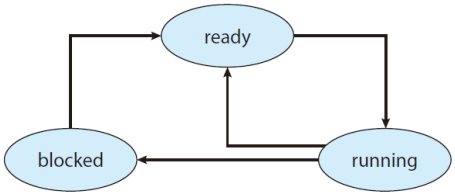
Operating System Homework 10

Jinyan Xu, 3160101126, Information Security

10.16 A simplified view of thread states is ready, running, and blocked, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O).



Assuming a thread is in the **running state**, answer the following questions, and explain your answers:

a. Will the thread change state if it incurs a page fault? If so, to what state will it change?

b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?

c. Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

Answer:

a) Yes, the state will change to blocked, as an I/O operation is required to bring the new page into memory.

b) No, the thread continues running if the address is resolved in the page table.

c) No, the thread continues running if the address is resolved in the page table.

10.24 Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings:

- 2, 6, 9, 2, 4, 2, 1, 7, 3, 0, 5, 2, 1, 2, 9, 5, 7, 3, 8, 5
- 0, 6, 3, 0, 2, 6, 3, 5, 2, 4, 1, 3, 0, 6, 1, 4, 2, 3, 5, 7
- 3, 1, 4, 2, 5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1
- 4, 2, 1, 7, 9, 8, 3, 5, 2, 6, 8, 1, 0, 7, 2, 4, 1, 3, 5, 8
- 0, 1, 2, 3, 4, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 4, 3, 2, 1, 0

Indicate the number of page faults for each algorithm assuming demand paging with three frames.

Answer:

a) 2, 6, 9, 2, 4, 2, 1, 7, 3, 0, 5, 2, 1, 2, 9, 5, 7, 3, 8, 5

FIFO: 18 page faults

2	6	9	2	4	2	1	7	3	0	5	2	1	2	9	5	7	3	8	5
2	2	2	2	4	4	4	7	7	7	5	5	5	5	9	9	9	3	3	3
	6	6	6	6	2	2	2	3	3	3	2	2	2	2	5	5	5	8	8
		9	9	9	9	1	1	1	0	0	0	1	1	1	1	7	7	7	5

LRU: 17 page faults

2	6	9	2	4	2	1	7	3	0	5	2	1	2	9	5	7	3	8	5
2	2	2	2	2	2	2	2	3	3	3	2	2	2	2	2	7	7	7	5
	6	6	6	4	4	4	7	7	7	5	5	5	5	9	9	9	3	3	3
		9	9	9	9	1	1	1	0	0	0	1	1	1	5	5	5	8	8

OPT: 13 page faults

2	6	9	2	4	2	1	7	3	0	5	2	1	2	9	5	7	3	8	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	9	9	9	3	3	3
	6	6	6	4	4	1	1	1	1	1	1	1	1	1	1	7	7	8	8
		9	9	9	9	9	7	3	0	5	5	5	5	5	5	5	5	5	5

b) 0, 6, 3, 0, 2, 6, 3, 5, 2, 4, 1, 3, 0, 6, 1, 4, 2, 3, 5, 7

FIFO: 16 page faults

0	6	3	0	2	6	3	5	2	4	1	3	0	6	1	4	2	3	5	7
0	0	0	0	2	2	2	2	2	2	1	1	1	6	6	6	2	2	2	7
	6	6	6	6	6	6	5	5	5	5	3	3	3	1	1	1	3	3	3
		3	3	3	3	3	3	3	4	4	4	0	0	0	4	4	4	5	5

LRU: 19 page faults

0	6	3	0	2	6	3	5	2	4	1	3	0	6	1	4	2	3	5	7
0	0	0	0	0	0	3	3	3	4	4	4	0	0	0	4	4	4	5	5
	6	6	6	2	2	2	5	5	5	1	1	1	6	6	6	2	2	2	7
		3	3	3	6	6	6	2	2	2	3	3	3	1	1	1	3	3	3

OPT: 13 page faults

0	6	3	0	2	6	3	5	2	4	1	3	0	6	1	4	2	3	5	7
0	0	0	0	2	2	2	2	2	2	1	1	1	1	1	1	2	2	2	7
	6	6	6	6	6	6	5	5	4	4	4	4	4	4	4	4	3	3	3
		3	3	3	3	3	3	3	3	3	3	0	6	6	6	6	6	5	5

c) 3, 1, 4, 2, 5, 4, 1, 3, 5, 2, 0, 1, 1, 0, 2, 3, 4, 5, 0, 1

FIFO: 15 page faults

3	1	4	2	5	4	1	3	5	2	0	1	1	0	2	3	4	5	0	1
3	3	3	2	2	2	2	3	3	3	3	1	1	1	1	1	1	5	5	5
	1	1	1	5	5	5	5	5	2	2	2	2	2	2	3	3	3	0	0
		4	4	4	4	1	1	1	1	0	0	0	0	0	0	4	4	4	1

LRU: 16 page faults

3	1	4	2	5	4	1	3	5	2	0	1	1	0	2	3	4	5	0	1
3	3	3	2	2	2	1	1	1	2	2	2	2	2	2	2	2	5	5	5
	1	1	1	5	5	5	3	3	3	0	0	0	0	0	0	4	4	4	1
		4	4	4	4	4	4	5	5	5	1	1	1	1	3	3	3	0	0

OPT: 11 page faults

3	1	4	2	5	4	1	3	5	2	0	1	1	0	2	3	4	5	0	1
3	3	3	2	5	5	5	5	5	2	2	2	2	2	2	3	4	5	5	5
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		4	4	4	4	4	3	3	3	0	0	0	0	0	0	0	0	0	0

d) 4, 2, 1, 7, 9, 8, 3, 5, 2, 6, 8, 1, 0, 7, 2, 4, 1, 3, 5, 8

FIFO: 20 page faults

4	2	1	7	9	8	3	5	2	6	8	1	0	7	2	4	1	3	5	8
4	4	4	7	7	7	3	3	3	6	6	6	0	0	0	4	4	4	5	5
	2	2	2	9	9	9	5	5	5	8	8	8	7	7	7	1	1	1	8
		1	1	1	8	8	8	2	2	2	1	1	1	2	2	2	3	3	3

LRU: 20 page faults

4	2	1	7	9	8	3	5	2	6	8	1	0	7	2	4	1	3	5	8
4	4	4	7	7	7	3	3	3	6	6	6	0	0	0	4	4	4	5	5
	2	2	2	9	9	9	5	5	5	8	8	8	7	7	7	1	1	1	8
		1	1	1	8	8	8	2	2	2	1	1	1	2	2	2	3	3	3

OPT: 16 page faults

4	2	1	7	9	8	3	5	2	6	8	1	0	7	2	4	1	3	5	8
4	4	4	7	9	8	8	8	8	8	8	8	0	7	7	4	4	4	4	8
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3
		1	1	1	1	3	5	5	6	6	1	1	1	1	1	1	1	5	5

e) 0, 1, 2, 3, 4, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 4, 3, 2, 1, 0

FIFO: 12 page faults

0	1	2	3	4	4	3	2	1	0	0	1	2	3	4	4	3	2	1	0
0	0	0	3	3	3	3	3	3	0	0	0	0	0	4	4	4	4	4	4
	1	1	1	4	4	4	4	4	4	4	4	2	2	2	2	2	2	1	1
		2	2	2	2	2	2	1	1	1	1	1	3	3	3	3	3	3	0

LRU: 11 page faults

0	1	2	3	4	4	3	2	1	0	0	1	2	3	4	4	3	2	1	0
0	0	0	3	3	3	3	3	3	0	0	0	0	3	3	3	3	3	3	0
	1	1	1	4	4	4	4	1	1	1	1	1	1	4	4	4	4	1	1
		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

OPT: 11 page faults

0	1	2	3	4	4	3	2	1	0	0	1	2	3	4	4	3	2	1	0
0	0	0	3	3	3	3	3	3	0	0	0	0	3	3	3	3	3	1	1
	1	1	1	4	4	4	4	1	1	1	1	1	1	4	4	4	4	4	0
		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

10.35 A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. **We can associate with each page frame a counter of the number of pages associated with that frame.** Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:

- What is the initial value of the counters?
- When are counters increased?
- When are counters decreased?

- How is the page to be replaced selected?

b. How many page faults occur for your algorithm for the following reference string with four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

c. What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames?

Answer:

a) The initial value of the counters is 0. When a new page is associated with the frame, its counter is increased. When a page has been replaced from the memory, and this page won't use any more, its counter is decreased. Find a frame with the smallest counter, using a FIFO queue when have same counter.

b) 14 page faults

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7
1^1	1^1	1^1	1^1	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2
	2^1	2^1	2^1	2^1	2^1	2^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
		3^1	3^1	3^1	3^1	3^1	3^1	6^1	6^1	8^1	8^1	8^1	9^2	9^2
			4^1	4^1	4^1	4^1	4^1	4^1	7^2	7^2	7^2	7^2	7^2	7^2

8	9	5	4	5	4	2
8^3	8^3	8^2	8^1	8^1	8^1	2^1
1^2	1^2	5^2	5^2	5^2	5^2	5^2
9^2	9^2	9^2	4^1	4^1	4^1	4^1
7^2	7^2	7^2	7^2	7^2	7^2	7^2

- **Green unit means page fault; yellow unit means other counters are decreased by this replace.**
- **Orange page number means the last time use for this page.**

c) 11 page faults

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7
1	1	1	1	1	1	1	1	6	7	7	7	7	7	7
	2	2	2	5	5	5	5	5	5	5	5	5	5	5
		3	3	3	3	3	3	3	3	8	8	8	8	8
			4	4	4	4	4	4	4	4	4	4	9	9

8	9	5	4	5	4	2
7	7	7	4	4	4	4
5	5	5	5	5	5	2
8	8	8	8	8	8	8
9	9	9	9	9	9	9

10.36 Consider a demand-paging system with a paging disk that has an average access and transfer time of **20 milliseconds**. Addresses are translated through a page table in main memory, with an access time of **1 microsecond** per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference if the page-table entry is in the associative memory.

Assume that 80 percent of the accesses are in the associative memory and that, of those remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

Answer:

EAT = 0.8 * 1 + 0.2 * (0.9 * 2 + 0.1 * 20002)
= 0.8 + 0.2 * (1.8 + 2000.2)
= 0.8 + 400.4
= 401.2 μs

10.40 In a 1,024-KB segment, memory is allocated using the buddy system. Using Figure 10.26 as a guide, draw a tree illustrating how the following memory requests are allocated:

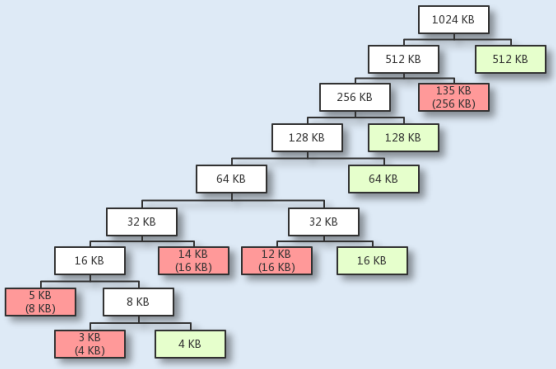
- Request 5 KB
- Request 135 KB.
- Request 14 KB.
- Request 3 KB.
- Request 12 KB.

Next, modify the tree for the following releases of memory. Perform coalescing whenever possible:

- Release 3 KB.
- Release 5 KB.
- Release 14 KB.
- Release 12 KB.

Answer:

Allocation:



Release: