**Ηλίας Σέττας 3150156**
**Χρήστος Γκουρνέλος 3140033**

**Operating System: Ubuntu**
**Cores: 1 (1 of 4)**

**Main)** Ξεκινάει ελέγχοντας τον αριθμό των args και στη συνέχεια εντοπίζει τον αριθμό ακεραίων, νημάτων, seed και τον τύπο συγχρονισμού μέσω της συνάρτησης find(). Ελέγχει για κατάλληλο αριθμό νημάτων και τύπου συγχρονισμού και στη συνέχεια αρχικοποιεί τον πίνακα array ο οποίος δέχεται τυχαίες τιμές μέσω της συνάρτησης randomize(). Γράφονται τα δεδομένα του στο αρχείο και ξεκινάει το μέτρημα χρόνου. Δημιουργείται ένας ακόμα πίνακας τύπου pthread για την αποθήκευση των νημάτων και ανάλογα με το mode δημιουργούνται τα νήματα μέσα σε μια for loop κάνοντας παράλληλα και έλεγχο σφαλμάτων. Κάθε νήμα λειτουργεί στη συνάρτηση sort σύμφωνα με την επιλογή του mode και μόλις τελειώσουν επιστρέφουν στο κύριο νήμα και καταστρέφονται. Για το array καλείται η συνάρτηση merge() για την συγχώνευση των επιμέρους τμημάτων και σταματάει το μέτρημα του χρόνου εκτέλεσης. Τέλος, τυπώνεται ο χρόνος στο τερματικό, γράφονται τα δεδομένα στο αρχείο και απελευθερώνεται μνήμη από το array.

**Find)** Για κάθε μία λέξη από τις 5 στα arguments δοκιμάζει να βρει αν οι πρώτοι της χαρακτήρες ταιριάζουν με αυτό που ψάχνουμε (-numbers=, -threads=, -seed=, -mode=). Αν όχι πάει στο επόμενο αλλιώς δημιουργεί έναν πίνακα char για να αποθηκεύσει το υπόλοιπο κομμάτι της λέξης που προφανώς είναι ο αριθμός και στη συνέχεια το μετατρέπει σε ακέραιο και το επιστρέφει. Σε περίπτωση σφάλματος επιστρέφεται -1.

**Randomize)** Χρησιμοποιεί την τιμή seed για τη δημιουργία του srand και μετά παράγει τυχαίους αριθμούς για κάθε θέση του πίνακα array. Βάλαμε όριο στους αριθμούς αυτούς numbers*10.

**Merge)** Έχουμε τον πίνακα array ο οποίος έχει ξεχωριστά τμήματα του ταξινομημένα από διαφορετικά νήματα. Τον αντιγράφουμε στον πίνακα arrayCopy και για κάθε νήμα κρατάμε σε έναν πίνακα τη θέση που βρίσκεται το μικρότερο στοιχείο του στον πίνακα. Στη συνέχεια διασχίζουμε τον array και σε κάθε θέση του βάζουμε το μικρότερο από τα στοιχεία που έχει κάθε νήμα. Όταν επιλέγεται ένα στοιχείο από κάποιο νήμα αυξάνουμε το pos του για να πάει στο επόμενο μικρότερο στοιχείο του. Έτσι ο πίνακας ταξινομείται και επιστρέφεται στην main.

**WriteFile)** Γράφει στο αρχείο results.dat όλα τα στοιχεία του array. Στην main καλείται 2 φορές, μια για να γράψει τον αρχικό πίνακα (Initial) και μία για τον τελικό ταξινομημένο(Sorted). Κατά το άνοιγμα και κλείσιμο του αρχείου γίνονται και οι κατάλληλοι έλεγχοι σφαλμάτων.

**Sort1)** Μόνο ένα νήμα μπορεί να διαβάζει και να γράφει στον πίνακα array. Μόλις ξεκινήσει η συνάρτηση χρησιμοποιείται κλείδωμα ώστε να σταματήσουν τα άλλα νήματα να διαβάζουν και να γράφουν. Υπολογίζεται το άνω και κάτω όριο του πίνακα που θα έχει πρόσβαση το συγκεκριμένο νήμα και με τη χρήση 2 for loop ταξινομούνται τα στοιχεία αυτά. Γίνεται έλεγχος για σφάλματα και στη συνέχεια εκτυπώνονται στο τερματικό τα στοιχεία του πίνακα που ταξινομήθηκαν μαζί με τον αριθμό του νήματος. Στη συνέχεια ξεκλειδώνει και έτσι τα υπόλοιπα νήματα μπορούν να συνεχίσουν να διαβάζουν και να γράφουν.

**Sort2)** Χρησιμοποιεί σχετικά τον ίδιο κώδικα με το sort1 αλλά καθώς ένα νήμα γράφει μπορούν και τα υπόλοιπα να διαβάζουν τον πίνακα. Έτσι το lock και το unlock μπαίνουν μέσα στις 2 loop και περικλείουν το κομμάτι κώδικα που γίνεται αλλαγή στη θέση 2 στοιχείων.

**Sort3)** Δεν χρησιμοποιείται καθόλου το κλείδωμα και έτσι μπορεί το κάθε νήμα να γράφει και να διαβάζει στοιχεία ταυτόχρονα με τα υπόλοιπα. Μοιράζεται τον ίδιο κώδικα με τις συναρτήσεις παραπάνω αλλά δεν χρησιμοποιεί lock.

**Print)** Για την εκτύπωση των αποτελεσμάτων η κάθε συνάρτηση ταξινόμησης χρησιμοποιεί ένα κοινό κομμάτι κώδικα. Αυτό το κομμάτι αρχικά κλειδώνει μια περιοχή χρησιμοποιώντας διαφορετικό mutex (όχι το mutex που χρησιμοποιήθηκε για να ελέγξουμε την ταξινόμηση) και περιμένει να του σταλεί σήμα για να συνεχίσει. Για να αποφύγουμε τα τυχαία wake ups το cond_wait είναι μέσα σε ένα while loop το οποίο λειτουργεί εφόσον το threadID είναι διαφορετικό από το ID του νήματος το οποίο πρέπει να εκτυπώσει την εργασία του. Όταν βρεθεί το νήμα το οποίο πρέπει να κάνει την εκτύπωση, αφού εκτυπώσει ό,τι είναι να εκτυπώσει, κάνει counter = counter - 1 και έπειτα κάνει broadcast το σήμα και unlock την περιοχή για να "ξυπνήσουν" και τα επόμενα threads. Εφόσον ένα νήμα τελειώσει όλη αυτήν την διαδικασία τερματίζεται.

Παρακάτω ακολουθούν τα αποτελέσματα για πίνακα με 65536 στοιχείων για καθένα από τους 12 δυνατούς τρόπους ταξινόμησης:

**-mode=1 –threads=2: 2257355**

**-mode=1 –threads=4: 1145294**

**-mode=1 –threads=8: 600342**

**-mode=2 –threads=2: 2237878**

**-mode=2 –threads=4: 1130281**

**-mode=2 –threads=8: 570895**

**-mode=3 –threads=2: 2256034**

**-mode=3 –threads=4: 1135224**

**-mode=3 –threads=8: 570525**

**Παρατήρηση:** Βλέπουμε ότι ο 3$^{ος}$ τρόπος συγχρονισμού είναι και ο πιο γρήγορος. Επίσης γνωρίζουμε ότι το κλείδωμα του πίνακα για ανάγνωση και εγράφη όταν χρησιμοποιείται από κάποιο νήμα στη συγκεκριμένη εργασία δεν αποτρέπει σφάλματα καθώς το κάθε νήμα εργάζεται στο δικό του τμήμα του πίνακα και δεν επηρεάζει τους τομείς των άλλων νημάτων. Συνεπώς μας ενδιαφέρει η ταχύτητα και η καλύτερη επιλογή θα ήταν το mode=3. Να σημειωθεί ότι οι παραπάνω χρόνοι καταγράφηκαν χωρίς τη χρήση printf() για την εκτύπωση των αποτελεσμάτων στο τερματικό.