# 现代操作系统第三次作业

# Modern Operating Systems homework

陈涵 2019 级软件工程 1 班 201936380086

## CHAPTER 3| MEMORY MANAGEMENT

**4.** Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of

(a) 12 MB

(b) 10 MB

(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

**Answer:**

| | (a) 12 MB | (b) 10 MB | (c) 9 MB |
|---|---|---|---|
| First fit | 20 MB | 10 MB | 18 MB |
| Best fit | 12 MB | 10 MB | 9 MB |
| Worst fit | 20 MB | 18 MB | 15 MB |
| Next fit | 20 MB | 18 MB | 9 MB |

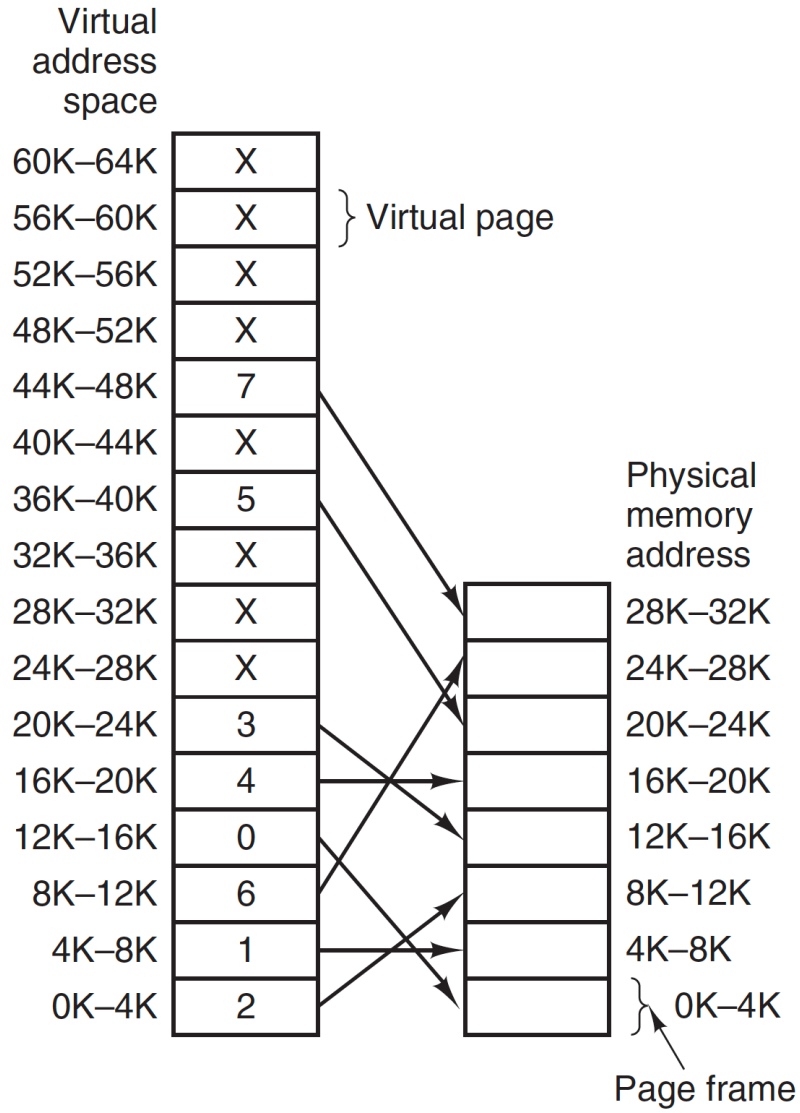
**6.** For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4-KB page and for an 8-KB page: 20000, 32768, 60000.

**Answer:**

page number $= \frac{virtual\ address}{page\ size}$ offset $= virtual\ address\ \boldsymbol{mod}\ page\ size$

| | 20000 | 32768 | 60000 |
|---|---|---|---|
| 4-KB page | (4, 3616) | (8, 0) | (14, 2656) |
| 8-KB page | (2, 3616) | (4, 0) | (7, 2656) |

**7.** Using the page table of Fig. 3-9, give the physical address corresponding to each of the following virtual addresses:




(a) 20
(b) 4100
(c) 8300

**Answer:**

(a) 20-0+8K=20+8192
(b) 4100-4K+4K=4100
(c) 8300-8K+24K=8300+16384=24684

| virtual addresses | 20 | 4100 | 8300 |
|---|---|---|---|
| physical address | 8212 | 4100 | 24684 |

**17.** Suppose that a machine has 38-bit virtual addresses and 32-bit physical addresses.
(a) What is the main advantage of a multilevel page table over a single-level one?
(b) With a two-level page table, 16-KB pages, and 4-byte entries, how many bits should be allocated for the top-level page table field and how many for the next level page table field? Explain.

**Answer:**

(a)

Using multilevel page table can reduce the number of actual pages of page table that need to be in memory because of its hierarchic structure. In a program with numbers of data, we only need the top-level page table, an instruction page and one data page.

(b)

16KB = $16 \cdot 1024 = 2^{14}$ bytes, needs 14bits to address one page. That leaves 24 bits for the page fields. Since one entry has 4 bytes, one page can hold $2^{12}$ page table entries, so 12 bits will be allocated for the top-level page fields and 12 bits for the next level page table fields.

**19.** A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second-level page table field, and an offset. How large are the pages and how many are there in the address space?

**Answer:**

In the 32 bits, 9 bits for the top-level page, 11 bits for the second-level page, 20 bits are used for page number which leaves 12 bits for the offset. Hence one page has $2^{12}$ bytes = 4KB. 20 bits for page number, so the quantity of pages is $2^{20} = 1048576$.

**25.** A computer with an 8-KB page, a 256-KB main memory, and a 64-GB virtual address space uses an inverted page table to implement its virtual memory. How big should the hash table be to ensure a mean hash chain length of less than 1? Assume that the hash-table size is a power of two.

**Answer:**

The main memory has $\frac{256KB}{8KB} = 32$ pages. A 32 bits hash table will have a mean chain length of 1. Since the hash-table size is a power of 2, to make it under 1, we have to go to the next size, 64 bits. Spreading 32 entries over 64 table slots will give a mean chain length of 0.5, which ensures fast lookup.

**28.** If FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with the reference string 0172327103 if the four frames are initially empty? Now repeat this problem for LRU.

**Answer:**

Using FIFO page replacement algorithm:

reference: **0**172327103

| | | | |
|---|---|---|---|
| X | X | X | X |

page fault: 1 | page0 load on page frame0

| | | | |
|---|---|---|---|
| 0 | X | X | X |

reference: 0**1**72327103

| | | | |
|---|---|---|---|
| 0 | X | X | X |

page fault: 2 | page1 load on page frame1

| | | | |
|---|---|---|---|
| 0 | 1 | X | X |

reference: 01**7**2327103

| | | | |
|---|---|---|---|
| 0 | 1 | X | X |

page fault: 3 | page7 load on page frame2

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | X |

reference: 017**2**327103

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | X |

page fault: 4 | page2 load on page frame3

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | 2 |

reference: 0172**3**27103

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | 2 |

page fault: 5 | page3 load on page frame0

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

reference: 01723**2**7103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page2 is at page frame3, no page fault occurs.

reference: 017232**7**103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page7 is at page frame2, no page fault occurs.

reference: 0172327**1**03

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page1 is at page frame1, no page fault occurs.

reference: 01723271**0**3

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 6 | page0 load on page frame1

| | | | |
|---|---|---|---|
| 3 | 0 | 7 | 2 |

reference: 0172327103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 6 | page3 is at page frame0, no page fault occurs.

using FIFO page replacement algorithm, 6 page faults will occur with reference string 0172327103.

Using LRU page replacement algorithm:

reference: **0**172327103

| | | | |
|---|---|---|---|
| X | X | X | X |

page fault: 1 | page0 load on page frame0

| | | | |
|---|---|---|---|
| 0 | X | X | X |

reference: 0**1**72327103

| | | | |
|---|---|---|---|
| 0 | X | X | X |

page fault: 2 | page1 load on page frame1

| | | | |
|---|---|---|---|
| 0 | 1 | X | X |

reference: 01**7**2327103

| | | | |
|---|---|---|---|
| 0 | 1 | X | X |

page fault: 3 | page7 load on page frame2

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | X |

reference: 017**2**327103

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | X |

page fault: 4 | page2 load on page frame3

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | 2 |

reference: 0172**3**27103

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | X |

page fault: 5 | page0 is least recently used, page3 load on page frame0

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

reference: 01723**2**7103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page2 is at page frame3, no page fault occurs.

reference: 017232**7**103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page7 is at page frame2, no page fault occurs.

reference: 0172327**1**03

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 5 | page1 is at page frame1, no page fault occurs.

reference: 01723271**0**3

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 6 | page3 is least recently used, page0 load on page frame0

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | 2 |

reference: 0172327103

| | | | |
|---|---|---|---|
| 3 | 1 | 7 | 2 |

page fault: 7 | page2 is least recently used, page3 load on page frame3

| | | | |
|---|---|---|---|
| 0 | 1 | 7 | 3 |

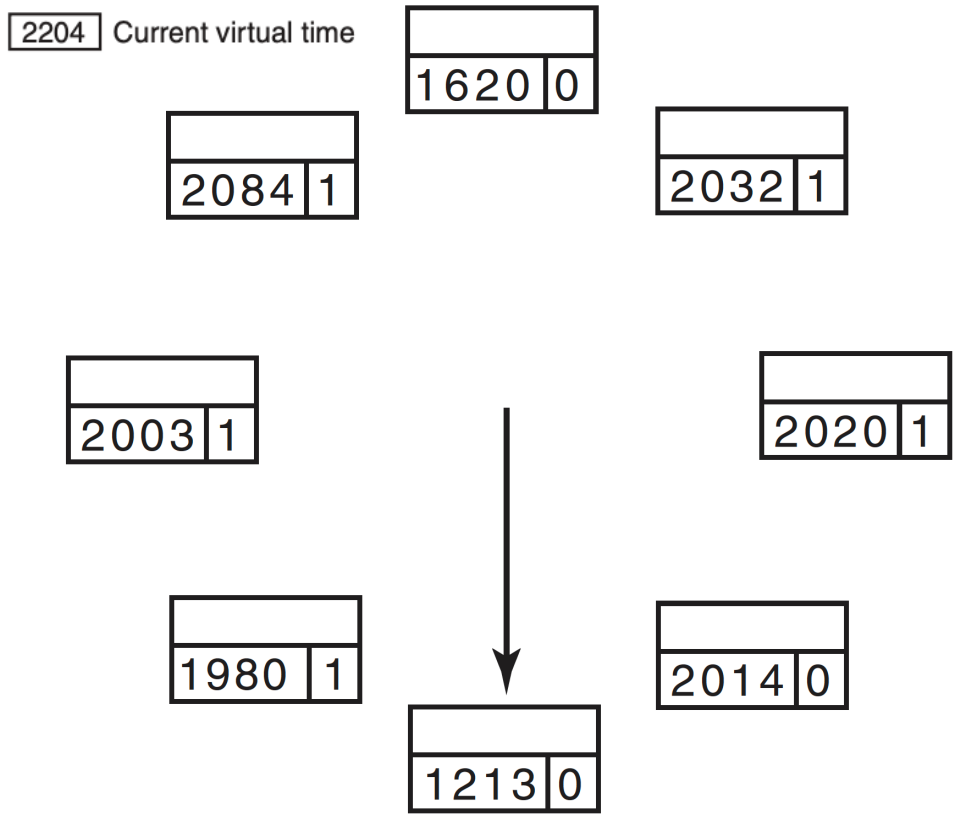
above all, using FIFO page replacement algorithm, 6 page faults will occur. Using LRU page replacement algorithm, 7 page faults will occur with reference string 0172327103.

**30.** A small computer on a smart card has four page frames. At the first clock tick, the R bits are 0111 (page 0 is 0, the rest are 1). At subsequent clock ticks, the values are 1011, 1010, 1101, 0010, 1010, 1100, and 0001. If the aging algorithm is used with an 8-bit counter, give the values of the four counters after the last tick.

**Answer:**

| R bits in i th ticks | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Page 0: | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| Page 1: | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Page 2: | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Page 3: | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**32.** In the WSClock algorithm of Fig. 3-20(c), the hand points to a page with R = 0. If τ = 400, will this page be removed? What about if τ = 1000?




(c)

**Answer:**

Current virtual time is 2204, 2204-1213=991. The age of this page is 991. If τ=400, this page will be removed from the working set. If τ=1000, the page's age is lower than τ, so it will stay at working set.

**36.** A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):

| Page | loaded | Last ref. | R | M |
|---|---|---|---|---|
| 0 | 126 | 280 | 1 | 0 |
| 1 | 230 | 265 | 0 | 1 |
| 2 | 140 | 270 | 0 | 0 |
| 3 | 110 | 285 | 1 | 1 |

(a) Which page will NRU replace?
(b) Which page will FIFO replace?
(c) Which page will LRU replace?
(d) Which page will second chance replace?

**Answer:**

(a)
Page 0 belongs to Class 2: referenced, not modified.
Page 1 belongs to Class 1: not referenced, modified.
Page 2 belongs to Class 0: not referenced, not modified.
Page 3 belongs to Class 3: referenced, modified.
Above all, page 2 will be removed by NRU.

(b)
The loaded sequence is page 3, page 0, page 2, page 1, so FIFO will remove page 3.

(c)
According to *'Last ref.'* column, page 1 is the least recently used page which will be removed by LRU.

(d)
Page 2 isn't used in the second chance, so second chance algorithm will replace it.

**38.** Consider the following two-dimensional array:

```
int  X[64][64];
```

Suppose that a system has four-page frames and each frame is 128 words (an integer occupies one word). Programs that manipulate the X array fit into exactly one page and always occupy page 0. The data are swapped in and out of the other three frames. The X array is stored in row-major order (i.e., X[0][1] follows X[0][0] in memory). Which of the two code fragments shown below will generate the lowest number of page faults? Explain and compute the total number of page faults.

*Fragment A*

```
for (int j = 0; j < 64; j++)
      for (int i = 0; i < 64; i++) X[i][j] = 0;
```

*Fragment B*

```
for (int i = 0; i < 64; i++)
      for (int j = 0; j < 64; j++) X[i][j] = 0;
```

**Answer:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Fragment B will generate the lowest number of page faults. Since 2 rows takes 128 words (an integer occupies one word), 2 rows occupy one page like table above. In Fragment A, the inner loop will cause 32 page faults since X[2k][j] and X[2k+2][j] are in different pages. Fragment A will totally cause 32 • 64=2048 page faults. In fragment B, two inner loops use one page. There will be only 32 page faults in total.