Operating system questions with their answers (Memory management, Virtual memory, Processes synchronization) Part two

Data · June 2015

DOI: 10.13140/RG.2.1.3971.6001

1 author:

Qasim Mohammed Hussein
Tikrit University
77 PUBLICATIONS **52** CITATIONS

SEE PROFILE

Operating system questions with their answers

(Memory management, Virtual memory, Processes synchronization)

Memory management

Q1: Why does the computer must keep several processes in main memory?

Answer Q1

To improve both the utilization of the CPU and the speed of its response to users

Q2: What are the differences between?

a) Logical and physical address?
b) Page table and segment table?
c) First-fit placement and best-fit placement?
d) Contiguous and non – contiguous storage allocation
e) M***ultiple contiguous fixed partitions (MFT) and multiple contiguous variable partitions (MVT).***
f) Segmentation and paging storage?

Answer Q2

a) Logical and physical address

Logical address	Physical address
It is the address generated by the CPU	It is the address actually seen by the memory hardware

b) Page table and segment table

	Page table	Segment table
1	It contains the base address of each page in physical memory	It contains the segment address in *physical memory*.
2	The entries in page table are page number and corresponding frame number	The entries are segment base and segment limit

c) First-fit placement and best-fit placement?

	First-fit	Best-fit
1	The incoming job is placed in the first hole that is big enough.	The incoming job is placed in the small hole that is big enough.
2	It does not need search the entire hole set to place the job.	It need search the entire hole set to place the job.
3	It is faster to allocate.	It is slower.
4	More waste in storage	Less waste in storage

d) Contiguous and non – contiguous storage allocation

	Contiguous storage	Non-contiguous storage
1	A program occupies a single contiguous block of storage locations.	A program is divided into several blocks or segments that may be placed in memory in pieces not necessary adjacent to one another.
2	Faster in store	Slower in store
3	More waste in memory	Less waste in memory

e) M***ultiple contiguous fixed partitions (MFT) and multiple contiguous variable partitions (MVT).***

	Divide memory into several fixed-sized partitions. Each partition may contain exactly one process.	Memory is a collection of variable-sized segments that vary dynamically.
	More fragmentation problems	Less fragmentation problems

f) Segmentation and paging storage?

	Segmentation	Paging
	The physical memory is breaking into variable-sized blocks called segments	The physical memory is breaking into fixed-sized blocks called frames and logical memory is breaking into blocks of the same size called pages
	Address generated by CPU is divided into segment number (S) and segment offset (d).	Address generated by CPU is divided into Page number (p) and Page offset (d).
	Each segment has a name and a length (variable size)	A pages are fixed size
	Physical address = segment base + offset	Physical address = page size * frame number + offset
	May be external fragmentation	No external fragmentation

Q3: State and explain **Storage management strategies?**

Answer Q3

There are many strategies are used to obtaining the best possible use of the main storage. Some of these strategies are:

1. **Fetch strategies** are concerned with **when** to obtain the next piece of program or data for transfer to main storage from secondary storage. There are two approaches.
   A) **Demand fetch strategies**: The next piece of program or data is brought into the main memory when it is referenced by a running program.
   B) **Anticipatory fetch strategies**: They make predict and guesses and anticipating the future where program control will go next.

2. **Placement strategies:** They are concerned with determining **where** in main storage to place a new program. Strategies are most commonly used to select a free hole from the set of available holes.

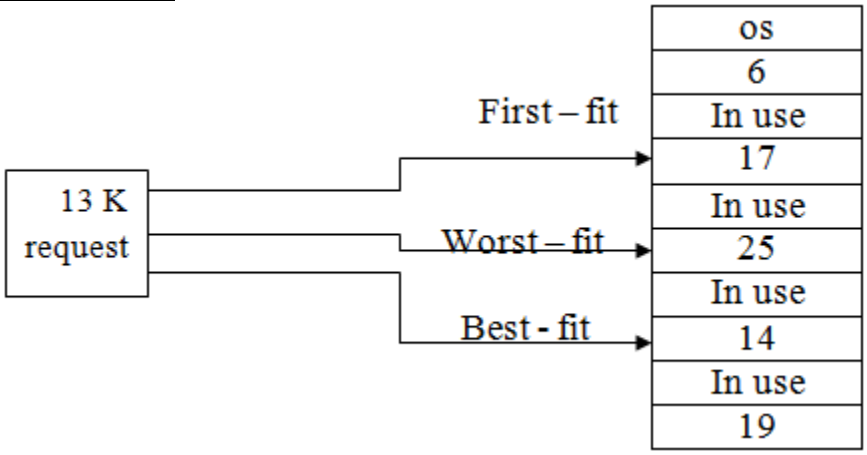
   There are three placement strategies:

   A. First fit. The incoming job places in the first hole that is big enough. It does not need search all the hole set. It is faster strategies to placement.
   B. **Best fit.** Allocate the *smallest* hole that is big enough. It needs to search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.
   C. **Worst fit**. Allocate the *largest* hole enough. Again, it must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

3. **Replacement strategies**: They are concerned with determining **which** piece of program of data to displace to make room for a new program or data.

Q4: Suppose that we have free segments with sizes: 6, 17, 25, 14, and 19. Place a program with size 13kB in the free segment using first-fit, best-fit and worst fit?

Answer Q4



Q5: What are the advantages of?

a) Overlays allocation storage.
b) Compaction.
c) Page table.
d) Segment table.

Answer Q5

(a) It allows allocating to run a program that its size larger than the amount of memory.
(b) To solve the fragmentation problem in segmentation storage.
(c) It uses to the physical memory of each page**.**
(d) It uses to the physical memory of each segment.

Q6: Consider a user program of logical address of size **6** pages and page size is **4** bytes. The physical address contains **300** frames. The user program consists of **22** instructions **a, b, c, . . . u, v .** Each instruction takes **1 byte**. Assume at that time the free frames are **7, 26, 52, 20, 55, 6, 18, 21, 70,** and **90**.

Find the following? (10 degrees)

A) Draw the logical and physical maps and page tables?
B) Allocate each page in the corresponding frame?
C) Find the physical addresses for the instructions **m, d, v, r**?
D) Calculate the fragmentation if exist?

Answer Q6

0	. a . b . c . d
1	. e . f . g . h
2	. i . j . k . l
3	. m . n . o . p
4	. q . r . s . t
5	. u . v . .

Logical map

Page number	Frame number
0	7
1	26
2	52
3	20
4	55
5	6

page table

	Contents
6	. u . v . .
7	. a . b . c . d
20	. m . n . o . p
26	. e . f . g . h
52	. i . j . k . l
55	. q . r . s . t

Physical map

The physical address = page size * frame number + offset

The physical address of m = 4*20 +0 = 80

The physical address of d = 4*7+3 = 31

The physical address of v = 4* 6 +1 = 25

The physical address of r = 4*55+ 1= 221

The external fragmentation = 0

The internal fragmentation = 2

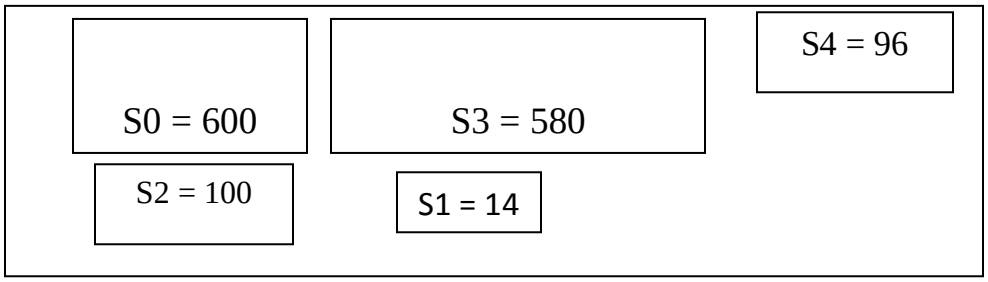
Q7: Consider a program consists of five segments: S0 = 600, S1 = 14 KB, S2= 100 KB, S3 =580 KB, and S4 = 96 KB. Assume at that time, the available free

space partitions of memory are 1200–1805, 50 – 150, 220-234, and 2500-3180. Find the following:

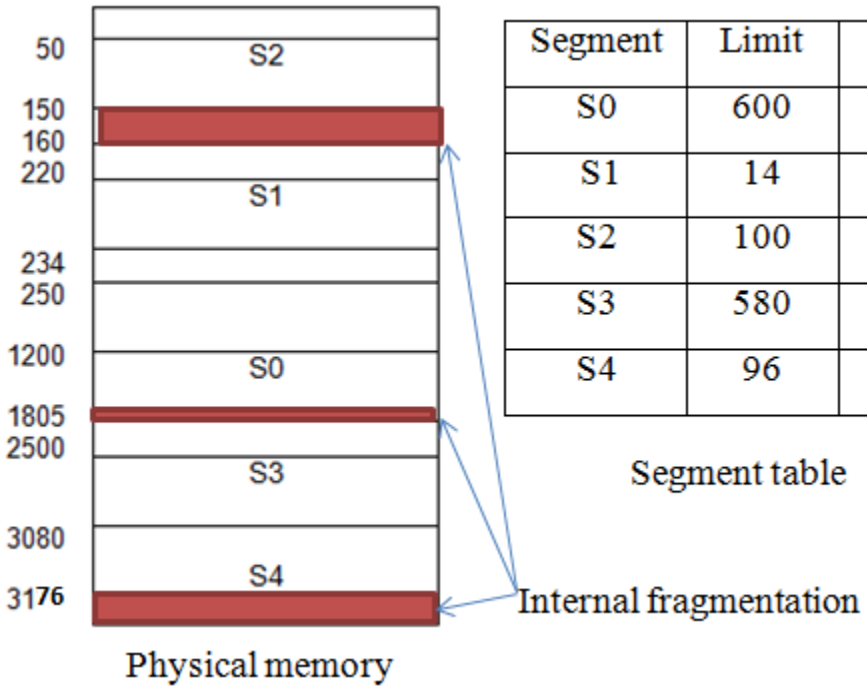
1. Draw logical to physical maps and segment table?
2. Allocate space for each segment in memory?
3. Calculate the external fragmentation and the internal fragmentation?
4. What are the addresses in physical memory for the following logical addresses: 0.580, (b) 1.17 (c) 2.66 (d) 3.82 (e) 4.20?

Answer Q7

The logical map is:



Logical map



Physical memory

Segment	Limit	Base
S0	600	1200
S1	14	220
S2	100	50
S3	580	2500
S4	96	3080

Segment table

External fragmentation =0.

Internal fragmentation = (160-150) +(1805-1800) + (3180-3176)

= 10 + 5 + 4 = 19

Fragmentation = external fragmentation + internal fragmentation = 0 + 19 = 19

4. The physical addresses are

a) 0.580➔the physical address of 0.580 = 1200+580 = 1780.

b) 1.17 ➔ because d > limit of S1, the address is **wrong**.

c) 2.66➔ the physical address of 2.66= 50 + 66 = 116

d) 3.82➔ the physical address is of 3.82 is = 2500 + 82 = 2582

e) 4.20 ➔ the physical address 4.20 = 3080+20 = 3100

virtual memory

Q8: Define the virtual memory? What are its advantages?

Answer Q8

Virtual memory is a technique that allows the execution of processes that are not completely in memory.

Advantages:

1) Enables users to run programs that are larger than actual physical memory.
2) VM makes the task of programming much easier.
3) Virtual memory allows processes to share files easily and to implement shared memory.
4) It provides an efficient mechanism for process creation.

Q9: What is the demand paging?

Answer Q

Demand paging is a method of virtual memory management that loads pages when they are demanded during program execution.

Q10: What are the advantages of using pager (i.e. demand paging)?

Answer Q10

1) Less swap time
2) Less I/O needed
3) Less amount of physical memory needed
4) More users

5) Faster response time

Q11: What are the differences between pager and swapper?

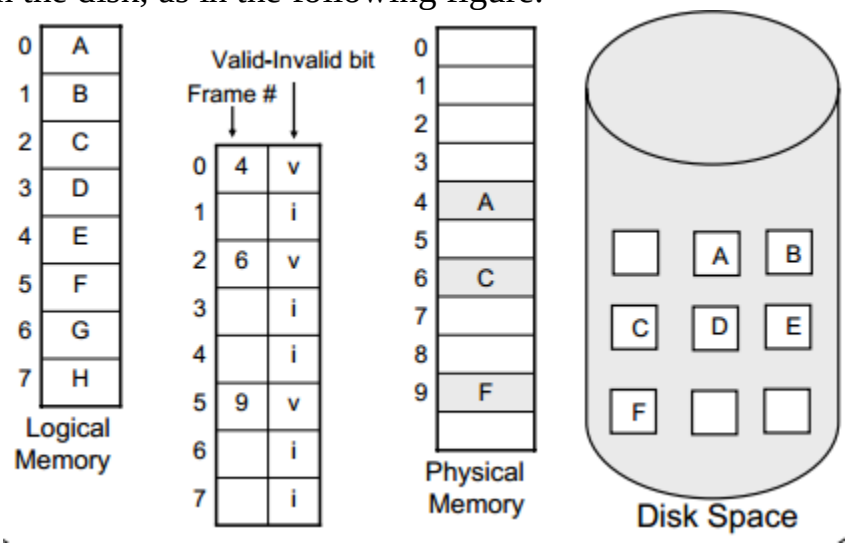
Answer Q11

	Pager	Swapper
1	Pager Swaps a page into memory when this page will be needed into memory.	swapper swaps the entire processes into memory
2	It use in demand-paging system	It uses in paging system

Q12: How can the system distinguish between the pages that are in main memory from the pages that are on the disk?

Answer Q12

The system uses valid-invalid bit is used. This bit is set to "valid" when the page in memory, while it set to "invalid" when the page either not valid or is the page is valid but is on the disk, as in the following figure.



Q13: When will the page faults occur? What is the procedure for handling the page fault?

Answer Q13

The page fault will occur when the process tries to access a page which was not into main memory.

The procedure for handling the page fault is:

1) Check an internal table for this process to determine whether the reference was a valid or an invalid memory access.
2) If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
3) Find a free frame.
4) Schedule a disk operation to read the desired page into the newly allocated frame.
5) When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6) Restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

Q14: How can measure the performance of demand paging?

Answer Q14

To measure the demand paging , the **effective access time** for a demand –paged memory is calculated by:

Effective access time = (1 – p) x ma + p x page fault time

Where **p**: The probability of page fault, $0 < p < 1$;

ma: Memory access time , ranges from 10 to 200 nanosecond.

Q15: Assume an average page-fault service time is 25 milliseconds and a memory access time is 100 nanoseconds. Find the Effective Access Time?

Answer Q15

Effective Access Time (EAT)= (1 – p) x (ma) + p x (page fault time)

= (1 – p) x 100 + p x 25,000,000

= 100 – 100 x p + 25,000,000 x p

Q16: What are the steps to modify the page-fault service routine to include page replacement?

Answer Q16

1. Find the location of the desired page on the disk.
2. Find a free frame:
   a) If there is a free frame, use it.
   b) If there are no free frames, use a page-replacement algorithm to select a **victim frame.**
   c) Write the victim frame to the disk, change the pages table.
3. Read the desired page and store it in the free frame. Adjust the page table.
4. Restart the user process.

Q17: What are the operations of page replacement algorithm?

Answer Q17

Every page replacement algorithm is operated by the following three operations:
1) Search: To search required pages from main memory.
2) Delete: To delete the evict page from main memory.
3) Insert: To insert the page into main memory.

Q18: What are the principles of the following replacement algorithms?

a) FIFO.
b) LRU.

Answer Q18

a) A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen.
b) In LRU, we can replace the page that *has not been used* for the longest period of time.

Q19: Consider the following page reference using **three** frames that are initially empty. Find the page faults using FIFO algorithm, where the page reference sequence: 7,0,1, 2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1?

Answer Q19

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	0	0	1	2	3	0	4	2	2	2	3	0	0	0	1	2	7
	0	0	1	1	2	3	0	4	2	3	3	3	0	1	1	1	2	7	0
		1	2	2	3	0	4	2	3	0	0	0	1	2	2	2	7	0	2
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

Page fault

The page fault = 15.

Q20: Consider the following page reference using three frames that are initially empty. Find the page faults using LRU algorithm, where the page reference sequence: 7,0,1, 2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1?

Answer Q20

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7

Page fault= 12

Process synchronization

Q21: Define (1) Cooperating process (2) Independent process (3) Race condition (4) Critical section (5) **Semaphore**

Answer Q21

1) **Cooperating process**: It is the process than can affect or be affected by other processes executing in the system.
2) Independent process: it is the process than cannot affect or be affected by other processes executing in the system
3) **Race condition**: When several processes access and manipulate the same data at the same time of the execution depends on the particular order in which the access takes place, is called a **race condition.**
4) **Critical section**: It is a segment of code in system process **in** which the process may be changing common variables, updating a table, writing a file, and so on.
5) **Semaphore** is a synchronization tool which can be used to deal with the critical-section problem (does not require busy waiting). A semaphore S has two standard operations: **wait( ) and signal( )**. The wait( ) operation was termed **P**, signal( ) was called **V**.

Q22: What do we need to guard against the race condition?

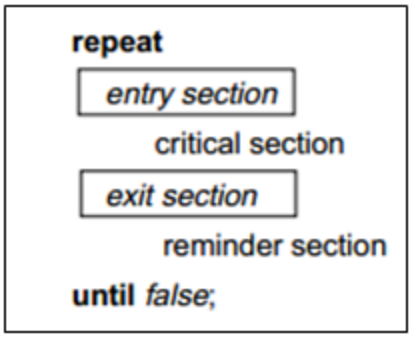
Answer Q22

To guard against the race condition, we need to ensure that only one process at a time can be manipulating the variable count. We require that the processes by synchronized in some way.

Q23: What is general structure of a typical process P?

Answer Q23

Each process must request permission to enter its critical section. The section of code implementing this request is **the entry section**. The critical section may be followed by an **exit section**. The remaining code is the **remainder section**. The general structure of a typical process *Pi* is shown in following figure



Q24: What are the requirements of solution the critical-section problem?

Answer Q24

The requirements are:

1. **Mutual exclusion**. If process P_i is executing in its critical section, then no other processes can be executing in their critical section.
2. **Progress**. If no process is executing in its critical section and some processes wish to enter their critical section, then only those processes that are not executing in their remainder sections can participate in the decision on which will enter its CS next, and this selection cannot be postponed indefinitely. (No process should have to wait forever to enter its critical section.
3. **Bounded waiting**. There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

Q25: Define the semaphore operations?

Answer Q25

The classical definitions of wait and signal are:

- The **P operation** (wait) on semaphore S, written P(S), operates as follows:
  if S > 0 then S := S – 1;
  else (wait on S)

```
wait (S)
 {
  while S<=0;   // no loop
  S--;
 }
```

- The **V operation** (signal) on semaphore S, written V(S), operates as follows:

  if (one or more processes are waiting on S)
  then (let one of these processes proceed)
  else S:= S + 1;