1115201700091 (-1)

Τροπος Εκτελεσης:

αν είναι lru 0

./psim <αλγ/μος> <αριθ.π/αισίων> <q>< ws-window> <max-references>.

πχ. ./psim lru 10 5 0 50

Frames: 10
Page requests: 50
Page faults: 26
Diskwrites: 3.

./psim lru 30 12 0 200
Frames: 30
Pagerequests: 200
Pagefaults: 82
Diskwrites: 16

./psim lru 10 15 0 200
Frames: 10
Pagerequest: 200.
Page faults: 103
Diskwrites: 24.

./psim ws 25 20 10 300
Frames: 25
Pagerequests: 300.
Pagefaults: 141.
Diskwrites: 47.

./psim ws 100 25 15 500
Frames: 100 Pagerequests 500 Pagefaults: 102
Diskwrites: 18.

Αν η σελίδα δεν είναι στην μνήμη. Κατορθώς έχουμε Pg. fault.
Αν έχουμε διαθέσιμο frame απλά την βάζουμε στον IPT (και συνεπώς στην μνήμη).
Αν δεν έχουμε διαθέσιμο frame, προφανώς πρέπει να δημιουργήσουμε ένα με την πολιτική του LRU προφανώς αφαιρούμε από τη IPT την σελίδα που χρησιμοποιείται πιο παλιά και ελευθερώνουμε το frame της
-> GetVictim
-> Getframe
-> Fillframe // with the new Page.
αν ο indicator της ήταν W τότε diskwrites++;

nuweworkingSet:
Παίρνουμε ένα Page αν αυτό το Page ανήκει σε διαφορετική διεργασία από αυτήν που μέχρι τώρα θα πρέπει να ελέγξουμε και να επαναφέρουμε το workingset της στη main Memory (SetBacktoMem)
-> κάνουμε update το WS.
Αν το Page είναι στο MainMemory απλά κάνουμε ένα Update στα στοιχεία του. (w.l) αλλά και στη θέση του στο recentList-workingSet
Αν το Page δεν είναι στο Main Memory (Pgfaults)
-> Αν δεν είναι: Βρίσκουμε και ελευθερώνουμε ένα frame στο οποίο βρίσκεται σελίδα που δεν ανήκει στο workingSet.
Τέλος τοποθετούμε την νέα σελίδα σε αυτό το frame.

InvPgT.h + InvPg.cpp

→ Η δομη PgEntry αναπαριστά τα addresses (pid, PageNum, dirty(R,W))

→ Η δομη InvPgT: το Inverted Pagetable.

getFrame: given a page returns the frame in which this page is
getfreeframe: returns a freeframe.
FillFrame: Puts a Page in a frame.

WS.h, WS.cpp:

→ Η δομη working set:
Διατηρεί δύο λίστες από διευθύνσεις:
recentList: διατηρεί διευθύνσεις όπου όσο πιο κοντά στην λίστα είναι μια διεύθυνση τόσο πιο πρόσφατα χρησιμοποιήθηκε.
workingset: διατηρεί τις διευθύνσεις που ανήκουν στο working set.
→ Η δομη WS_Handler διαχειρίζεται τα workingSet διαχειρίζεται τα workingSets καθώς έχουμε περισσότερες από μια διεργασίες.

(Working set) : Usupdate: αφαίρεσε την least-recently σελίδα από το working set αν δεν έχει ξαναχρησιμοποιηθεί πιο πρόσφατα φυσικά
→ Αν δεν υπάρχει χώρος,
Αν υπάρχει χώρος και η σελίδα δεν υπάρχει στο working set → Βάζει.

WS_Handler: Give_Pr_TakeSet
↳ Given a process id returns its working set.

LRU.h + LRU.cpp.

Η δομή LRU διατηρεί μια λίστα που αντιστοιχεί αναπαριστά την "χρονική ροή" με την οποία ζητήθηκαν οι σελίδες

□ □ □ - - - □

most recent request <——> least recent request.

→ GetVictim: Προφανώς "θύμα" είναι αυτή στο τέλος της λίστας όπως ορίζεται για LRU.

→ MakeMRU: Make it most recently Used
αναφορά σε σελίδα της λίστας άρα πρέπει να πάει στην αρχή της.

MemSim.h + MemSim.h.

MemSim: Η δομή υπεύθυνη για το Simulation της Μνήμης → έχει τις μεταβλητές αποτελεσμάτων,
→ τον IPT.
→ αλλά και τις δομές που χρειάζονται για τους αλγορίθμους LRU και WS.

→ Παίρνουμε τις αναφορές getRef καλώντας όποτε χρειάζεται μία εναλλάξ μεταξύ των αρχείων και συνεπώς των ροών αναφοράς και τρέχουμε είτε τον LRU είτε τον WS.

LRU algo: → Αν η σελίδα υπάρχει ήδη στην μνήμη απλά ανανέωσε τα δεδομένα
→ δηλαδή θεώρησέ την πιο πρόσφατη και άλλαξε τον indicator (R/W) αν είναι διαφορετικός

→