**Λειτουργικά Συστήματα Υπολογιστών**

*Αναφορά στην 3η Εργαστηριακή Άσκηση*

Αλέξανδρος Σκούρας, 03120105

Ιωάννης Τσαντήλας, 03120883

*Εξάμηνο: Εαρινό 2022-23*

**3.1 Συγχρονισμός σε υπάρχοντα κώδικα**

* *Χρησιμοποιήστε το παρεχόμενο Makefile για να μεταγλωττίσετε και να τρέξετε το πρόγραμμα. Τι παρατηρείτε; Γιατί;*

Παρατηρούμε πως παράγονται δύο εκτελέσιμα αρχεία, με τα αντίστοιχα object files τους, αφού στο Makefile έχουμε:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Ενώ στον κώδικα έχουμε:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, γραφικά

Περιγραφή που δημιουργήθηκε αυτόματα

Επομένως, στο εκτελέσιμο **simplesync-atomic**, τα if-statements των συναρτήσεων increase/decrease θα έχουν τιμή **1** (και άρα θα εκτελεστεί το περιεχόμενο τους, δηλαδή υλοποίηση με atomic operations) ενώ στο εκτελέσιμο **simplesync-mutex**, τα if-statements θα έχουν τιμή **0**, και άρα θα εκτελεστούν τα περιεχόμενα των else (δηλαδή εκτέλεση με POSIX mutexes).

Περνώντας στο τρέξιμο των δύο εκτελέσιμων, και τα δύο επιστρέφουν μία φαινομενικά τυχαία τιμή στο value. Αυτό συμβαίνει γιατί και στις δύο περιπτώσεις δεν υπάρχει συγχρονισμός, καθώς η αύξηση και η μείωση της τιμής του value δεν γίνεται με την σειρά που επιθυμούμε, παρά είναι αποτέλεσμα race condition, μεταξύ των εντολών ++(\*ip) και --(\*ip). Όταν τα αντίστοιχα threads φτάνουν στις εντολές, μπορεί ένα να δεσμεύσει την CPU πολλαπλές φορές προτού το άλλο καταφέρει να εκτελέσει τη δική του εντολή ή οι εντολές assembly να εκτελεστούν με λανθασμένη σειρά.

* *Επεκτείνετε τον κώδικα του simplesync.c ώστε να υπάρχει συγχρονισμός.*

Αλλάζουμε λίγο τον κώδικα μας:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

**Ερωτήσεις**

Ερωτήσεις 1 και 2

Για το «**κακό**» αρχείο, χωρίς συγχρονισμό:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Ενώ, για τα **διορθωμένα** και συγκεκριμένα για το **atomic** εκτελέσιμο έχουμε έξοδο:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Ενώ για το **mutex**:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε πως το «**κακό**» αρχείο χρειάζεται **λιγότερο χρόνο** εκτέλεσης από ότι τα διορθωμένα. Αυτό συμβαίνει διότι στην πρώτη περίπτωση κάθε νήμα έχει ταυτόχρονη πρόσβαση στην κοινή μεταβλητή και εκτελεί ταυτόχρονα τις εντολές που του έχουν οριστεί(παράλληλος υπολογισμός). Στην δεύτερη περίπτωση, η μνήμη δεσμεύεται πρώτα για το ένα νήμα και αφού αυτό τελειώσει, δεσμεύεται από το άλλο, αυξάνοντας έτσι τον χρόνο εκτέλεσης.

Παρατηρούμε επίσης πως το **atomic** εκτελέσιμο είναι **πιο γρήγορο** από το mutex. Αυτό συμβαίνει διότι οι atomic εντολές είναι χαμηλού επιπέδου, σε αντίθεση με τα POSIX mutexes, τα οποία καταλαμβάνουν χρόνο και πόρους από το λειτουργικό σύστημα κάθε φορά που καλούμε τις συναρτήσεις **lock** και **unlock**, οι οποίες με τη σειρά τους για να εκτελεσθούν απαιτούν **system-calls**.

Ερωτήσεις 3 και 4

Για το εκτελέσιμο atomic, η εντολή **sync\_fetch\_and\_add** βρίσκεται στην γραμμή 52, ενώ το for-loop στην γραμμή 47. Εκτελώντας **gcc -DSYNC-ATOMIC -S -g simplesync.c** έχουμε:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Αντίστοιχα, για το mutex εκτελέσιμο, όπου οι lock, ++(\*ip) και unlock είναι στις γραμμές 59, 61 και 62 αντίστοιχα, έχουμε:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

**3.2 Παράλληλος υπολογισμός του συνόλου Mandelbrot**

*(Δεν έγιναν αλλαγές στις συναρτήσεις compute\_mandel\_line και output\_mandel\_line)*

*3.2.1 Με semaphores*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

*3.2.2 Με condition variables(Παραθέτουμε δύο εκδοχές)*

*3.2.2.1: Πολλά condition variables*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

*3.2.2.2: Ένα condition variable*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

**Ερωτήσεις**

Ερώτηση 1

Μέσω της εντολής:

**sem\_t \*sema**= malloc(nr\_threads \* sizeof(sem\_t));

Ορίζουμε τόσους σημαφόρους όσο και το πλήθος των threads που απαιτεί ο χρήστης. Ξεκλειδώνουμε τους σημαφόρους κυκλικά αφού τους έχουμε αρχικοποιήσει κατάλληλα(Σημαφόρος του thread[0] unlocked και οι υπολοιποι locked), δίνοντας κάθε φορά πρόσβαση στο κατάλληλο thread για να εκτυπώσει την γραμμή του Mandelbrot που του αναλογεί.

Ερώτηση 2

Το σειριακό πρόγραμμα μας δίνει:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, λευκό, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

Ενώ το παράλληλο, για 2 threads:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε πως ο συνολικός χρόνος εκτέλεσης μειώνεται(σχεδόν υποδιπλασιάζεται), όπως θα περιμέναμε λόγω παράλληλου υπολογισμού των γραμμών(Θέτοντας την compute\_mandel\_line εκτός του κρίσιμου τμήματος όπως θα εξηγήσουμε και στο ερώτημα 4). Δηλαδή ο υπολογιστικός φόρτος κατανέμεται σε 2 threads αντί για ένα με αποτέλεσμα την γρηγορότερη απόκριση του συστήματος.

Ερώτηση 3

Όπως δείξαμε και παραπάνω υλοποιούμε δύο εκδοχές του προγράμματος, μια με 1 μεταβλητή συνθήκης και μία με πολλαπλές(όσες και ο αριθμός των threads). Στην πρώτη περίπτωση κάνουμε χρήση της εντολής pthread\_cond\_broadcast με την οποία «ξυπνάμε» όλα τα νήματα τα οποία βρίσκονται σε κατάσταση αναμονής(pthread\_cond\_wait). Αντίθετα στην δεύτερη εκδοχή χρησιμοποιούμε την pthread\_cond\_signal για να ενεργοποιήσουμε το αμέσως επόμενο νήμα(που περιμένει) που είναι υπεύθυνο για να τυπώσει την γραμμή του. Όπως φαίνεται στην περίπτωση με τις πολλαπλές μεταβλητές συνθήκης έχουμε έναν πιο ελεγχόμενο τρόπο για να «κοιμίζουμε» και να «αφυπνίζουμε» τα κατάλληλα νήματα ενώ στο άλλο πρόγραμμα θα μπορούσε ένα νήμα να βγει και να ξαναμπεί σε κατάσταση αναμονής χωρίς λόγο.

Ερώτηση 4

Το παράλληλο πρόγραμμα που φτιάξαμε παρουσιάζει επιτάχυνση. Αυτό συμβαίνει διότι το κρίσιμο τμήμα περιλαμβάνει μόνο τη φάση εκτύπωσης των γραμμών και όχι τον υπολογισμό αυτών.

Ο υπολογισμός, σε αντίθεση με την εκτύπωση, δεν χρειάζεται να είναι μέρος του κρίσιμου τμήματος, αφού μπορεί να υλοποιηθεί από πολλαπλά threads ταυτόχρονα χωρίς να υπάρξει πρόβλημα, ενώ την εκτύπωση την αναλαμβάνει ένα thread ανά φορά ώστε να εμφανιστεί η κάθε γραμμή στην έξοδο με την σωστή σειρά.

Έτσι, για 5 και 9 threads έχουμε χρόνο εκτέλεσης αντίστοιχα:

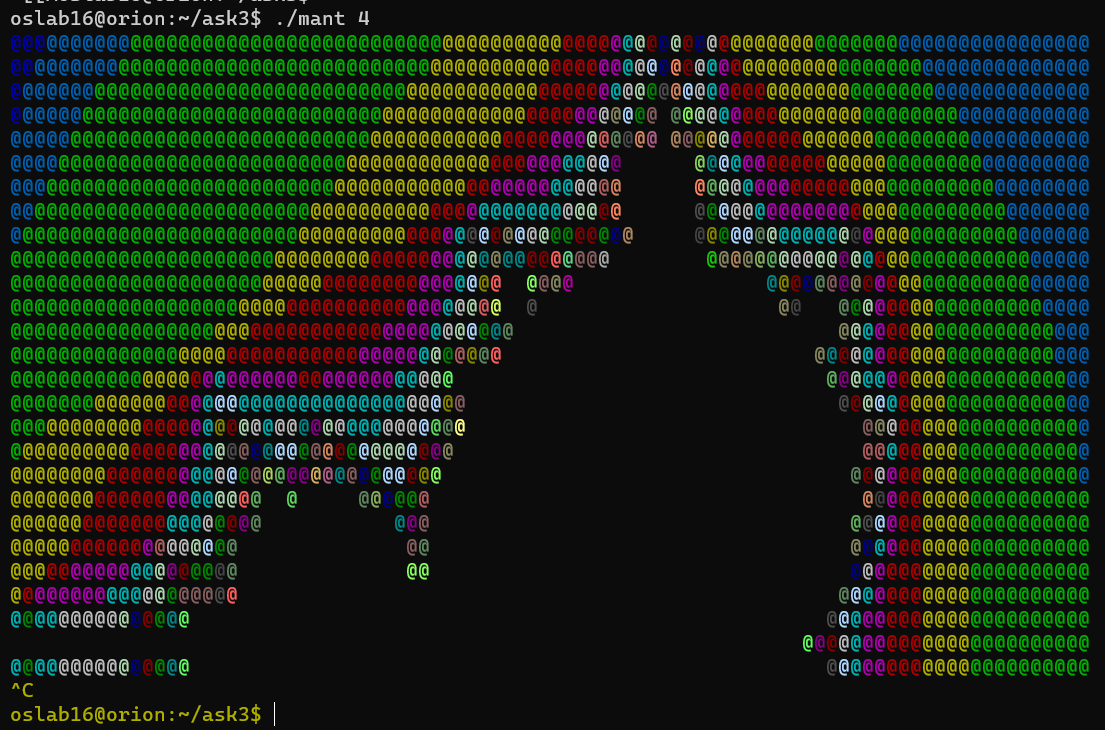
Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα Εικόνα που περιέχει κείμενο, γραμματοσειρά, λευκό, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Ερώτηση 5

Εάν πατήσουμε Ctrl-C ενώ το πρόγραμμα εκτελείται, αλλάζει το default χρώμα της γραμμής εντολών και υιοθετεί ένα από τα χρώματα που χρησιμοποιεί το Mandelbrot:



Για να εξασφαλίσουμε ότι ακόμη κι αν ο χρήστης πατήσει Ctrl-C, το τερματικό θα επαναφέρεται στην προηγούμενη κατάστασή του προσθέτουμε αυτή την συνάρτηση:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, γραφικά

Περιγραφή που δημιουργήθηκε αυτόματα

Και αυτή τη γραμμή κώδικα στην main:



Πλέον, όταν εκτελούμε Ctrl-C, το τερματικό είναι βαμμένο στα default χρώματα του:

