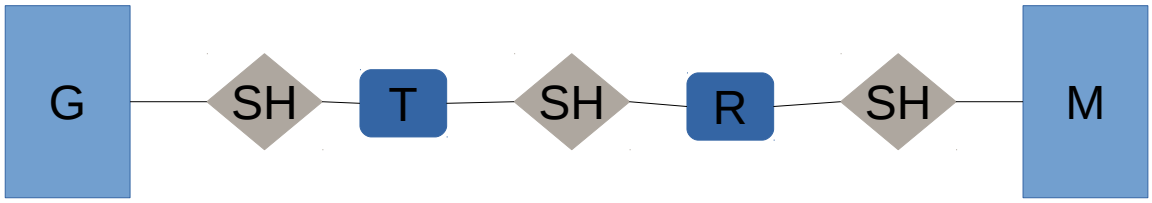
ΔΕΥΤΕΡΗ ΕΡΓΑΣΙΑ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ
Μανώλης Πιτσικάλης
ΑΜ: 1115201300143

Εκτελέσιμο επιλογές:

--lo	Το κατώτερο μέγεθος μιας διεργασίας
--hi	Το ανώτερο μέγεθος μια διεργασίας
-t	Ο μέσος χρόνος για την δημιουργία διεργασίας
-T	Ο μέσος χρόνος για την αποστολή νέου μηνύματος
-a	'n' για **n**ext-fit 'b' **b**est-fit 'u' για b**u**ddy
-S	Για το μέγεθος της μνήμης (προφάνως για να δουλεψει σωστά το buddy πρέπει να είναι δυναμη του 2)
--time	ο “χρόνος”-clock ticks της εκτέλεσης

πχ ./memsim --lo 32 --hi 512 -t 20 -T 8 -S 1024 --time 5000 -a u

Πληροφορίες για την υλοποιήση:



Υπάρχουν δύο διεργασίες μία είναι αυτή που παράγει τις vp και τα μηνύματα και η άλλη είναι η διεργασία της μνήμης. Κάθε μια από αυτές έχει και ένα thread transmitter και receiver αντίστοιχα. Η χρήση τους είναι η εξής

1. Transmitter : το νήμα έχει προσβάση σε μία λίστα η οποία υπάρχει μέσα στην διεργασία του G και περίεχει τα μηνύματα της χρονικής στιγμής 'i' αφού λάβει τα μηνύματα τα προωθεί ένα ένα στο νήμα receiver. Επιπλέον έχει και μια ακεραια τίμη η οποία χρησιμοποιείται για να ενημερώνεται εμεσσα του receiver η διεργασία της M για την ώρα.
2. Receiver : το νήμα έχει προσβάση σε κοινή μνημη μεταξύ των δύο thread από που περνάει ένα μόνο μήνυμα, επίσης έχει πρόσβαση σε ουρά της διεργασίας M στην οποία και βάζει τα μηνύματα της χρονικής στιγμής 'i' όταν ληφθεί το τελευταιό μήνυμα από το νήμα transmitter ,επίσης στέλνει κ τον χρόνο για να ενημερώσει την M.

Οι διεργασίες λειτουργούν ως εξής:

1. Η G παράγει μηνύματα (vp_start,vp_swap,vp_stop) και τα εισάγει σε λίστα, τέλος τα προωθεί μαζί και την τρέχουσα ώρα στο transmitter και μπλοκάρει μέχρι η διεργασία της μνήμης να συγχρονιστεί με το ρολοί του.
2. Η M αφού λάβει τα μηνύματα απο το thread receiver οπώς και την ώρα τότε θα κάνει τόσες επαναλήψεις μέχρι να φτάσει την ώρα που έστειλε η G έτσι μ αυτο τον τρόπο η M δεν θα τελειώσει πριν από το M αλλά σίγουρα μετά. Σε

κάθε επανάληψη (clock-tick) ελέγχεται η ουρά αν κάποιο αίτημα έχει ικανοποιηθεί αν ναι τότε το μήνυμα βγαινει απο την ουρά w. Στην συνέχεια για κάθε μήνυμα που ήταν στην ουρά και εληφθησαν μήνυματα υπάρχει μια λίστα ops_i με τα μηνύματα σε ανάμονη αν κάποιο απο αυτά έχει ξοδευσεί το χρόνο αναμονής του τότε ελέγχεται αν μπορεί να ικανοποιήθει τότε μπαινει στην ουρά w μέχρι να ικανοποιηθεί. Για καθε εισερχόμενο μήνυμα αν υπάρχει ήδη κάποιο για την διεργασία στην οποία αναφέρεται μέσα στην ουρα w η σε κάποια απο τις ουρές ops_i θα μπει τότε στην κατάληλλη ουρά για αναμονή. 'Οταν ρολόι φτάσει την στιγμή που στάλθηκε από το G τότε η διεργασία μπλοκάρει (και ξεμπλοκάρει η G) μέχρι να λάβει νέα μηνύματα.

Για τον συγχρονισμό τών διεργασίων(κοινη μνήμη - μπλοκαρισμα) χρησιμοποιούνται 9 σημαφόροι και για την μεταφόρα των μηνυμάτων 3 κοινές μνήμες.
Όταν το ρολόι του generator λήξει τότε θα “σταλεί” (χρησιμοποιειται bool end) στο transmitter μηνυμα τέλους θα προωθήσει το μήνυμα στον receiver και θα κλείσει , και μετά ο receiver θα προωθήσει το μήνυμα στην Memory και θα τερματίσει θα τερματίσει η Memory και μέτα θα τερματίσει και η διεργασια Generator.

Waiting queue:

1. Χρησιμοποιείται μια λίστα **w** στην οποία μπαίνει ένα μήνυμα αν γίνει απόπειρα να μπει στην μνήμη και αποτύχει.
2. Χρησιμοποιείται μια λίστα **ops** στην οποία για κάθε id στην w εισέρχονται μηνύματα τα οποία θα έχουν καθυστέρηση μέχρι το μήνυμα που βρίσκεται στην w να βγει. Αν βγεί το μήνυμα τότε μέ κάθε clock tick, ο χρόνος για να στάλει το μήνυμα που είχε μείνει σε ανάμονη μείωνεται κατα 1, άν είναι ίσο με 0 τότε το μήνυμα ελέγχεται αν μπορεί να ικανοποιηθει ,αν ναι διαγραφεταί απο την ops και ο χρόνος για αποστολή του επόμενου μηνύματος στην ops για αυτην την vp (αν υπαρχει) μείωνεται με κάθε clock tick, αν δεν μπορεί να ικανοποιήθει τότε το μήνυμα διαγράφεται από την ops και εισάγεται στην **w** και τά υπόλοιπα μηνύματα μέσα στην ops για αυτη την διεργασία μπλοκάρονται μέχρι να ικανοποιηθεί το αίτημα του μηνύματος

Αποτελέσματα εκτελέσεων

Με κάθε εκτέλεση κρατείται αναλυτικό αρχείο με όνομα mem_sim.log με την εισαγώγη μηνυμάτων συμβάντων κατάσταση μνήμης κλπ.

Μερικά αποτελέσματα από τις εκτελέσεις του προγράμματος για χρόνο 5000.

Memory time product 10 10

Request size	Next-fit	Best-fit	Buddy
32	0.0876375	0.0881016	0.0875375
64	0.182966	0.161169	0.191337
128	0.360237	0.375269	0.383337
256	0.700037	0.66675	0.719187
512	0.918575	0.681388	0.9192
1024	0.92195	0.9141	0.92065
2048	0.9247	0.9225	0.9229

Γενικά από τις εκτελέσεις φαίνεται οτάν τα μεγέθη είναι σταθέρα και δυνάμεις του δύο οι αλγόριθμοι να έχουν ίδια απόδοση, όταν όμως το μέγεθος των αιτήσεων δεν είναι σταθερό φαίνεται οι Best-fit Next- fit να είναι ίδιοι ενώ ο buddy να είναι χειρότερος.

Memory time product 10 10

Request size	Next-fit	Best-fit	Buddy
32-64	0.12871	0.133229	0.148685
32-128	0.214854	0.229936	0.221071
32-256	0.415865	0.418351	0.427664
32-512	0.761304	0.699648	0.63468
32-1024	0.821124	0.832831	0.623446
32-2048	0.814312	0.817818	0.625951