Operating Systems: Homework #5

Due on March 30, 2016 at 11:59pm

Professor Qu
Monday & Wednesday 3:30pm — 5:17pm

Nicholas Land

Problem 1

Why is the protection of processes' memory space important? Describe a scenario where absence of memory protection leads to problems.

Solution

Protection of processes' memory space is important because it prevents processes from accessing memory that have not yet been allocated. A scenario in which could cause problems with the absence of memeory protection would be when a process attempts to access memory that hasn't been allocated yet will cause the program to crash.

Problem 2

Consider a system where the virtual memory page size is 1KB (1024 bytes), and main memory consists of 4 page frames, which are empty initially. Now consider a process, which requires 8 pages of storage. At some point during its execution, the page table is as shown below:

Virtual Page #	Physical Page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4		No
5		No
6	0	Yes
7	1	Yes

1. List the virtual address ranges that will result in a page fault.
2. Give the following ordered references to the virtual addresses (i) 4500, (ii) 8000, (iii) 3000, (iv) 1100, please calculate the main memory (physical) addresses. If there is a page fault, please use LRU based page replacement to replace the page. How which page will be affected and compute the physical addresses after the page fault. We assume the reference string is . . . 2 4 7 3 0 4 3 0 7 5 0 7 6 0 2 3 6 4 7 6 3 2 before the new reference.

Solution

1. The virtual address ranges that will result in page fault are:
   Page 0: 0 – 1023
   Page 1: 1024 – 2047
   Page 4: 4096 – 5119
   Page 5: 5020 – 6143

2. References as follows:

(i) 4500 is a page fault. Reference string is:

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

Virtual Page #	Physical Page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4	1	Yes
5		No
6	0	Yes
7		No

*Page # * Page Size + Offset = Virtual Address*
Offset is found by $4500 - (4 * 1024) = 404$

$\therefore 4500 = 4 * 1024 + 404$
So . . . virtual page #4 points to physcial page #1, so the physical address is $1 * 1024 + 404 = 1428$.

(ii) After the page replacement, 8000 is a page fault. Reference string is:

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

Virtual Page #	Physical Page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4	1	Yes
5		No
6		No
7	0	Yes

Offset is found by $8000 - (7 * 1024) = 832$
$\therefore 8000 = 7 * 1024 + 832$
So . . . virtual page #7 points to physcial page #0, so the physical address is $0 * 1024 + 832 = 832$.

(iii) There is no page fault on 3000 so reference string & page table remain the same.

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

Virtual Page #	Physical Page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4	1	Yes
5		No
6		No
7	0	Yes

Offset is found by $3000 - (2 * 1024) = 952$

$\therefore 3000 = 2 * 1024 + 952$

So . . . virtual page #2 points to physcial page #2, so the physical address is $2*1024+952 = 3000$.

(iv) 1100 results in a page fault, reference string is:

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

Virtual Page #	Physical Page #	Valid Flag
0		No
1	3	Yes
2	2	Yes
3		No
4	1	Yes
5		No
6		No
7	0	Yes

Offset is found by $1100 - (1 * 1024) = 76$

$\therefore 1100 = 1 * 1024 + 76$

So . . . virtual page #1 points to physcial page #3, so the physical address is $3 * 1024 + 76 = 3148$.

Problem 3

Given a computer system with the following paging based addressing for virtual addresses. Please answer the following questions:

2 bits	5 bits	5 bits	5 bits	7 bits

1. What is the size of the virtual address space?
2. What is the page size?
3. What is the maximum number of pages for a process?
4. Given the system has a TLB hit ratio of 99% and page fault rate of 1%. Please formulate the effective memory access time.

Solution

1. The virtual address space is: $2 + 5 + 5 + 5 + 7 = 24 \implies 2^{24} = 16,777,216$ bytes or 16 MB.
2. The page offset field is $7 \therefore 2^7 = 128$ bytes.
3. The number of pages for each process is $2^{24} / \ 2^7 = 2^{17} \implies 131,072$
4. Effective memory access time is:
   $(\text{TLB Lookup} + \text{Memory Access Time}) * .99 + (\text{TLB Lookup} + 2(\text{Memory Access Time})) * .01$

Problem 4

Consider a system with 1MB of available memory and requests for 42KB, 396KB, 10KB, and 28KB. The system is using Buddy Allocation Algorithm.

a) Show the amount of memory allocated for each request and the state of memory after each request. Assume there is no memory release.

b) Why does internal fragmentation occur with buddy allocation? How much internal fragmentation exists in this scenario?

c) Why does external fragmentation occur with buddy allocation? How much external fragmentation exists in this scenario?

Problem 4 continued on next page...

Solution

a) 42KB $\implies$ 64KB : **64KB** 64KB 128KB 256KB 512KB
396KB $\implies$ 512KB : **64KB** 64KB 128KB 256KB **512KB**
10KB $\implies$ 16KB : **64KB 16KB** 16KB 32KB 64KB 128KB 256KB **512KB**
28KB $\implies$ 32KB : **64KB 16KB** 16KB **32KB** 64KB 128KB 256KB **512KB**

Key: **Bold** font denotes allocated memory.

b) Internal fragmentation is wasted memory visible only to the process making the memory request. Internal fragmentation occurs in buddy allocation because memory requests must be rounded up to the nearest power of 2.
Amount of internal fragmentation: $(64 - 42) + (512 - 396) + (16 - 10) + (32 - 28) = 148KB$

c) External fragmentation is wasted memory visible to the system outside of the requesting processes. External fragmentation occurs because not all requests (even after they've been rounded up to the nearest power of two) are the same size; therefore, it is possible that a memory request will not be able to satisfied, even when there is enough total free memory in the system.
Amount of external fragmentation: $16 + 128 + 256 = 400KB$