**3η εργαστηριακή: Μηχανισμοί εικονικής μνήμης**

Ειρήνη Στρουμπάκου 03121183

Χαράλαμπος Παπαδόπουλος 03120199

**1.1)**

3) Επιχειρούμε να τυπώσουμε την φυσική διεύθυνση του buffer και λαμβάνουμε το εξής μήνυμα:



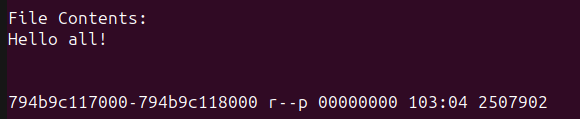
το οποίο είναι αναμενόμενο καθώς ο buffer δεν έχει αρχικοποιηθεί, άρα δεν καταλαμβάνει χώρο στην φυσική μνήμη, παρά μόνο στην εικονική.

4)



Αφού αρχικοποιήσαμε τον buffer καταλαμβάνει πλέον χώρο στην μνήμη.

5)



Η απεικόνιση αυτή του αρχείου μας δείχνει:

794b9c117000-794b9c118000: τις θέσεις που καταλαμβάνει στην εικονική μνήμη

r--p: τα δικαιώματα του αρχείου, συγκεκριμένα ότι ειναι read-only και ότι είναι private

0: offset, δηλαδή ξεκινάμε από την αρχή του αρχείου

103:04: major:minor devide numbers

2507902: αναγνωριστικό του αρχείου

6)



Παρατηρούμε επίσης ότι τυπώνει /dev/zero (deleted), το οποίο είναι ένα ειδικό αρχείο των unix συστημάτων για αρχικοποίηση του χώρου μνήμης που δεσμεύεται με μηδενικά. Ύστερα το αρχείο αυτο διαγράφεται.

7)

Παρατηρούμε ότι οι χάρτες είναι πανομοιότυποι, το οποίο είναι και αναμενόμενο αφού δεν έχουμε κάνει καμία τροποποίηση στα δεδομένα.

8)



Αρχικά χρησιμοποιούν τον ίδιο private buffer.

9)



Αφού γράψουμε στον buffer του παιδιού αλλάζει η θέση μνήμης του, το οποίο είναι αναμενόμενο αφού η διαχείριση μνήμης υλοποιείται με λογική copy-on-write, έτσι δεν μπορεί μία διεργασία να επηρεάζει τα δεδομένα της άλλης.

10)

  
Για λόγους έμφασης επιλέξαμε να αρχικοποιήσουμε τον shared buffer ώστε να εμφανίζεται η θέση μνήμης και στις δύο διεργασίες. Παρατηρούμε, πως σε αντίθεση με πριν, οι τροποποιήσεις των δεδομένων γίνονται στις ίδιες θέσεις μνήμης, το οποίο σημαίνει ότι οι διεργασίες επεξεργάζονται τα ίδια δεδομένα.

11)



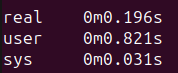
Τυπώνοντας τον χάρτη μνήμης επιβεβαιώνουμε ότι τα δικαιώματα του παιδιού άλλαξαν από rw, μόνο σε r. Το s συμβολίζει το shared buffer.

Στο 1ο μέρος της άσκησης δεν αντιμετωπίσαμε κάποιο πρόβλημα, καθώς κάθε μεμονωμένο ερώτημα ήταν πολύ καλά διατυπωμένο και ήθελε συγκεκριμένα πράγματα.

**1.2 )**

1.2.1)

Χρονομετρούμε την υλοποίηση με διεργασίες και σημαφόρους



Παρατηρείται σημαντική βελτίωση στην επίδοση, με την βέλτιστη να επιτυγχάνεται για 6 διεργασίες.

Το αποτέλεσμα αυτό δεν ήταν το αναμενόμενο, καθώς συνήθως τα threads είναι πιο “ελαφριά” και γρήγορα. Αυτό μπορεί να οφείλεται τόσο στις λεπτομέρειες της υλοποίησης του λογισμικού μας συστήματος, η οποία πιθανώς να ευνοεί τα processes, καθώς και σε πιθανώς μη βέλτιστο συγχρονισμό στην υλοποίηση με τα threads.

1.2.2)

Στο συγκεκριμένο ερώτημα χρησιμοποιήσαμε έναν κοινό buffer όπου κάθε διεργασία αφού έκανε το compute πήγαινε και έγραφε στην αντίστοιχη σειρά του κοινού buffer, ύστερα το output γινόταν γραμμή στο τέλος, σε αντίθεση με πριν που εκτύπωση και υπολογισμός γίνονταν παράλληλα.

Στην συγκεκριμένη υλοποίηση που έχουμε y \* x διαστάσεις κάθε διεργασία γράφει σε πολλές γραμμές που της αντιστοιχούν. Πχ εάν είχαμε 6 διεργασίες τότε η πρώτη διεργασία θα έγραφε στις γραμμές 0, 6, 12, κλπ. Σε μια υλοποίηση N \* x κάθε διεργασία θα έγραφε κάθε φορά στην ίδια γραμμή, γράφοντας “πάνω" από τον προηγούμενο υπολογισμό. Αυτό σημαίνει ότι η εκτύπωση θα γινόταν κάθε φορά που συμπληρώνεται η τελευταία σειρά του πίνακα, εν αντιθέσει με προηγουμένως που γινόταν μία και μόνο εκτύπωση στο τέλος.

Στο πρώτο μέρος της 2ης άσκησης επιλέξαμε να δημιουργήσουμε έναν πίνακα από σημαφόρους, του οποίου το μέγεθος είναι ίσο με το πλήθος των διεργασιών. Με αυτόν τον τρόπο, κάθε παιδί έχει τον δικό του σημαφόρο και ενημερώνεται όταν έχει έρθει η σειρά του να δράσει. Επιπλέον, με το που τελειώσει την λειτουργία του, δηλαδή το output της γραμμής που είχε αναλάβει, ενημερώνει τον επόμενο σημαφόρο ότι πρέπει να δράσει. Έτσι, επομένως, εξασφαλίζεται ότι θα τυπωθούν τα outputs με την σωστή σειρά, αφού για να ενεργοποιηθεί ένας σημαφόρος πρέπει να λάβει σήμα από τον προηγούμενό του.

Στο δεύτερο μέρος της άσκησης χρησιμοποιώντας 2 φορές, μία για κάθε άξονα, την create\_shared\_memory\_area φτιάχνουμε έναν κοινό buffer για να αποθηκεύουν οι διεργασίες τα compute που υπολόγισαν και το output γίνεται σε όλες μαζί στο τέλος,