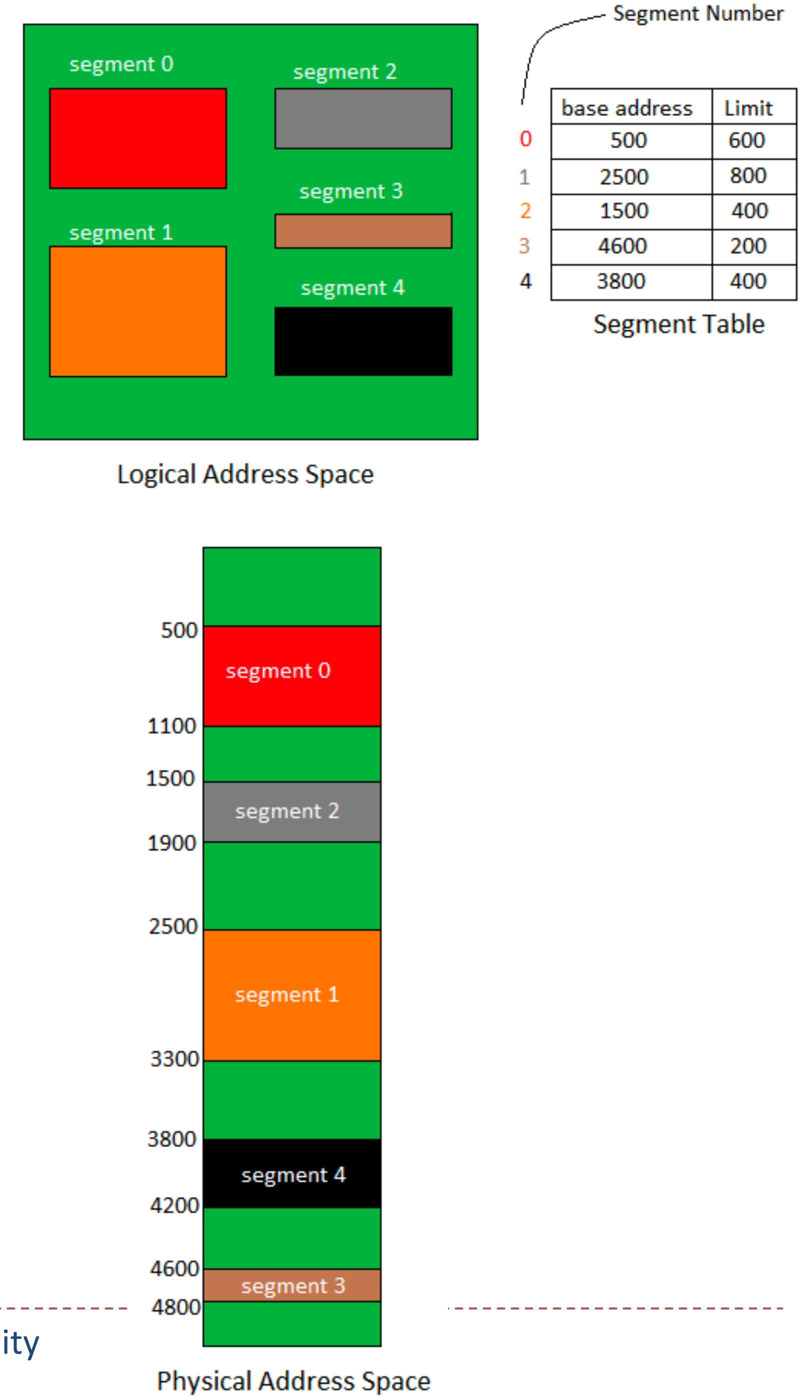
Segmentation

- Segmentation is convenient for multi-program sharing
  - E.g., Program A and B share the same piece of code or data in segment 0 and 4
  - Convenient for programming



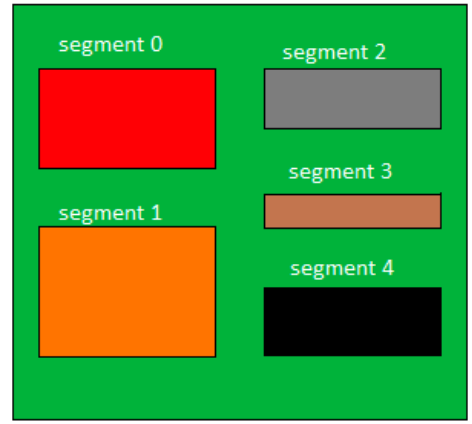
Segmentation

- Advantages:
  - The **logical independence** of the segments makes it easy to compile, manage, modify, and protect, and is also convenient for multi-program sharing
  - The segment length can be **dynamically** changed as needed, allowing free scheduling to make efficient use of the main memory space
  - **Convenient** for programming, including segment sharing, segmentation protection, dynamic linking, dynamic growth



Segmentation

- Disadvantages:
  - The **allocation** of memory space is difficult (what size, where, etc.)
  - It is easy to leave a lot of **fragments** between the segments, resulting in a decrease in space utilization

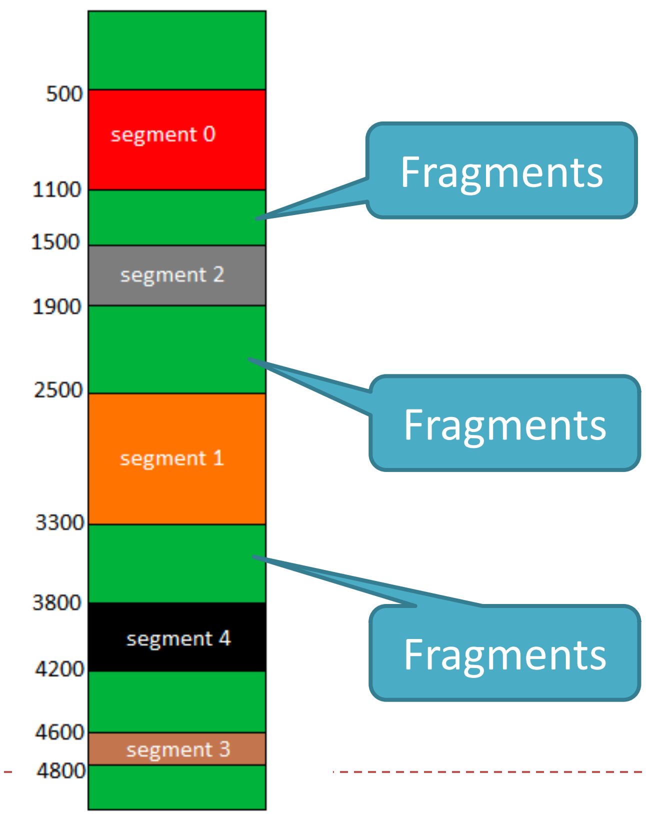


Logical Address Space

Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

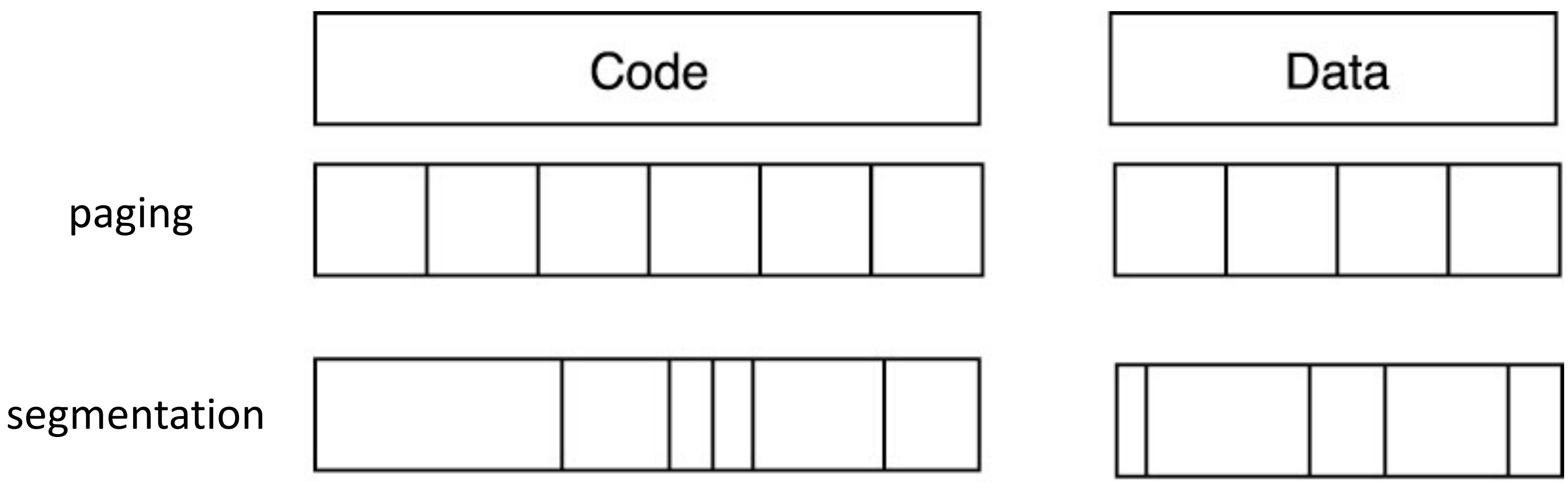
Segment Table



Physical Address Space

Paged vs. Segmented Virtual Memory

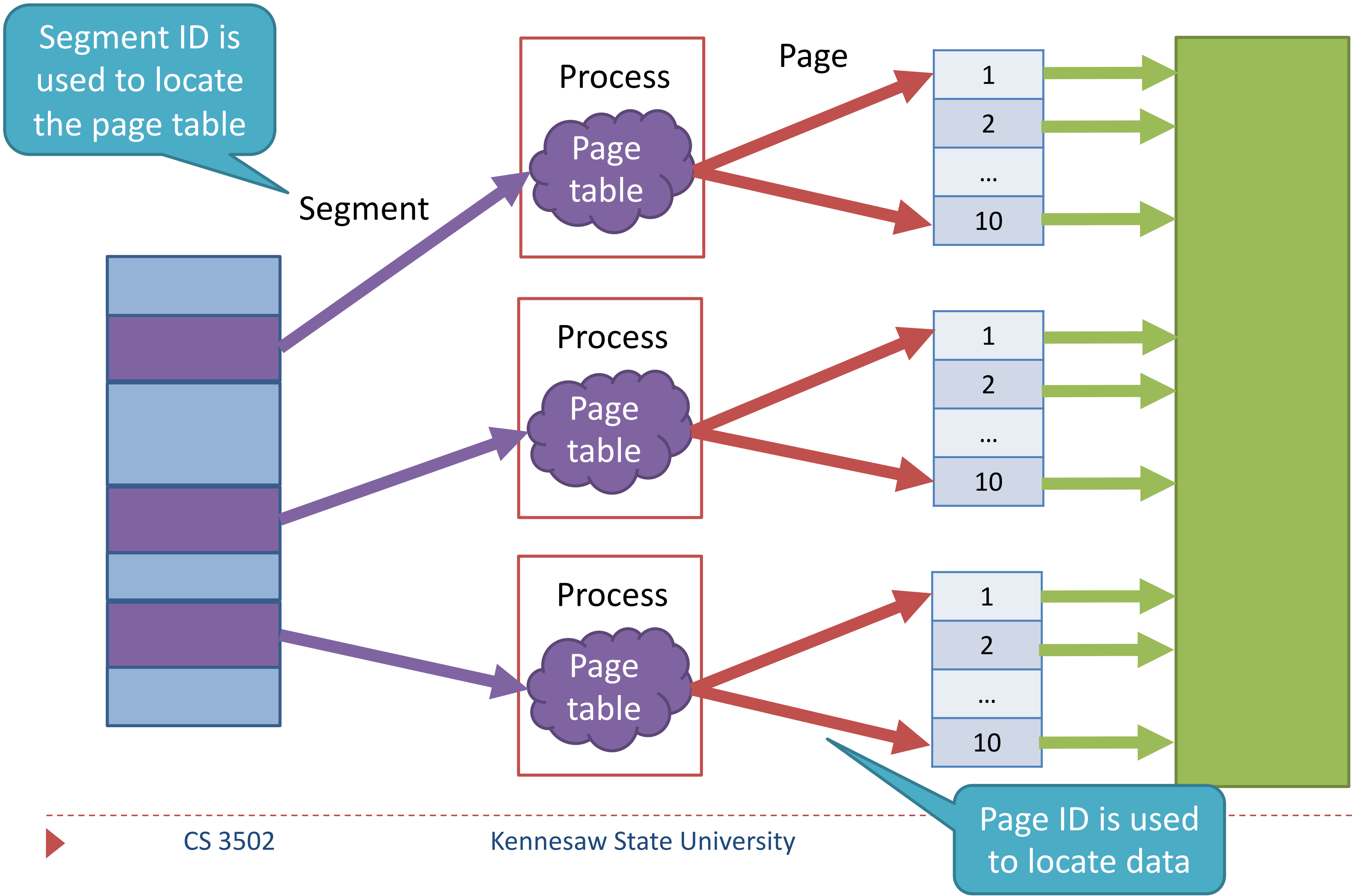
- Paged virtual memory
  - Memory divided into fixed sized pages
- Segmented virtual memory
  - Memory divided into variable length segments



Comparison of Segmentation and Paging

	Page	Segment
Definition	Main memory is partitioned into same sized pages	Main memory is partitioned into various segments
Address	One word (start address)	Two words (start and end address)
Programmer visible	No	Yes
Block replacement	Easy	Hard
Fragmentation	Internal	External

Combination of Page and Segment



Conclusion

- Page design
  - Internal fragmentation vs. External fragmentation
  - Local page replacement vs. Global page replacement
  - Page size small vs. Large
  - Shared page
  - Paging with Process life cycle
- Segmentation
  - Page vs. Segmentation