Λειτουργικά Συστήματα
Εργασία 2η

Ονοματεπώνυμο: Γεωργία Κουτίβα
Αριθμός Μητρώου: 1115201700060

Γενικά:

Το πρόγραμμα προσομοιώνει την διαχείριση μνήμης με βάση τους αλγορίθμους LRU και 2nd Chance. Κοινά στοιχεία και στους δύο αλγορίθμους είναι τα εξής:

- Τα page tables για τις διεργασίες bzip και gcc. Τα page tables είναι απλά hash tables, ένα για κάθε διεργασία, με την πολύ απλή συνάρτηση f(x) = x % number_of_table's_buckets, όπου x είναι ο αριθμός της σελίδας που θέλουμε να προσθέσουμε στο table. Η επιλογή του αριθμού των buckets έγινε τυχαία.
- Τα περιεχόμενα των κόμβων των overflow lists των hash buckets, τα οποία είναι μόνο ο αριθμός της σελίδας της εκάστοτε διεργασίας και ένα dirty bit, που γίνεται set σε ένα αν διαβάσουμε οποιαδήποτε αναφορά στην σελίδα που να είναι τύπου W.
- Η διαδικασία εναλλάξ ανάγνωσης από τα δύο αρχεία:
  - Αν θέσουμε αριθμό max references, το άθροισμα των αναφορών από τα δύο αρχεία θα είναι ίσο με τον αριθμό των max references, ακόμα και αν ο αριθμός των αναφορών που διαβάζουμε κάθε φορά δεν διαιρεί τέλεια τον αριθμό των max references και επομένως διαβάζουμε λιγότερες αναφορές από το δεύτερο αρχείο.
    - Συγκεκριμένα, αν έχουμε q=3 και max references = 10, θα διαβάσουμε 3 αναφορές από το πρώτο αρχείο, 3 από το δεύτερο, 3 από το πρώτο και μία από το δεύτερο.
  - Πάντοτε διαβάζουμε πρώτα από το αρχείο bzip, και μετά από το αρχείο gcc. Το πρόγραμμά θεωρεί ότι τα αρχεία βρίσκονται στον ίδιο φάκελο με τα source files του προγράμματος.
- Η διαδικασία ανάγνωσης από το κάθε αρχείο:
  - Πρώτα διαβάζουμε την διεύθυνση της αναφοράς και αν αυτή είναι R ή W. Η διεύθυνση αποθηκεύεται ως unsigned int. Έπειτα κάνουμε clip τα 12 χαμηλότερα bits (AND mask και 12 bits right shift). Έχουμε το page number έτοιμο.
  - Κατ'όπιν, πάμε να εισάγουμε το page number στο hash table του αρχείου από το οποίο διαβάζουμε.
    - Αν κατά την αναζήτηση στην overflow list του αντίστοιχου bucket διαπιστώσουμε ότι η σελίδα ήδη υπάρχει, τότε ενημερώνουμε την δομή που έχουμε για την προσομοίωση του αλγορίθμου (η οποία κρατά σελίδες **και από τα 2** processes), και αυτή προβαίνει στην κατάλληλη ενέργεια ανάλογα με τον αλγόριθμο που εκτελούμε.
    - Αν δεν βρεθεί η σελίδα ήδη στο hash table, πάμε να την εισάγουμε στην δομή για την προσομοίωση του αλγορίθμου. Αν χρειαστεί, η δομή θα θυματοποιήσει μία ήδη υπάρχουσα σελίδα, και θα μας επιστρέψει τόσο τον αριθμό σελίδας, όσο και την διεργασία στην οποία ανήκει η σελίδα. Αφαιρούμε τον κόμβο από το αντίστοιχο hash table (όχι κατ'ανάγκην αυτό από το οποίο διαβάζουμε), και αυξάνουμε τον μετρητή των page faults.
      - Τόσο στην δομή για την προσομοίωση του hash table όσο και στην δομή για την προσομοίωση του το περιεχόμενο των κόμβων της δομής είναι κοινό: ο αριθμός της σελίδας που βρίσκεται (υποθετικά) στην μνήμη και η διεργασία από την οποία προήλθε, ώστε να ξέρουμε σε ποιο hash table αυτή κρατείται.
      - Κατόπιν της αφαίρεσης από το hash table, εξετάζουμε αν η σελίδα ήταν dirty. Αν ναι, αυξάνουμε τον μετρητή των writebacks.
    - Αν η αναφορά που μόλις διαβάσαμε ήταν τύπου w, φροντίζουμε να κάνουμε set το dirty bit στον κόμβο της σελίδας στο hash table.

    - Όταν διαβάσουμε όλες τις αναφορές που έχουμε ορίσει ότι θα διαβάσουμε, καταστρέφουμε τα hash tables, μετράμε πόσες σελίδες είχαν μείνει στην μνήμη μετά την ‘λήξη’ των διεργασιών και προσθέτουμε στον μετρητή των writebacks όσους κόμβους του table βρήκαμε να είχαν dirty bit = 1.
- Τέλος, εκτυπώνουμε τον αριθμό των αναγνώσεων από τον δίσκο/page faults και των εγγραφών στον δίσκο/writebacks, καθώς και πόσες εγγραφές διαβάσαμε από το κάθε αρχείο και συνολικά.

Επιμέρους για τον κάθε αλγόριθμο:

- LRU:
    - Η δομή που χρησιμοποιείται στην προσομοίωση του αλγορίθμου είναι μία ουρά. Το μέγιστο μήκος της ουράς είναι ίσο με τον αριθμό των frames που λέμε στον προσομοιωτή ότι θα υπάρχουν διαθέσιμα στην ‘μνήμη’.
        - Όταν το πλήθος των στοιχείων στην ουρά ξεπεράσει τον αριθμό αυτό κατά 1, αφαιρούμε το στοιχείο στο οποίο δείχνει ο δείκτης bottom της ουράς (το πρώτο πρώτο στοιχείο το οποίο μπήκε στην ουρά από αυτά που υπάρχουν σε αυτήν) και επιστρέφουμε τον αριθμό σελίδας και την διεργασία της σελίδας αυτής.
        - Αν ένας κόμβος υπάρχει ήδη σε ένα από τα page tables (άρα και στην ουρά), τότε τον μετακινούμε να είναι στην αρχή της ουράς, θέτοντας τον δείκτη above της ουράς να δείχνει σε αυτόν και κάνοντας τις κατάλληλες συνδέσεις.
- Second Chance:
    - Η δομή που χρησιμοποιείται στην προσομοίωση του αλγορίθμου αυτού είναι επίσης μία ουρά, με διαφορετικές όμως λειτουργίες από την πρώτη. Το μέγιστο μήκος της ουράς είναι και πάλι ίσο με τον αριθμό των frames που λέμε στον προσομοιωτή ότι θα υπάρχουν διαθέσιμα στην ‘μνήμη’.
        - Όταν πάμε να εισάγουμε μία σελίδα στην ουρά και παρατηρούμε ότι, πριν την εισαγωγή, το πλήθος των στοιχείων της είναι ίσο με τον αριθμό των frames, τότε πρώτα καλείται η συνάρτηση θυματοποίησης και μετά γίνεται η εισαγωγή του νέου κόμβου στην ουρά.
        - Οι κόμβοι, πέρα από τον αριθμό της σελίδας και την διεργασία στην οποία αυτή ανήκει, έχουν και ένα recently used bit. Κάθε σελίδα που εισάγεται στην ουρά μπαίνει με αυτό το bit ίσο με μηδέν.
        - Αν μία σελίδα ήδη υπάρχει στο αντίστοιχο hash table, τότε στην ουρά για τον αλγόριθμο δεύτερης ευκαιρίας απλά θέτουμε το recently used bit ίσο με 1, χωρίς να πειράζουμε την σειρά των κόμβων στην ουρά.

Οδηγίες Μεταγλώττισης και Εκτέλεσης:

- Τα αρχεία bzip και gcc θα πρέπει να βρίσκονται στον ίδιο φάκελο με τα source files του προγράμματος.
- Για μεταγλώττιση: make
- Για καθαρισμό των αρχείων: make clean
- Για εκτέλεση:
    - ./main αναγνωριστικό αλγορίθμου frame_number q (max_references)
    - Το αναγνωριστικό μπορεί να είναι είτε
        - LRU είτε
        - 2ND
    - Παράδειγμα: Για 10 frames, q=100 και max_references = 10.000 τρέξτε
        - Για LRU: ./main LRU 10 100 10000
        - Για second chance: ./main 2ND 10 100 10000
    - Το τελευταίο όρισμα είναι προαιρετικό και μπορείτε να το παραλείψετε.