L29- TUTORIAL ON MEMORY MANAGEMENT TECHNIQUES-2

Page Replacement Algorithm

Q1: Consider the following page numbers requested by an application. The main memory has 3 frames which are initially empty. Find the number of page faults in each of the following page replacement algorithm.

(a) FIFO (b) LRU (c) LFU (d) optimal

Page reference: 4, 6,1, 4, 5, 3, 1, 4, 3, 4, 3, 6, 2, 1, 5

Page Replacement Algorithm

Page reference: 4, 6,1, 4, 5, 3, 1, 4, 3, 4, 3, 6, 2, 1, 5

FIFO page replacement algorithm

4	6	1	4	5	3	1	4	3	4	3	6	2	1	5

Page Replacement Algorithm

Page reference: 4, 6,1, 4, 5, 3, 1, 4, 3, 4, 3, 6, 2, 1, 5

LRU page replacement algorithm

4	6	1	4	5	3	1	4	3	4	3	6	2	1	5

Page Replacement Algorithm

Page reference: 4, 6,1, 4, 5, 3, 1, 4, 3, 4, 3, 6, 2, 1, 5

LFU page replacement algorithm

4	6	1	4	5	3	1	4	3	4	3	6	2	1	5

Page Replacement Algorithm

Page reference: 4, 6,1, 4, 5, 3, 1, 4, 3, 4, 3, 6, 2, 1, 5

Optimal page replacement algorithm

4	6	1	4	5	3	1	4	3	4	3	6	2	1	5

Page Tables

Q2: There are 32 page frames in the physical memory of a system. It uses a paging scheme that employs 4KB pages. The page table for a program is indexed based on virtual page number and each page table entry contains the frame number and 3 control bits. The total size of page table is 512 bytes.

- ❖ Each page table entry has -------- bits of information.
- ❖ How many bits are there in the physical address?
- ❖ How many bits are there in the virtual address?

Page Tables

Q3: Consider a system that uses 20-bit virtual address space and 16-bit physical address space. It uses a paging scheme that employs 4KB pages. The page table for a program is indexed based on virtual page number and each page table entry contains the frame number and 4 control bits.

- ❖ What is the total number of page frames?
- ❖ What is the size of the page table?

Hashed Page Table

Q4: A MMU uses hashed page table (HPT) with 4-entries. Each entry in the HPT may be either null or pointing to another physical memory address to start a chain. Each element in the chain stores the virtual page number, its corresponding physical page number and the pointer to next element (if applicable)/null. The system uses 1MB virtual address space and has a main memory of capacity 32KB. The page size is 4KB. The hash function uses i= v%4 to index into the HPT, where ‘v’ is the virtual page number and ‘i’ is the index in the HPT. Consider two active user processes P1 and P2 whose HPT entries are all initialized with null. The system uses a proportional local frame allocation algorithm in the ratio 1:2:1 for OS kernel, P1 and P2, respectively. Frames process are allotted so as to reside continuous in the main memory with lowest frame numbers to OS kernel and highest frame numbers to P2. Frames are numbered as f0, f1,….fn. The system uses LRU page replacement policy. Tie breaker in LRU is done by FIFO replacement policy. Consider the following set of requests (process ID, virtual address) generated from the CPU, given in the order of arrival. (P1, 0x22456), (P1, 0x84252), (P2, 0x28400), (P1, 0x84840), (P1, 0x47200), (P2, 0x40408), (P2, 0x6A420), (P1, 0x46250), (P1, 0x22850), (P2, 0x28A00) (P1, 0x2A2A8), (P2, 0x29600).

Hashed Page Table

HPT with 4-entries. 1MB virtual address space and 32KB main memory.
Page size is 4KB. The hash function uses i= v%4.
Proportional local frame allocation algorithm in the ratio OS:P1:P2=1:2:1
Frames are numbered as f0, f1,….fn. f0,f1→OS, f2,f3,f4,f5→P1, f6,f7→P2
LRU page replacement policy with FIFO as tie breaker.
(process ID, virtual address)→(P1, 0x22456), (P1, 0x84252), (P2, 0x28400), (P1, 0x84840), (P1, 0x47200), (P2, 0x40408), (P2, 0x6A420), (P1, 0x46250), (P1, 0x22850), (P2, 0x28A00) (P1, 0x2A2A8), (P2, 0x29600).

(a) Draw a neat labelled diagram of the HPT of P1 and P2.
(b) Find the entries in the HPT of P1 and P2 that are null.
(c) Which index in the HPT of P1 and P2 has the longest chain?
(d) Find the process ID and page numbers that are mapped to various main memory frames.

Hashed Page Table

- ❖ Virtual address space = 1 MB = 2^{20}
- ❖ Page size = frame size= 4 KB = $2^2 * 2^{10} = 2^{12}$
- ❖ So, total number of pages = 2^8 = 256
- ❖ Main memory space = 32 KB = $2^5 * 2^{10} = 2^{15}$
- ❖ (P1, 0x22456), (P1, 0x84252), (P2, 0x28400), (P1, 0x84840), (P1, 0x47200), (P2, 0x40408), (P2, 0x6A420), (P1, 0x46250), (P1, 0x22850), (P2, 0x28A00) (P1, 0x2A2A8), (P2, 0x29600).

p	d
8	12

f	d
3	12

f0,f1→OS kernel,
f2, f3, f4, f5→P1, f6,f7→P2

For Process P1, the hash table indexing will be as follows:

page no	0x22	0x84	0x84	0x47	0x46	0x22	0x2A
i= v%4	2	0	0	3	2	2	2

For Process P2, the hash table indexing will be as follows:

page no	0x28	0x40	0x6A	0x28	0x29
i= v%4	0	0	2	0	1

0	
1	
2	
3	

0	
1	
2	
3	

Hashed Page Table

For Process P1, the hash table indexing will be as follows:

page no	0x22	0x84	0x84	0x47	0x46	0x22	0x2A
i= v%4	2	0	0	3	2	2	2

For Process P2, the hash table indexing will be as follows:

page no	0x28	0x40	0x6A	0x28	0x29
i= v%4	0	0	2	0	1

HPT (P1)

0	
1	
2	
3	

HPT (P2)

0	
1	
2	
3	

Main Memory Contents

Frame #	process	page number
f0	OS	
f1	OS	
f2	P1	
f3	P1	
f4	P1	
f5	P1	
f6	P2	
f7	P2	

Hashed Page Table

Hashed page table of P1 at the end:

0	~~→[0x84, f3]~~ NULL
1	NULL
2	→[0x22, f2]→ [0x46, f5] →[0x2A, f3]
3	→[0x47, f4]

Hashed page table of P2 at the end:

0	~~→ [0x28, f6]→ [0x40,f7]~~ →[0x28, f7]
1	→ [0x29, f6]
2	~~→[0x6A, f6]~~ NULL
3	NULL

Content of the main memory at the end:

Frame #	process	page number
f0	OS	
f1	OS	
f2	P1	0x22
f3	P1	~~0x84~~ 0x2A
f4	P1	0x47
f5	P1	0x46
f6	P2	~~0x28~~ ~~0x6A~~ 0x29
f7	P2	~~0x40~~ 0x28

- ❖ Entries i=0 and i=1 are NULL in HPT(P1)
- ❖ Entries i=2 and i=3 are NULL in HPT(P2)
- ❖ Entry i=2 has the longest chain (length=3) in HPT (P1)
- ❖ Entries i=0 and i=1 have the longest chain (length=1) in HPT (P2)

Thank You