Friday, October 29, 2021 11:34 AM

2 types of memory allocation schems

1. Variable partition scheme
2. Fixed partition scheme

Variable partition = external fragmentation
Fixed partition = external + internal fragmentation

Solution to external fragmentation ----

1. Segmentation ---- reduces the external fragmentation (but it can occur at segment level)
2. Paging ---- used in fixed partition scheme
    a. RAM is divided into equal size partitions = FRAMES
    b. Process is divided into equal size partitions = PAGES = basis of division = size !!!!

F1	
F2	Page1
F3	Page2
F4	
F5	
F6	Page3
F7	
F8	Page4
F9	Page5
F10	

How to keep track ? Which page is in which frame ?

Page Table (one page table per process)

Page number	Frame number
1	2
2	3
3	6
4	8
5	9

no size column is there as frame size is fixed

Frame size = 2kb
Process P1 = total size 10
Page= 2kb

page size = frame size

Page1	0 1 2 3
Page2	0 1 2 3
PAge3	0 1 2 3
Page4	0 1 2 3
Page5	0

each frame divided into 4 parts

Pages can be stored in non consecutive frames !!! THIS **totally** solves the problem of External Fragmentation

Paging still has the internal fragmentation (the last frame may not be completely used) !!! NO SOLUTION

Actual address calculation ----

The address of next instruction PC register has two parts Page Number , Page offset

next instruction is on page 3 offset 4

kernel finds the frame number of page 3 from the page table.
Kernel gets the base address of frame + page_offset = actual physical location of the instruction.

____

PAGED SEGMENTATION
RAM is divided into frames
Process is first divided into SEGMENTS
EACH segment is divided into PAGES

EVERY PROCESS HAS A SEGMENT TABLE	EVERY SEGMENT WILL HAVE A PAGE TABLE

This overcomes --- external fragmentation in Segmentation
This overcomes ----large page table problems in Paging !!!

____

ENTIRE PROCESS IS LOADED IN RAM + It should be in contiguous memory locations!!!!
To overcome external fragmentation
|
|
Entire process is loaded in RAM + process can be stored in **non** consecutive memory locations
If process is LARGER than RAM size then it cannot be RUN !!!
|
|
|
DON'T load the entire process in RAM + load process in non consecutive memory locations

Let us say 10 percent of the process pages are loaded in the RAM
Process has 100 pages
Load 10 pages in the FRAMES in RAM
WHERE ARE remaining 90 pages ?????? HARD DISK (the area where the remaining pages of running process are stored ------- SWAP SPACE / VIRTUAL MEMORY)

What are the advantages of not loading all pages in the RAM ??

1. a larger program that is larger than the RAM can be executed
2. The part of code that is never used is never loaded (loader effort is reduced)
3. More processes can be started at a time (degree of multiprogramming is high)

---

10 pages are in RAM ------- how many entries in the page table

Page number	Frame number
10 entries	

NEXT address to be executed is on page number 35 (and pages 1 to 10 are in RAM)

35 kidhar ?? Swap space

PAGE FAULT SOFTWARE INTERRUPT OCCURS !!!!

Page 35 is loaded in the RAM on DEMAND   (DEMAND PAGING)

Process resumes execution!!!

Virtual memory management !!!!

Page fault leads to loading the demanded page !!!!

WHERE is the new page loaded ?   In some free frame !!!!
If no free frame is present then ?????? PAGE REPLACEMENT !!!

Which page is replaced ??

1. Page of any process = global page replacement policy = this may make other processes to fault---
   Many processes may start faulting ---- this may lead to system hang as many processes are waiting due to page fault }}}}   THRASHING PROBLEM

2. Page of the faulting process = local page replacement policy
   a. TO avoid THRASHING PROBLEM ---- use local page replacement !!!!!

Which page of the faulting process should be replaced ???

Page replacement algorithms ---------

OPTIMUM POLICY ---- that page should be replaced that is not needed in the future !!!!

LRU = Least Recently Used = the page that was not used for a long time = it will be not be needed in future !!!!
MRU = Most Recently Used = the page that was used just now can be replaced so that it is not needed in future.
FIFO = the page that arrived first must be removed first

Page ACCESS STRING ----------

1,2,3,4,5,1,2,1,3,1,4,1

Process has 4 frames allocated to it

F1	
F2	
F3	
F4	

Find the number of page faults and page replacements using LRU

F1	1 LRU	5	5	5	5 LRU	4 sec

F2	2	2 LRU	1	1 recent	1 recent	1 recent
F3	3	3	3 LRU	2 second recent	2 third recent	2 LRU
F4	4	4	4	4 LRU	3 secondrecent	3 third

6=6 page replacements
6+4 = 10 page faults

Find the number of page faults and page replacements using FIFO

F1	1 first in	5	5	5	5 firstin	4
F2	2	2 firstin	1	1	1	1 firstin
F3	3	3	3 firstin	2	2	2
F4	4	4	4	4 firstin	3	3

5=5 page replacements
4 + 5 = 9 page faults

____

Page access string =1,3,2,3,1,4,1,5,6

MRU

F1	1		MRU	4 MRU	1 MRU	5 MRU	6 MRU
F2	3	MRU		3	3	3	3
F3	2 MRU			2	2	2	2

3+4 = 7 page faults
4 page replacements

____

OS -------- 3 questions --- any 2 to solve
2 shell script questions, 1 cprog

____