Πανεπιστήμιο Πατρών
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
Τομέας Λογικού των Υπολογιστών

Τελική Έκθεση για την εργαστηριακή άσκηση 1 του μαθήματος Λογισμικό και Προγραμματισμός Συστημάτων Υψηλής Επίδοσης

Θωμάς Ρεπαντής
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Κύκλος Σπουδών Ηλεκτρονικής και Υπολογιστών
Έτος: Ε΄
Α.Μ.: 4218
Ομάδα: hpc6
E-mail: darkzero@otenet.gr

Πάτρα, 21.11.2001

Εισαγωγή. Σκοπός της εργασίας είναι η υλοποίηση βιβλιοθήκης χρόνου εκτέλεσης η οποία περιέχει βασικούς αλγορίθμους συγχρονισμού και η μελέτη της συμπεριφοράς αυτών, υλοποιώντας δοκιμαστικά προγράμματα που τους χρησιμοποιούν και πραγματοποιούν μετρήσεις χρόνου.

Υλοποίηση βιβλιοθήκης αλγορίθμων συγχρονισμού. Οι αλγόριθμοι συγχρονισμού που υλοποιήθηκαν είναι 3 αλγόριθμοι αμοιβαίου αποκλεισμού (Test and Test and Set Lock (TTAS LOCK), Ticket Lock (TICKET LOCK), Array Based Queue Lock (QUEUE LOCK)) και ένας αλγόριθμος καθολικού φράγματος (Sense Reversing Centralized Incremental Barrier (INCR BARRIER)). Δοθέντων των περιγραφών των αλγορίθμων και της γλώσσας υλοποίησης (C) ο σχεδιασμός οδήγησε σε υλοποίηση των εξής συναρτήσεων: Για κάθε αλγορίθμο αμοιβαίου αποκλεισμού συνάρτησης αρχικοποίησης της δομής του κλειδιού (lock), συνάρτησης απόκτησης και συνάρτησης απαλλαγής από το κλειδί. Για τον αλγόριθμο καθολικού φράγματος συνάρτηση αρχικοποίσης της δομής του φράγματος (barrier) και συνάρτηση φράγματος. Χρησιμοποιήθηκαν όπου ήταν απαραίτητες ατομικές πράξεις οι δοθέντες συναρτήσεις assembly test_and_set και fetch_and_add. Οι παραπάνω συναρτήσεις βρίσκονται υλοποιημένες στα αρχεία: ttas_lock.c, ticket_lock.c, queue_lock.c, incr_barrier.c. Χρησιμοποιώντας την ar rs (αντίστοιχη των ar r και ranlib) δημιουργήσαμε τη στατική βιβλιοθήκη χρόνου εκτέλεσης libsync.a. Προκειμένου να χρησιμοποιηθούν οι υπηρεσίες που προσφέρει από άλλα προγράμματα δημιουργήσαμε και το αντίστοιχο header file (libsync.h).

Υλοποίηση πολυνηματικών προγραμμάτων. Προκειμένου να ελέγξουμε τη βιβλιοθήκη που υλοποιήσαμε και να μελετήσουμε τη συμπεριφορά των αλγορίθμων συγχρονισμού υλοποιήσαμε πολυνηματικά δοκιμαστικά προγράμματα που χρησιμοποιούν την παραπάνω βιβλιοθήκη. Τα προγράμματα αυτά -ένα για κάθε αλγόριθμο συγχρονισμού- βασίστηκαν στο δοθέν παράδειγμα υλοποίησης πολυνηματικού προγράμματος στο Solaris, που ήταν και το λειτουργικό σύστημα που χρησιμοποιήθηκε για την ανάπτυξη και τις μετρήσεις. Για την περίπτωση των αλγορίθμων αμοιβαίου αποκλεισμού τα νήματα αυξάνουν ένα μετρητή με βήμα 1 και οι αλγόριθμοι φροντίζουν να ισοκατανεμηθούν οι αυξήσεις στα νήματα. Για την περίπτωση του αλγορίθμου καθολικού φράγματος κάθε νήμα αυξάνει ξεχωριστό μετρητή και οι αλγόριθμοι φροντίζουν οι αυξήσεις να γίνονται συγχρονισμένα από όλα τα νήματα. Τα παραπάνω προγράμματα βρίσκονται υλοποιημένα στα αρχεία: ci_ttasl.c, ci_ticketl.c, ci_queuel.c και ci_incrbar.c. Επιπλέον υλοποιήθηκε σειριακό πρόγραμμα αύξησης του μετρητή χωρίς χρήση νημάτων και συγχρονισμού στο αρχείο: ci_seq.c. Τα προγράμματα δέχονται

σαν είσοδο από τη γραμμή εντολών τον αριθμό των νημάτων που θα χρησιμοποιηθούν και την τιμή στόχο για τους μετρητές και σαν έξοδο δίνουν την τελική τιμή των μετρητών και τις ακόλουθες μετρήσεις χρόνου που αναλύονται ακολούθως. Σημειώνουμε επίσης ότι τόσο στον κώδικα υλοποίησης της βιβλιοθήκης, όσο και σε αυτόν υλοποίησης των εφαρμογών δηλώδαμε ως volatile τις μεταβλητές που αλλάζουν από περισσότερα του ενός νήματα για να αποφύγουμε λογικά λάθη λόγω παλαιών τιμών μεταβλητών σε καταχωρητές. Τέλος εξασφαλίσαμε με padding ότι δομές δεδομένων ή και μεταβλητές που προσπελαύνονται αποκλειστικά από διαφορετικά νήματα δε βρίσκονται στην ίδια cache line ώστε να αποφευχθεί το false sharing που μειώνει την απόδοση

Υλοποίηση μετρήσεων χρόνου. Υλοποιήσαμε κώδικα για τη μέτρηση των ακόλουθων χρόνων:

- Συνολικός χρόνος για την αύξηση του μετρητή.
- Χρόνος αναμονής για την είσοδο στην κρίσιμη περιοχή, χρόνος εντός της κρίσιμης περιοχής και χρόνος για την έξοδο από την κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού.
- Χρόνος αναμονής για την επίτευξη του φράγματος ανά νήμα για τον αλγόριθμο καθολικού φράγματος.

Οι μετρήσεις έγιναν με ακρίβεια μικροδευτερολέπτου που προσφέρει η συνάρτηση gettimeofday. Επιλέξαμε ο κώδικα μέτρησης χρόνου να είναι μέρος των εφαρμογών και όχι της βιβλιοθήκης, απλοποιώντας έτσι τον προγραμματισμό, αφού δε χρειάζεται ανταλλαγή των δεδομένων των μετρήσεων μεταξύ συναρτήσεων της βιβλιοθήκης και των εφαρμογών. Χρησιμοποιώντας τις εντολές προς τον προεπεξεργαστή της C #ifdef, #ifndef, #endif δώσαμε τη δυνατότητα τα προγράμματα να μεταγλωτίζονται και χωρίς να συμπεριλαμβάνεται ο κώδικας μέτρησης χρόνου. Αυτό εξαρτάται από τη σημαία MEASURE_TIME, στην οποία δίνουμε τιμή περνώντας το κατάλληλο όρισμα γραμμής εντολών στο μεταγλωττιστή.

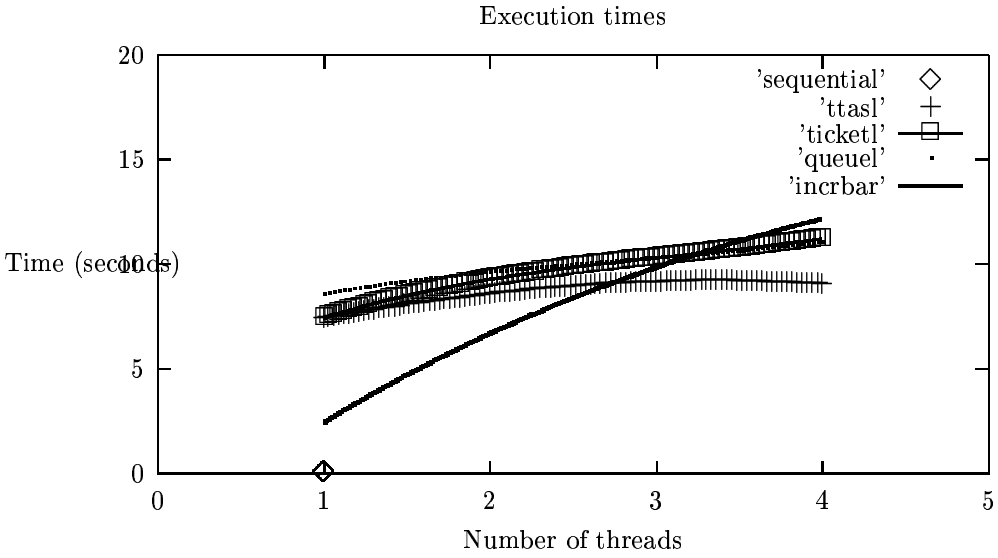
Makefile, ορίσματα γραμμής εντολών. Προκειμένου να απλοποιήσουμε τη διαδικασία δημιουργίας της βιβλιοθήκης και των εφαρμογών και να αποφεύγουμε άσκοπες μεταγλωττίσεις πηγαίου κώδικα που δεν έχει τροποποιηθεί χρησιμοποιήσαμε τη make και γράψαμε το κατάλληλο Makefile. Εκεί φαίνονται και οι παράμετροι που περάσαμε στο μεταγλωττιστή (gcc): -DREENTRANT

-D_SMP_ (για το threading) -DMEASURE_TIME (για τη μέτρηση χρόνου) -O6 (για τη μέγιστη βελτιστοποίηση, ώστε να μειώσουμε το χρόνο εκτέλεσης) και προαιρετικά -Wall για να βλέπουμε όλες τις προειδοποιήσεις και στο συνδέτη (gcc): -L$(LIB_PATH) (για τον εντοπισμό της βιβλιοθήκης που υλοποιήσαμε) -lthread (για σύνδεση με τη βιβλιοθήκη νημάτων) -lsync (για σύνδεση με τη βιβλιοθήκη συγχρονισμού που υλοποιήσαμε) για τη δημιουργία των εφαρμογών. Αντίστοιχα κατά τη μεταγλώττιση της βιβλιοθήκης περνάμε τις ακόλουθες παραμέτρους στο μεταγλωττιστή: -mcpu=ultrasparc -DMEASURE_TIME -O6 και προαιρετικά τη -Wall. Είναι φανερό ότι πρέπει να δηλώσουμε τον επεξεργαστή του οποίου την assembly χρησιμοποιουμε στις συναρτήσεις υλοποίησης των ατομικών πράξεων.

Μετρήσεις. Οι μετρήσεις -όπως και η υλοποίηση- πραγματοποιήθηκαν σε συμμετρικό πολυεπεξεργαστικό σύστημα κοινής μνήμης με 4 επεξεργαστές UltraSparcII @ 400MHz και λειτουργικό σύστημα Solaris (galois.ceid.upatras.gr). Δυστυχώς το σύστημα χρησιμοποιούνταν και από άλλους χρήστες (4) κατά τη διάρκεια των μετρήσεων. Προσπαθήσαμε όμως να εκτελούμε τις μετρήσεις σε χρονικές στιγμές που ο φόρτος του συστήματος -όπως δίνεται από την uptime- ήταν χαμηλός, δυστυχώς όμως υπαρκτός (0.4-0.6). Η -κοινή για όλα τα προγράμματα- τιμή στόχος για το μετρητή (3000000) ήταν τέτοια ώστε το πολυνηματικό πρόγραμμα που χρησιμοποιεί TTAS LOCK να εκτελείται σε 8 δευτερόλεπτα περίπου. Εκτελέσαμε το σειριακό πρόγραμμα και τα πολυνηματικά προγράμματα για 1, 2, 3 και 4 νήματα και μετρήσαμε συνολικό χρόνο εκτέλεσης και τις επιμέρους μετρήσεις χρόνου ανά νήμα που αναφέραμε. (Δίνοντας ps -L κατά τη διάρκεια του χρόνου εκτέλεσης είδαμε τα νήματα που δημιουργήσαμε και τα επιπλέον νήματα ελέγχου που κατασκευάζει αυτόματα η βιβλιοθήκη νημάτων χρόνου εκτέλεσης του λειτουργικού συστήματος και τα οποία δεν παίρνουν επεξεργαστικό χρόνο.) Ακολουθούν οι μετρήσεις υπό τη μορφή πινάκων και διαγραμμάτων και σχολιασμός - ερμηνεία των αποτελεσμάτων:

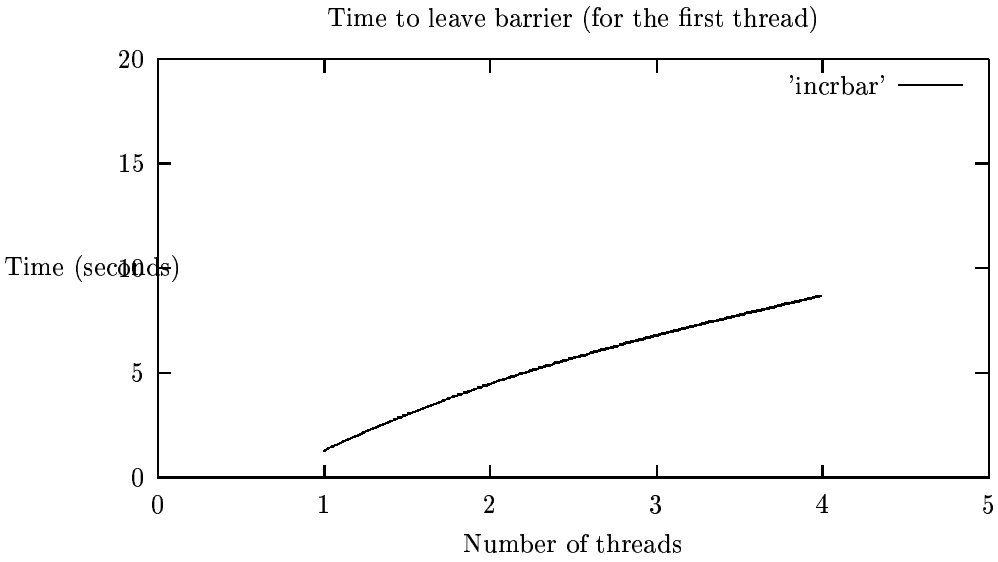
Συνολικός χρόνος για την αύξηση του μετρητή.

$NumOfThreads$	$seq(sec)$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$	$incrbar(sec)$
1	0.020452	7.475668	7.443133	8.528470	2.434491
2	–	8.854702	9.919350	9.896064	7.299423
3	–	9.614305	10.142482	10.044320	10.198547
4	–	9.088257	11.212895	11.008391	12.174739


Execution times
Time (seconds)
Number of threads
'sequential'
'ttasl'
'ticketl'
'queuel'
'incrbar'
0
5
10
15
20
0
1
2
3
4
5

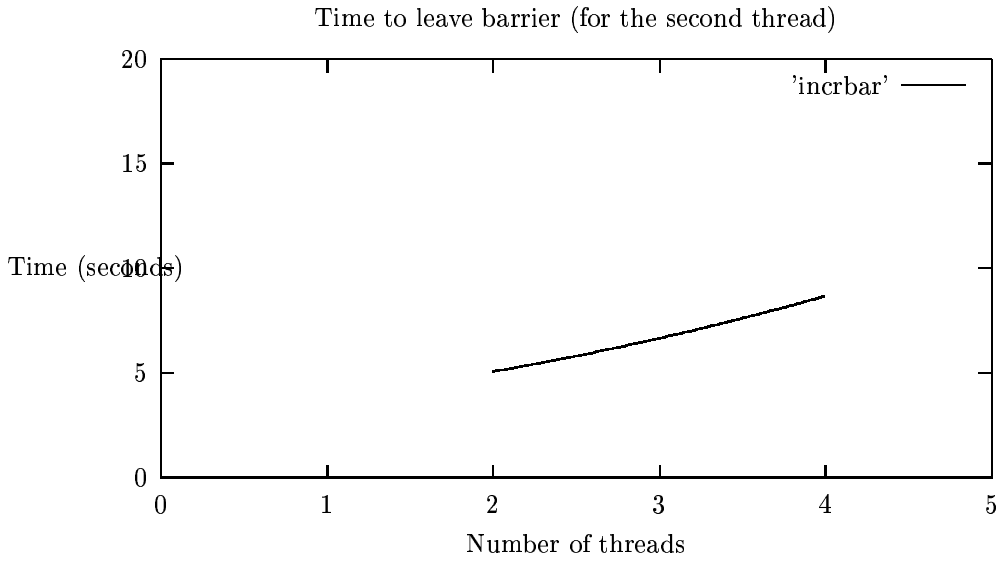
Χρόνος αναμονής για την επίτευξη του φράγματος ανά νήμα για τον αλγόριθμο καθολικού φράγματος (για το πρώτο νήμα).

$NumOfThreads$	$incrbar(sec)$
1	1.279253
2	5.072922
3	6.832716
4	8.698251


Time to leave barrier (for the first thread)
'incrbar'
Time (seconds)
Number of threads
0
5
10
15
20
0
1
2
3
4
5

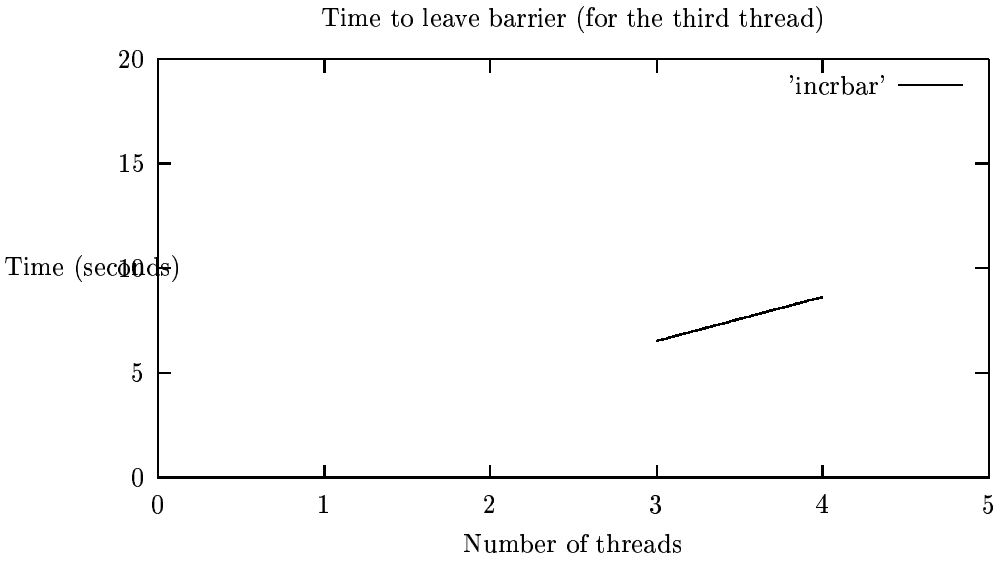
Χρόνος αναμονής για την επίτευξη του φράγματος ανά νήμα για τον αλγόριθμο καθολικού φράγματος (για το δεύτερο νήμα).

$NumOfThreads$	$incrbar(sec)$
1	–
2	5.067157
3	6.440653
4	8.670451


Time to leave barrier (for the second thread)
'incrbar'
Time (seconds)
Number of threads
0
5
10
15
20
0
1
2
3
4
5

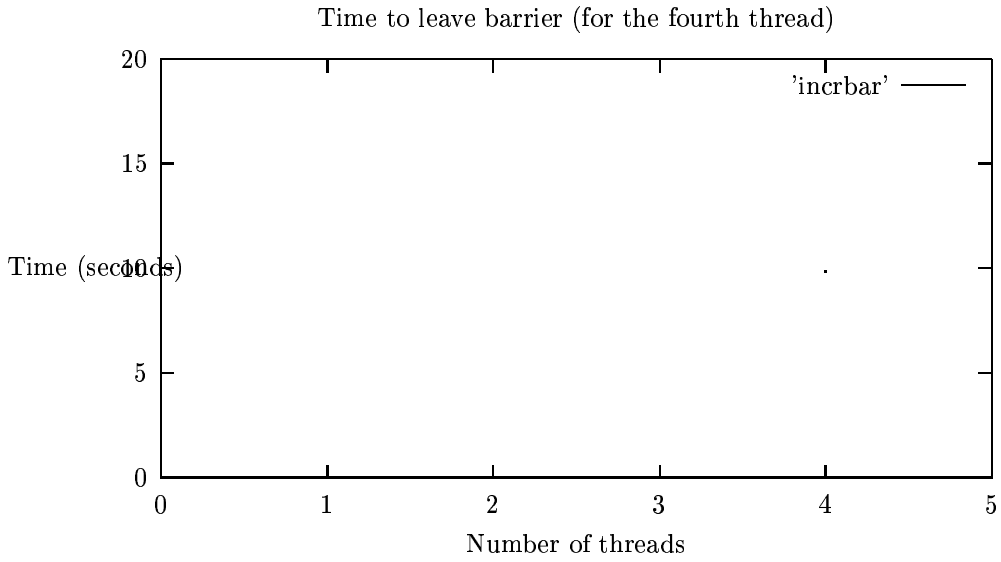
Χρόνος αναμονής για την επίτευξη του φράγματος ανά νήμα για τον αλγόριθμο καθολικού φράγματος (για το τρίτο νήμα).

$NumOfThreads$	$incrbar(sec)$
1	–
2	–
3	6.521257
4	8.625914


Time to leave barrier (for the third thread)
'incrbar'
Time (seconds)
Number of threads
0
5
10
15
20
0
1
2
3
4
5

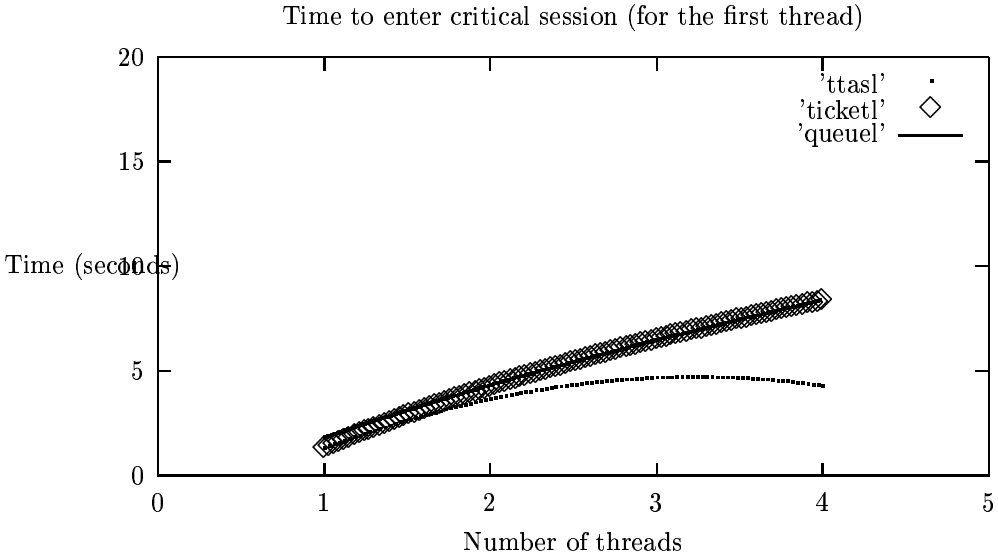
Χρόνος αναμονής για την επίτευξη του φράγματος ανά νήμα για τον αλγόριθμο καθολικού φράγματος (για το τέταρτο νήμα).

$NumOfThreads$	$incrbar(sec)$
1	–
2	–
3	–
4	9.837565


Time to leave barrier (for the fourth thread)
'incrbar'
Time (seconds)
Number of threads
0
5
10
15
20
0
1
2
3
4
5

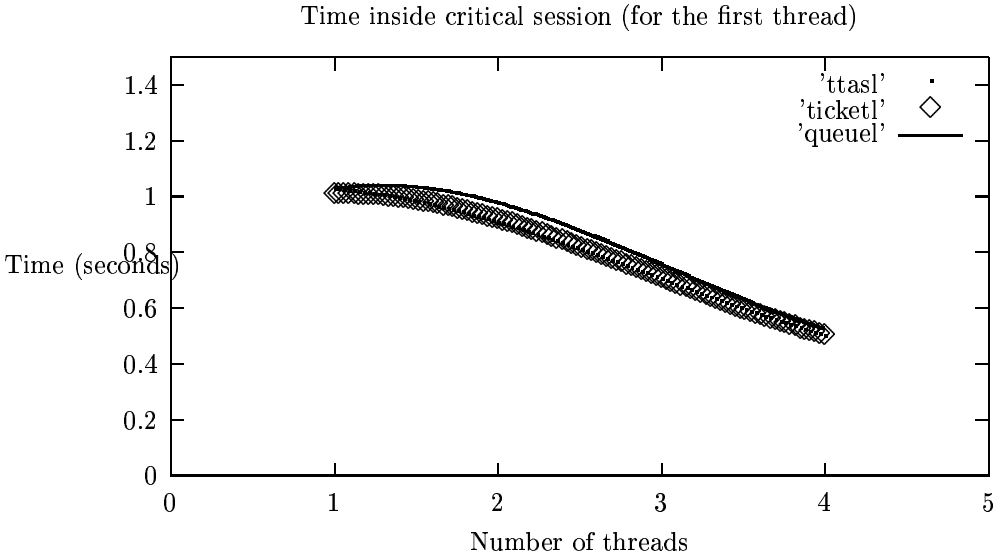
Χρόνος αναμονής για την είσοδο στην κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το πρώτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	1.244317	1.289892	1.812163
2	4.219099	4.626788	4.553098
3	5.334969	6.598308	6.531871
4	4.221880	8.340300	8.398936


Time to enter critical session (for the first thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
5
15
20
0
1
2
3
4
5

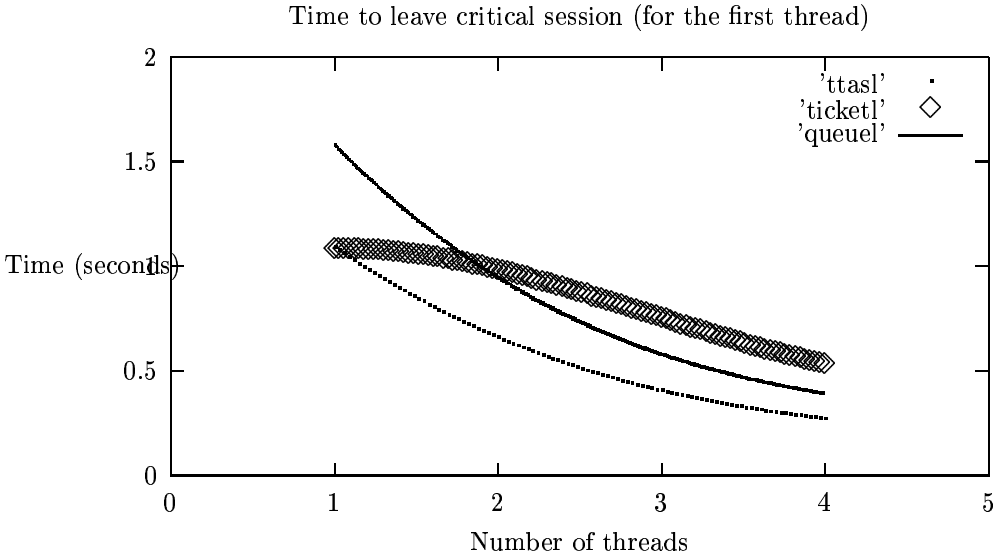
Χρόνος εντός της κρίσιμης περιοχής ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το πρώτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	1.021117	1.006139	1.028742
2	0.971922	1.020484	1.118181
3	0.671692	0.659559	0.709898
4	0.496247	0.500623	0.526439


Time inside critical session (for the first thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

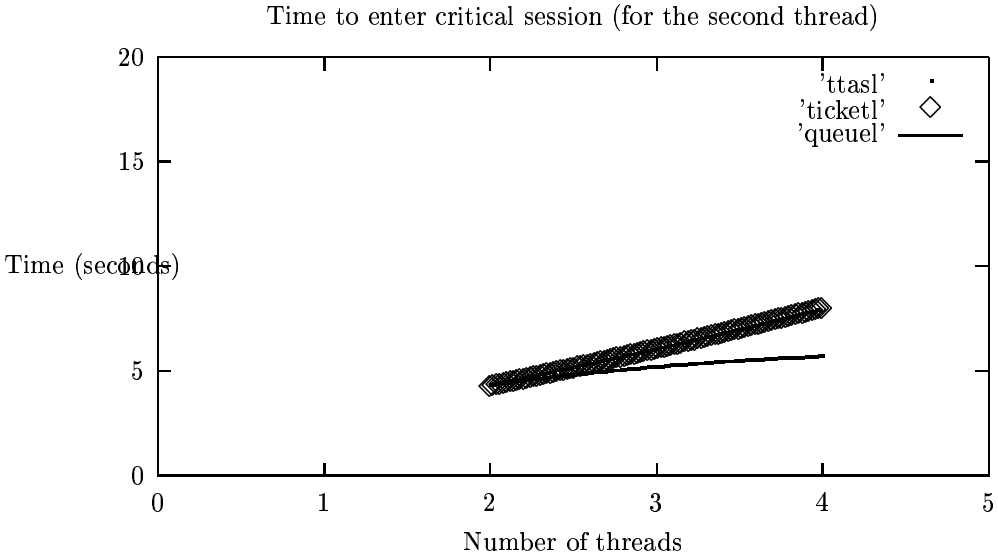
Χρόνος αναμονής για την έξοδο από την κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το πρώτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	1.086850	1.079664	1.582353
2	0.542493	1.091136	0.786159
3	0.359983	0.698927	0.521675
4	0.265446	0.528755	0.393678


Time to leave critical session (for the first thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.5
1
1.5
2
0
1
2
3
4
5

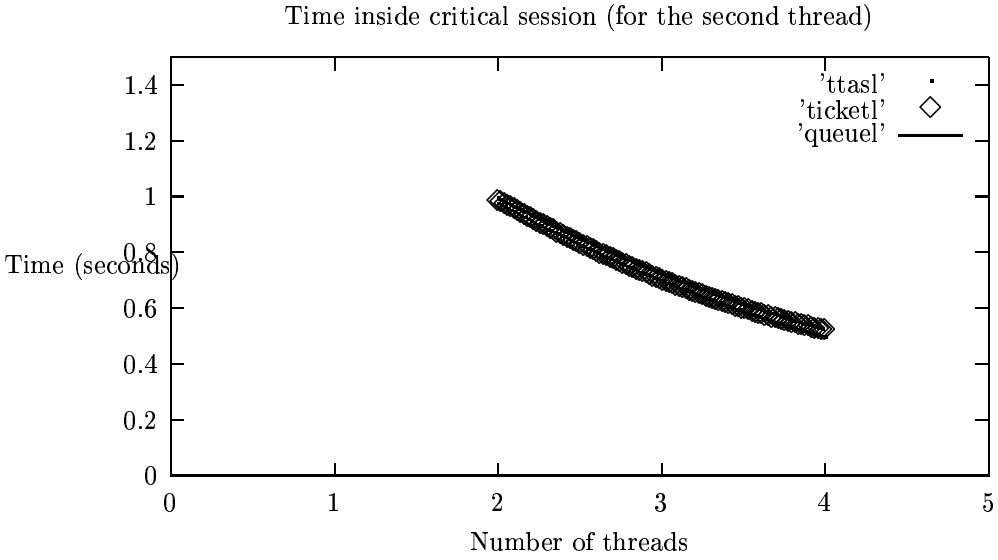
Χρόνος αναμονής για την είσοδο στην κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το δεύτερο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	4.272108	4.218223	4.328076
3	5.310802	5.896722	5.961089
4	5.620136	7.964387	7.952627


Time to enter critical session (for the second thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
5
15
20
0
1
2
3
4
5

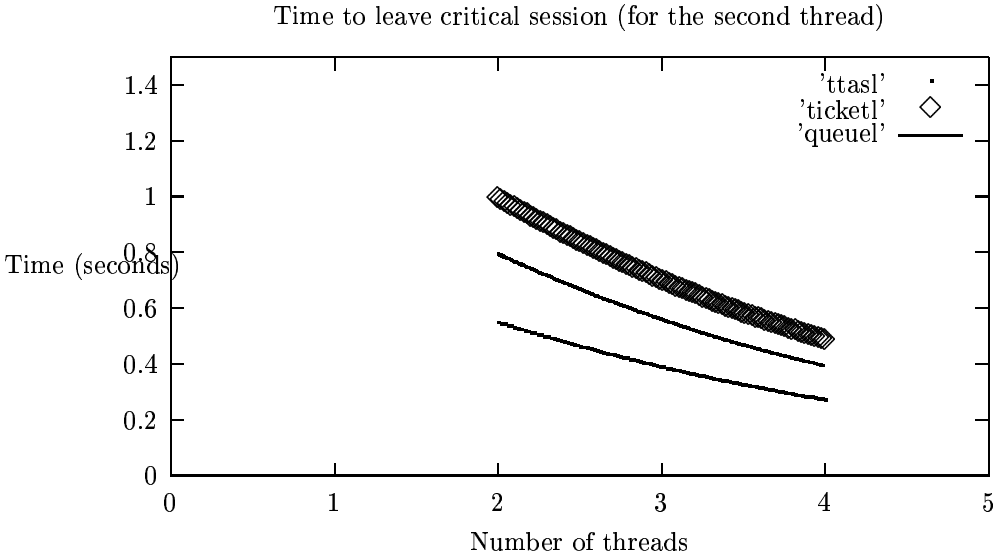
Χρόνος εντός της κρίσιμης περιοχής ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το δεύτερο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	0.969894	0.984863	0.996424
3	0.652155	0.648228	0.690846
4	0.490392	0.517337	0.524801


Time inside critical session (for the second thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

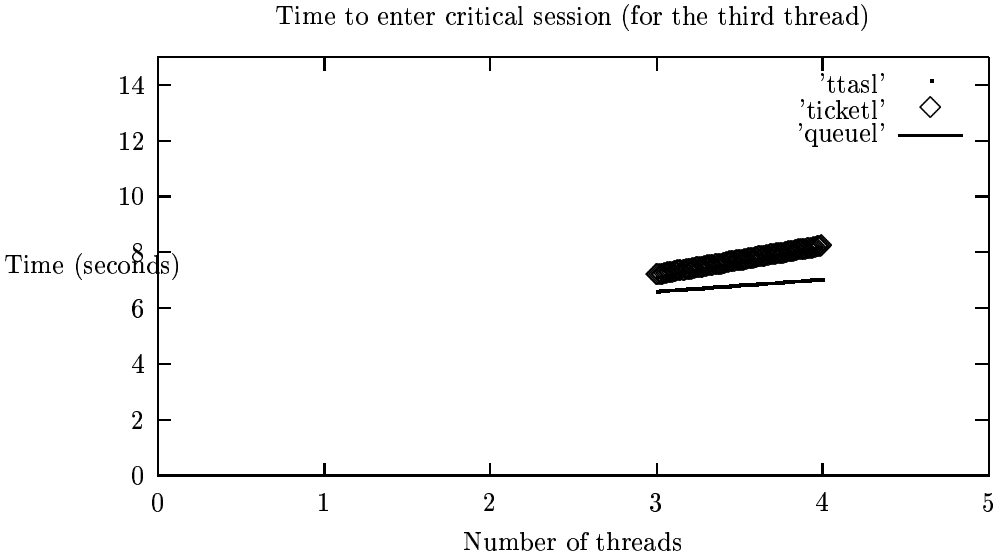
Χρόνος αναμονής για την έξοδο από την κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το δεύτερο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	0.543338	0.993575	0.795801
3	0.362365	0.656913	0.526890
4	0.266731	0.484656	0.394691


Time to leave critical session (for the second thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

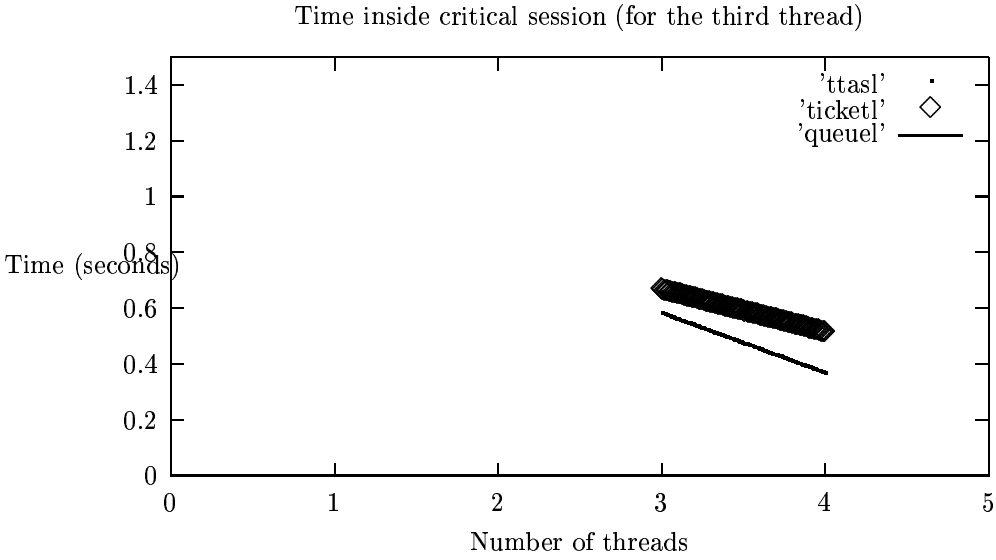
Χρόνος αναμονής για την είσοδο στην κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τρίτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	6.543653	7.148246	7.079426
4	6.968351	8.186422	8.195417


Time to enter critical session (for the third thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
2
4
6
8
10
12
14
0
1
2
3
4
5

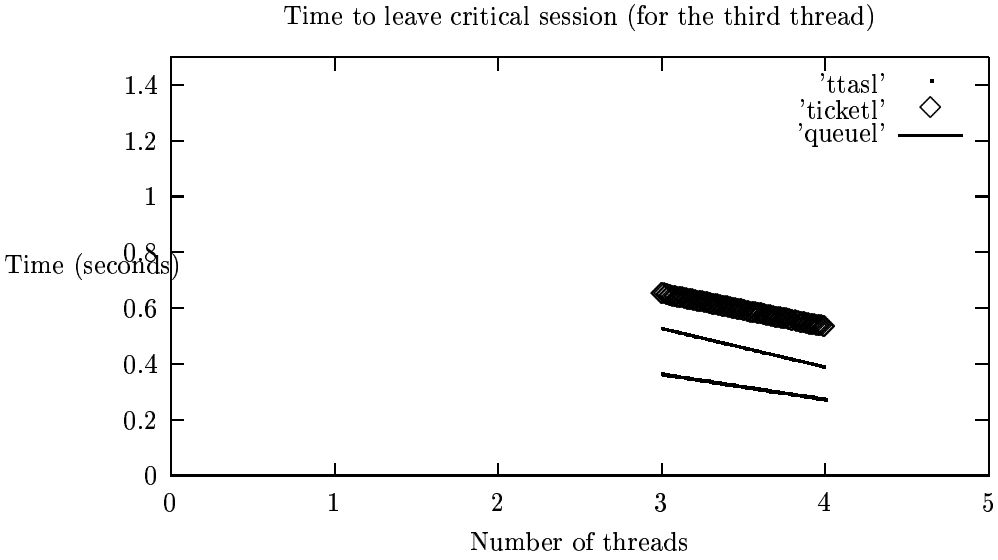
Χρόνος εντός της κρίσιμης περιοχής ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τρίτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	0.577038	0.664280	0.658402
4	0.362703	0.511774	0.487315


Time inside critical session (for the third thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

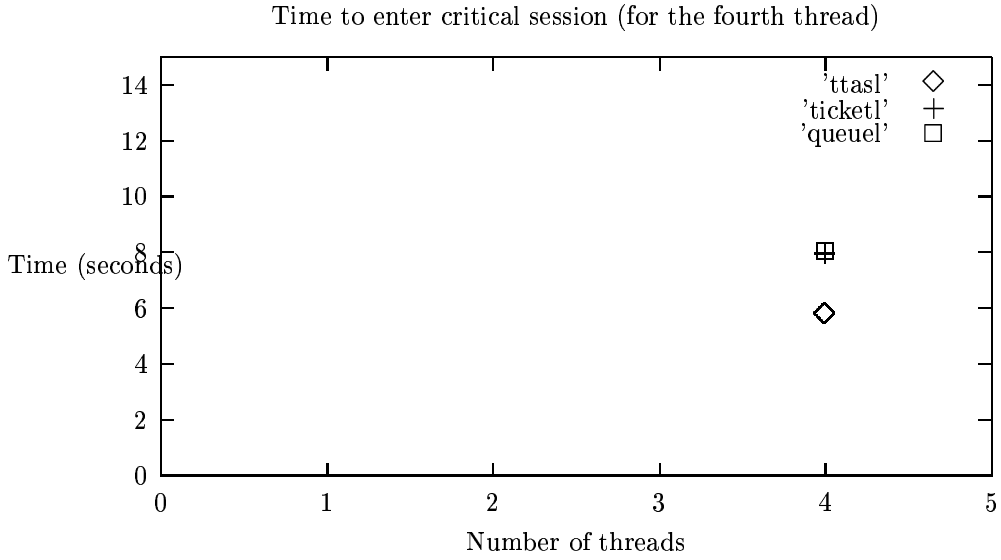
Χρόνος αναμονής για την έξοδο από την κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τρίτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	0.357196	0.648517	0.527890
4	0.266870	0.528697	0.389592


Time to leave critical session (for the third thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

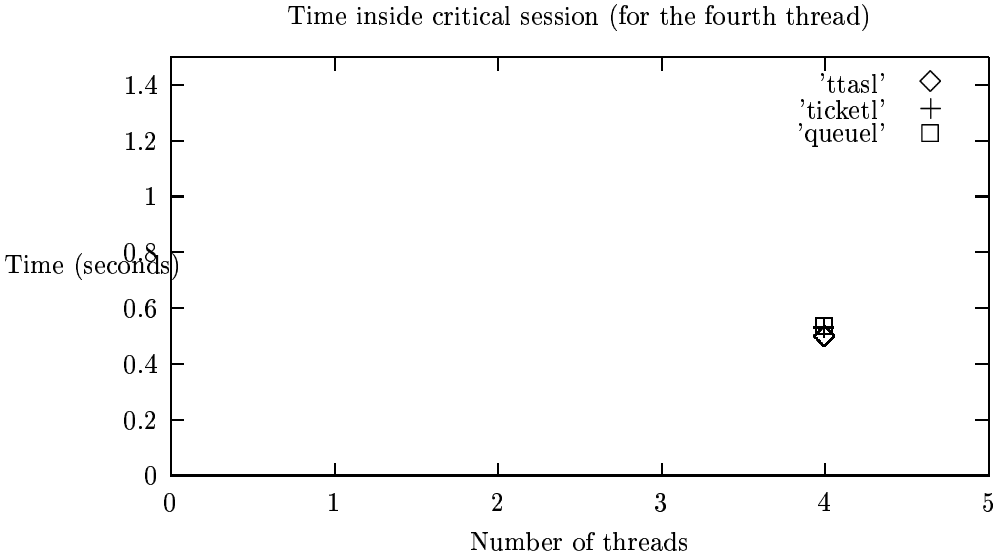
Χρόνος αναμονής για την είσοδο στην κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τέταρτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	–	–	–
4	5.768634	7.959763	7.949932


Time to enter critical session (for the fourth thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
2
4
6
8
10
12
14
0
1
2
3
4
5

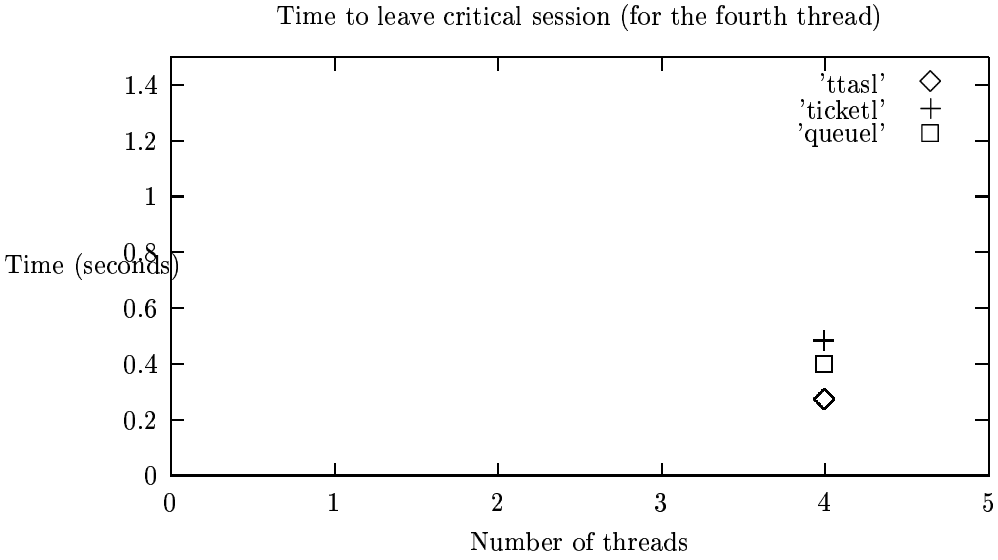
Χρόνος εντός της κρίσιμης περιοχής ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τέταρτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	–	–	–
4	0.493841	0.531490	0.527106


Time inside critical session (for the fourth thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

Χρόνος αναμονής για την έξοδο από την κρίσιμη περιοχή ανά νήμα για τους αλγορίθμους αμοιβαίου αποκλεισμού (για το τέταρτο νήμα).

$NumOfThreads$	$ttasl(sec)$	$ticketl(sec)$	$queuel(sec)$
1	–	–	–
2	–	–	–
3	–	–	–
4	0.267975	0.485613	0.392256


Time to leave critical session (for the fourth thread)
Time (seconds)
Number of threads
'ttasl'
'ticketl'
'queuel'
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5

Συμπεράσματα. Παρατηρούμε ότι ο χρόνος εκτέλεσης αυξάνει όσο αυξάνει ο αριθμός των νημάτων. Αυτό είναι αναμενόμενο, διότι αφένός μεν για την περίπτωση του αμοιβαίου αποκλεισμού όπως έχουμε υλοποιήσει την αύξηση του μετρητή τα νήματα δεν τον αυξάνουν παράλληλα αλλά σειριακά και υπάρχει το κόστος συγχρονισμού, αφέτέρου δε για την περίπτωση του καθολικού φράγματος οι ξεχωριστοί μετρητές αυξάνονται παράλληλα, αλλά όλα τα νήματα περιμένουν σε κάθε αύξηση το αργότερο νήμα, οπότε και πάλι το κόστος συγχρονισμού είναι μεγάλο. Με βάση τα παραπάνω είναι αναμενόμενο και το σειριακό πρόγραμμα να εκτελείται πολύ πιο γρήγορα από τα πολυνηματικά. Συγκρίνοντας τους αλγορίθμους αμοιβαίου αποκλεισμού παρατηρούμε ότι οι απλοί αλγόριθμοι άμεσης φραγής (TTAS Lock, Ticket Lock), στους οποίους δεν υπάρχει συγκεκριμένη σειρά για τα νήματα τρέχουν πιο γρήγορα (κυρίως για μικρό αριθμό νημάτων) από πιο πολύπλοκους (QUEUE Lock), στους οποίους υπάρχει συγκεκριμένη σειρά νημάτων. Αυτό ερμηνεύεται από την απλότητα των πρωτοκόλλων και από την αναισθησία τους στον πολυπρογραμματισμό που υλοποιεί το λειτουργικό σύστημα, προσπαθώντας να εξυπηρετήσει και άλλες εφαρμογές ή χρήστες. Για μεγαλύτερο αριθμό νημάτων είναι επίσης αναμενόμενο το ότι οι αλγόριθμοι που προσπαθούν να τα οργανώσουν καλύτερα, όπως το QUEUE Lock μπορεί να τρέχουν πιο γρήγορα. Αξίζει επίσης να παρατηρήσουμε ότι στους αλγορίθμους αμοιβαίου αποκλεισμού ο περισσότερος χρόνος χάνεται -για όλα τα νήματα και ιδιαίτερα για τα πρώτα- στην αναμονή για την είσοδο στην κρίσιμη περιοχή. Τέλος δεν μπορούμε άμεσα να συγκρίνουμε τους αλγορίθμους αμοιβαίου αποκλεισμού με τον αλγόριθμο καθολικού φράγματος (INCR BARRIER) (ο οποίος φαίνεται ωστόσο πως αποδίδει καλά), αφού οι εφαρμογές που τους χρησιμοποιούν δεν εκτελούν την ίδια εργασία.

Συνημμένα. Επισυνάπτεται ο πηγαίος κώδικας σε μορφή κειμένου. Επίσης σε δισκέττα επισυνάπτονται το παρόν κείμενο σε ηλεκτρονική μορφή, οι έξοδοι από τις εκτελέσεις των προγραμμάτων και οι αντίστοιχες μετρήσεις, το Makefile, καθώς και όλοι οι πηγαίοι κώδικες και τα αντίστοιχα εκτελέσιμα (και η βιβλιοθήκη).