Στυλιανός Ψαρά sdi1800226

main_fun.c

char* GET_PAGE_NUM(char str)**

Παίρνω σαν όρισμα την λογική διεύθυνση την μεταφέρω σε πίνακα και παίρνω τους πρώτους 5 χαρακτήρες που αποτελούν τον αριθμό τις σελίδας και τον επιστρέφω.

char* GET_OFFSET(char str)**

Παίρνω σαν όρισμα την λογική διεύθυνση την μεταφέρω σε πίνακα και παίρνω τους τελευταίους 3 χαρακτήρες που αποτελούν offset και το επιστρέφω.

HASHED PAGE TABLES

int search_insert(Node* head, char page_num, char** of,char** r_w, int max_Frames)**

Αρχικά τρέχω μέσα στην λίστα και ψάχνω τον αριθμώ τις σελίδας και αν το βρω αλλάζω το R/W επίσης ελέγχω αν είναι w κάνω το dirty bit 1 και επιστρέφω 1 που δηλώνει ότι υπάρχει ήδη η σελίδα στην μνήμη . Στην περίπτωση που δεν υπάρχει η σελίδα στην μνήμη άλλα υπάρχει χώρος προσθέτω νέο κόμβο στην λίστα του hashed table με τα στοιχειά του trace και ελέγχω αν είναι w κάνω το dirty bit 1 αλλιώς 0 και επιστρέφω 2 που δηλώνει ότι η σελίδα δεν υπάρχει στην μήνη άλλα υπάρχει χορός για εισαγωγή . Τέλος στην περίπτωση που η σελίδα δεν υπάρχει στην μνήμη και δεν υπάρχει χώρος επιστρέφω 3 .
(Ο έλεγχος αν τα frames στην μνήμη έχουν γεμίσει γίνετε μεσώ μιας στατικής μεταβλητής counter η οποία αυξάνετε κάθε φορά που γίνετε εισαγωγή (2) και μειώνεται κάθε φορά που θα χρειαστεί διαγραφή (3))

int search_delete(Node* head, char victim_num, int* read_counter, int* write_counter)**

Αρχικά τρέχω στην λίστα και ψάχνω το victim και μόλις το βρω αυξάνω τον μετρητή των reads και αν πρέπει να γραφτεί στον δίσκο δηλαδή το dirty bit είναι 1 αυξάνω και τον μετρητή των writes και μετά διαγράφω το victim από την λίστα.

LRU

int LRU_Victim(int time[], int frame_num)

Η συνάρτηση αυτή παίρνει σαν όρισμα ένα πίνακα που περιέχει το χρόνο άφιξης κάθε σελίδας μέσα στην μνήμη και ψάχνει μέσα στον πίνακα μέχρι να βρει την παλαιότερη σελίδα δηλαδή την σελίδα με τον μικρότερο χρόνο άφιξης και επιστρέφει την θέση του victim στον πίνακα.

SECOND CHANCE

int push(SC_Node first,SC_Node** last ,char** p_num,int process)**

Ελέγχω αν η κυκλική ουρά είναι αδεία και βάζω τα στοιχειά της πρώτης σελίδας στον πρώτο κόμβο αλλιώς δημιουργώ καινούργιο κόμβο και τον προσθέτω στο τέλος της ουράς.

int delete_victim(SC_Node first, SC_Node** last, char** victim_page_num,int victim_process)**

Ψάχνω μέσα στην ουρά να βρω το σωστό victim μόλις εντοπίσω τον κόμβο τον διαγράφω και θέτω σαν last στην ουρά τον κόμβου που βρισκόταν μπροστά του και σαν first τον κόμβο που βρισκόταν πίσω του.

char* SECOND_CHANCE(SC_Node first,SC_Node** last,int* victim_process)**

Τρέχω μέσα στην ουρά από τον κόμβο που είναι first και πίσω μέχρι να βρω τον πρώτο κόμβο που έχει bit=0 του οποίου επιστρέφω τον αριθμό σελίδας και βάζω σε μια μεταβλητή victim_process τον αριθμό της διεργασίας . Στην περίπτωση που ο τρέχων κόμβος έχει bit=1 το μηδενίζω και συνεχίζω στον προηγούμενο.

MAIN

Αρχικά στην main γίνονται οι κατάλληλες αρχικοποιήσεις και ανοίγω τα 2 αρχεία bzip και gcc. Στην συνέχεια μέσα σε ένα μεγάλο loop διαβάζω γραμμή γραμμή από τις διεργασίες και έχω έναν έλεγχο που αλλάζει διεργασία αν πείρα q traces. Έπειτα χωρίζω την λογική διεύθυνση και κάνω insert στο κατάλληλο page table. Στην συνεχεία χωρίζεται σε 2 περιπτώσεις lru και second chance.

LRU
1)Αν υπάρχει η σελίδα ήδη στην μνήμη:Αυξάνει τον χρόνο άφιξης της συγκεκριμένης σελίδας

2)Αν δεν υπάρχει άλλα έχει χώρο: Κάνει εισαγωγή στους πίνακες των frames την νέα σελίδα

3) Αν δεν υπάρχει άλλα δεν έχει χώρο: Βρίσκει το victim με lru το διαγράφει από το hashed page table και κάνει εισαγωγή το νέο αριθμό σελίδας στην θέση του θύματος και στο hashed page table

SECOND CHANCE
1)Αν υπάρχει η σελίδα ήδη στην μνήμη:Βρίσκει την σελίδα στην ουρά και θέτει το bit=1

2)Αν δεν υπάρχει άλλα έχει χώρο: Κάνει εισαγωγή στην ουρά την νέα σελίδα

3) Αν δεν υπάρχει άλλα δεν έχει χώρο: Βρίσκει το victim με sc το διαγράφει από το hashed page table και κάνει εισαγωγή το νέο αριθμό σελίδας στην ουρά και στο hashed page table

compile

gcc main.c main_fun.c -o main

./main lru 20000 100 250

(lru/sc, max αριθμός traces που θα διαβάσει και από τα 2 process, αριθμός frames, q)