Monday, April 4, 2022 5:48 PM

Pure Segmentation , Pure Paging

Combination of Paging and Segmentation = **Paged Segmentation** (real life)

Process ---- divide it in segments
Each segment ----divide it in pages
For Each segment there will be a page table

RAM is also divided into FRAMES --- pages are stored in FRAMES

ADV ---
Instead of ONE LARGE PAGE TABLE in Pure paging ----- small page tables per segment
External Fragmentation is REMOVED

---

variable partition fixed partition	entire process is loaded in consecutive memory locations
Paging Segmentation	Entire process is loaded in non consecutive memory locations
Demand Paging	Partial process is loaded in non consecutive memory locations

For example my process has 100 pages .
Partial process (10 percent pages) is loaded in memory ---- 10 pages are loaded in memory

Remaining 90 pages are in a special region of the HARD DISC ----------**SWAP SPACE**

Swap SPACE is acting as if it is the extended memory ---------- **VIRTUAL MEMORY** !!

The page table of the process will have how many entries ?
only 10 pages are present in the page table ---- only those that get a FRAME

page number	Frame number

Page 0
main ()
{
...
....
....
....
f1() ---------- Page 43(is in the swap space)
.....
....
}

Kernel will try to find the page in the page table ---- but it isn't there .
Kernel raises an interrupt ----- PAGE FAULT INTERRUPT (page nahi mil raha RAM me)

Page Fault Interrupt Handler starts running ---
Process is moved to waiting state
The handler will load the DEMANDED page in a **free** frame
After this is done , update the page table
Move the process from waiting to ready

MECHANISM to load the page on demand (by page fault interrupt) ========= DEMAND PAGING !!!!

What is the advantage of Demand Paging?

1. External fragmentation is removed
2. More processes can be started by LTS , as entire process is not loaded in the RAM , RAM space is saved.
   Degree of Multiprogramming is HIGH --- at a time more processes can be in the RAM
3. programs that are larger than the RAM can be executed
4. loader effort is reduced … many parts of the process are never needed , so they are never loaded

---

If a page is demanded , but **NO free frame** is available in the RAM ???????????

Kernel will **Replace a page in a frame** with the DEMANDED Page ---------- PAGE REPLACEMENT

PAGE REPLACEMENT POLICY ---- GLOBAL PAGE REPLACEMENT or LOCAL PAGE REPLACEMENT ???

Global Page Replacement	If process A is **faulting - it is OK to replace a page of process B or C or D** **This may make other processes as faulting--- if a chain reaction happens --- many process on the system may start faulting!!!** This leads to a problem called as **THRASHING** ---- system appears to have hanged--- many processes are faulting
Local Page Replacement	If process A is faulting - then another page of process A only is replaced - solution to THRASHING

Local Page Replacement ---- Which page of the process should be replaced .

The page that will not be needed in future should by replaced ---OPTIMUM decision

Page Replacement algorithms ----------

1. FIFO = the page loaded first is replaced first .(Oldest arrival page is replaced first)
2. MRU = **most recently used** page is replaced first (access time is latest is replaced first)
3. LRU = **least recently used** page is replaced first (access time oldest is replaced first)

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

Number of frames for the process ---- 3

F1	1 arr1
F2	2 arr2
F3	3 arr3

->

F1	4 arr4
F2	2 arr2
F3	3 arr3

F1	4 arr4
F2	1 arr5
F3	3 arr3

F1	4 arr4
F2	1 arr5
F3	2 arr6

F1	5 arr7
F2	1 arr5
F3	2 arr6

Using FIFO find the number of page replacements ? 4 , page faults ---4 +3 =7

Using MRU find the number of page replacements ? 2

F1	1 (3rd mru)
F2	2 (2nd mru)
F3	3 (mru)

F1	1 (3rd mru) (3rd mru) , (mru) (2nd mru) , (mru) , (2nd mru)
F2	2 (2nd mru) , (mru) ,(2nd mru) , (mru) ,(2nd mru),(3rd mru)
F3	4 (mru) (2nd mru) ,(3rd mru) , 3mru ,3rd mru , (mru)

F1	1 (2nd mru) , 3rd mru
F2	2 (3rd mru) , mru
F3	5 (mru) , 2nd mru

Using LRU find the number of page replacements ? 4

F1	1 (lru)	4 (3) (2)	4 (LRU) ,(3)	4 (2)	4 (LRU)				
F2	2 (2)	2 LRU ,(3)	2 (2) ,(3) ,(2),LRU	5 (3)	5 (2)				
F3	3 (3)	3 (2) ,LRU	1 (3), (2) ,(3) ,(2)	1 LRU	2 (3)				

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

Chmod command ---- used for allocating permissions to files and folders

How to see the permissions of a file ?
ls -l

There are 3 permissions - r , w, x

r	Read
w	Write
x	execute

Users are allocated to groups

Group	Users
Faculty	prachi , shrinivas, janhavi,sneha
Admin	Manoj
guest	xyz
Student	u1, u2,u3,……

I have logged in as u3

If I create a file ---------file1	Owner =u3
	Group = u1 ,u2
	Other = prachi, shrinivas, janhavi,sneha,manoj, xyz

Rwx rwx r-x

User=owner	rwx
Group	rwx
other	r-x

If a folder has r permission ---- we can ls the folder
If a folder has w permission --- we can vi , mkdir, rm , rmdir, cp , mv
If a folder has x permission --- we can cd to the folder

Rwx rwx r-x	remove x permission for OTHER	chmod o-x lambda
rwxrwxr--	add w permission for OTHER	chmod o+w lambda
rwxrwxrw-	remove w permission from all 3	chmod ugo-w lambda
r-xr-xr--	remove x from u and g add w to u and g	chmod ug -x lambda chmod ug+w lambda
		OR
		chmod ug = rw lambda
rw-rw-r--	All the 3 should have all 3 permissions	chmod ugo=rwx lambda
rwxrwxrwx	give rwx to u, rw- to g and r-- to o	chmod 764 lambda
rwxrw-r--	give only w permission to all 3	chmod 222 lambda

OCTAL way

- - -

000	0
001	1

010	2
011	3
100	4
101	5
110	6
111	7

---

date , cal , touch (create a 0kb file)

cp ./one/alpha ./two	cp source destination
cp -r ./one ./two	To copy folder use -r
mv ./beta ./one	Moves from source to destination
mv ./lambda ./two	Same mv without -r for moving folders also
mv ./two/alpha ./two/theta	Move can be used to RENAME a file