

Clock Page Replacement

2024 Winter ECE 353: Systems Software
Jon Eyolfson

Lecture 30
2.0.0

Clock Algorithm

Data structures:

- Keeps a circular list of pages in memory
- Uses a reference bit for each page in memory (light grey in next slides)
- Has a "hand" (iterator) pointing to the last element examined

Algorithm, to insert a new page:

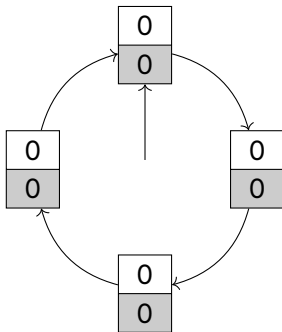
- Check the hand's reference bit, if it's 0 then place the page and advance hand
- If the reference bit is 1, set it to 0, advance the hand, and repeat

For page accesses, set the reference bit to 1

Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

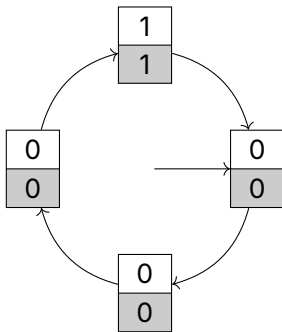


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1

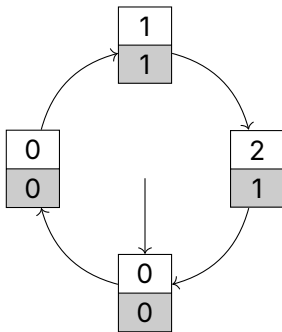


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2

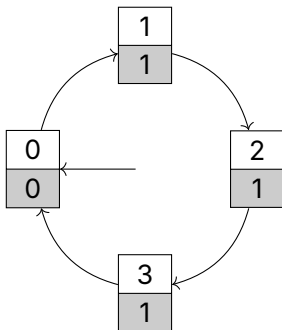


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3

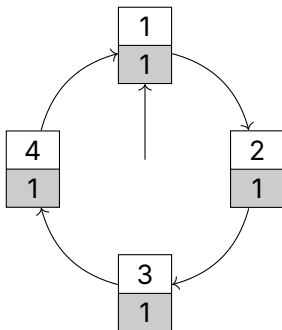


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4

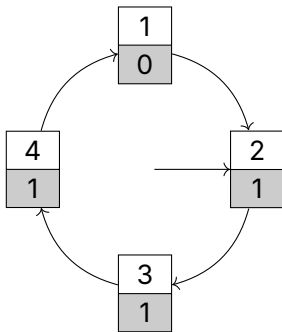


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5

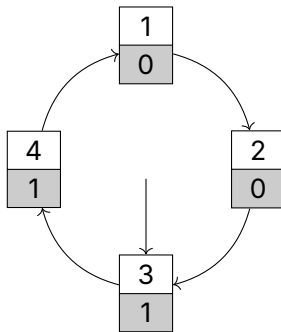


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5

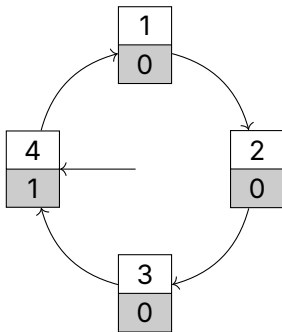


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5

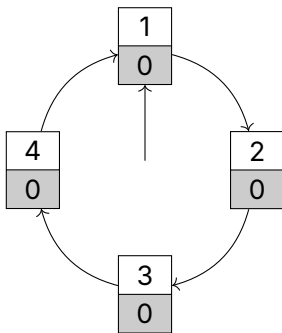


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5

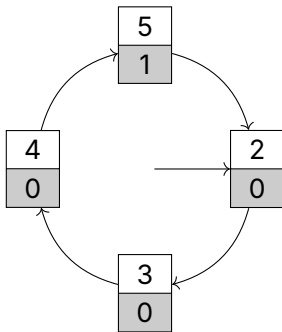


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5

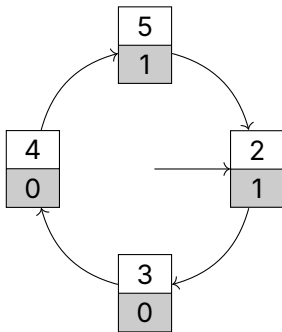


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2

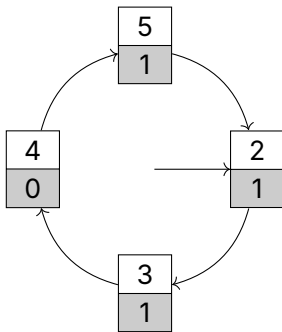


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3

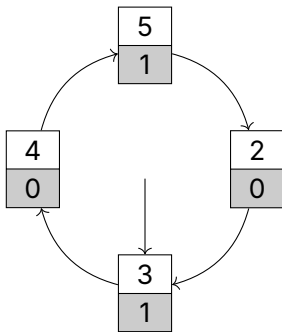


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3 1

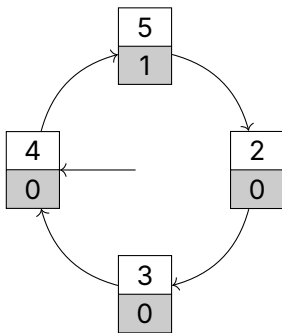


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3 1

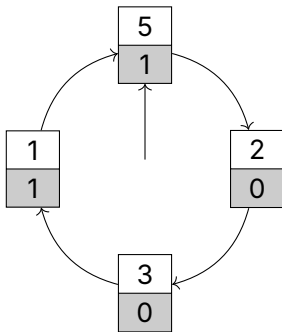


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3 1

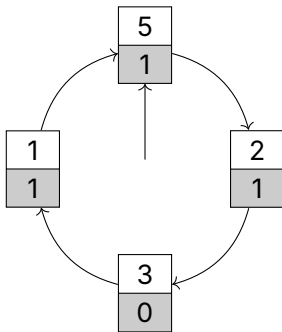


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3 1 2

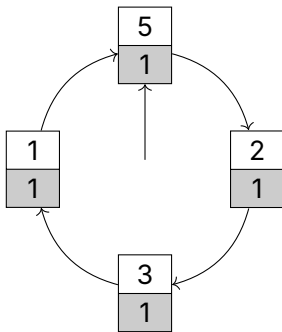


Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

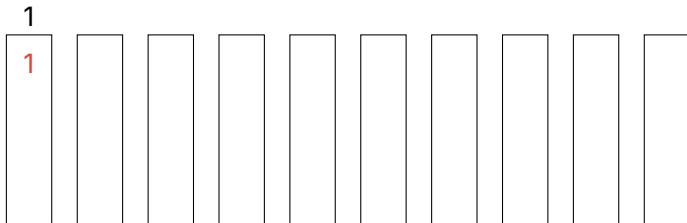
1 2 3 4 5 2 3 1 2 3



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

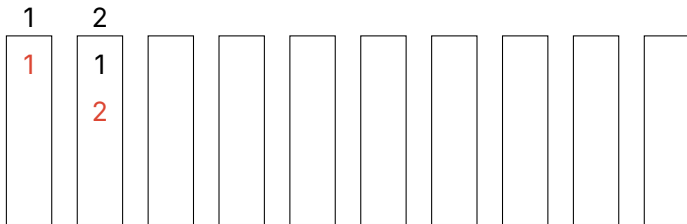
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

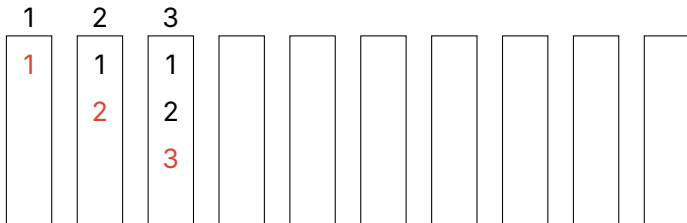
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

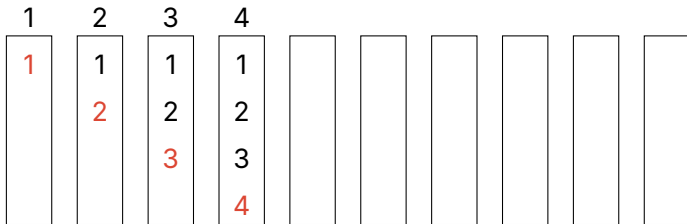
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

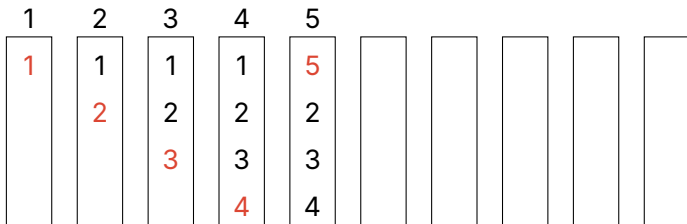
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

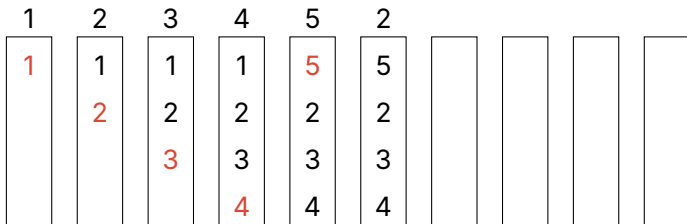
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

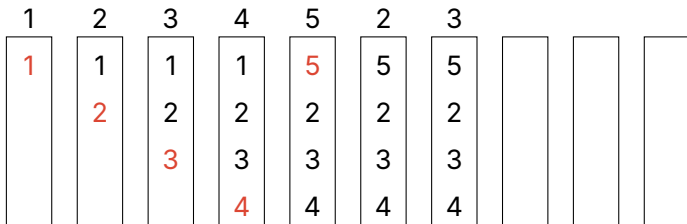
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

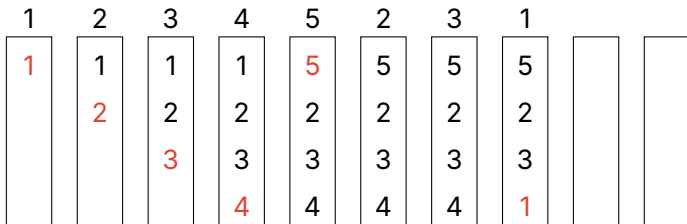
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

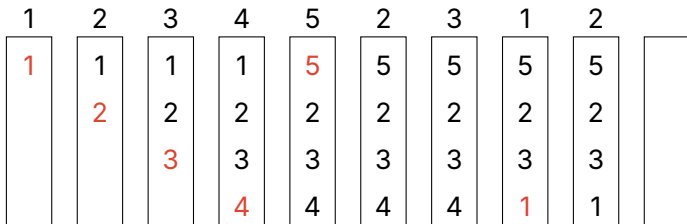
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1	2	3	4	5	2	3	1	2	3
1	1	1	1	5	5	5	5	5	5
	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3
			4	4	4	4	1	1	1

Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:

1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1	2	3	4	5	2	3	1	2	3
1	1	1	1	5	5	5	5	5	5
	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3
			4	4	4	4	1	1	1

6 page faults

For Performance You May Choose to Disable Swapping

Memory is cheap, and has quite high capacity

You'd rather know you need more memory than run slowly

Linux runs an OOM (out of memory) killer, that SIGKILLS the memory hog

Larger page sizes allow for speedups (2 MiB or 1 GiB page size)

Trade more fragmentation for more TLB coverage

The Clock Algorithm is an Approximation of LRU

Data structures:

- Keeps a circular list of pages in memory
- Uses a reference bit for each page in memory (light grey in next slides)
- Has a "hand" (iterator) pointing to the last element examined

Algorithm, to insert a new page:

- Check the hand's reference bit, if it's 0 then place the page and advance hand
- If the reference bit is 1, set it to 0, advance the hand, and repeat

For page accesses, set the reference bit to 1