Λειτουργικά Συστήματα
1η Εργασία

Παναγιώτης Κάτσος(3180077),
Επαμεινώνδας Ιωάννου(3140059),
Πέτρος Τσότσι(3180193)

Περίληψη

Η συγκεκριμένη εργασία απαιτεί χρήση των νημάτων POSIX threads, μέσω των οποίων ένα κατάστημα θα δέχεται και θα αποστέλλει παραγγελίες. Ο κώδικας για την υλοποίηση αυτού του συστήματος πραγματοποιήθηκε στη γλώσσα C και αποτελείται από ένα header file που περιέχει όλες τις απαραίτητες δηλώσεις μεταβλητών,σταθερών και συναρτήσεων, και το c αρχείο με την υλοποίηση αυτών. Βασιστήκαμε σε αρκετά μεγάλο βαθμό στο υπάρχον υλικό φροντιστηρίων και εργαστηρίων του μαθήματος. Παρακάτω παρουσιάζεται λεπτομερής επεξήγηση των δύο αυτών αρχείων και με ποιο τρόπο επιτυγχάνεται το αρχικό ζητούμενο.

1 Ανάλυση header file

Αρχικά στο header file γίνονται include οι κλασσικές βιβλιοθήκες της c(stdlib,stdio) μαζί με τη βιβλιοθήκη pthread.h για χρήση νημάτων, αλλά και 3 επιπρόσθετες: η stdbool(για χρήση bool τύπων στην isNumber),η unistd για χρήση της συνάρτησης sleep στη main και η ctype για χρήση της συνάρτησης isdigit. Έπειτα υπάρχουν οι προβλεπόμενες δηλώσεις σταθερών και μεταβλητών όπως Ncook(διαθέσιμοι παρασκευαστές), Noven(διαθέσιμοι φούρνοι),Norderlow, Norderhigh που αποτελούν το εύρος των τεμαχίων που μπορεί να έχει μια παραγγελία, Torderlow,Torderhigh που αποτελούν το εύρος του χρόνου μέσα στο οποίο θα έρχεται κάθε νέα παραγγελία, καθώς και Tprep(χρόνος παρασκευής μίας πίτσας), και Tbake(χρόνος ψησίματος). Περιέχονται επίσης οι δηλώσεις των κατάλληλων mutexes που είναι το `oven_lock`(για τους φούρνους), το `cook_lock`(για τους παρασκευαστές), το `screen_lock`(για το κλείδωμα της οθόνης όταν τυπώνεται η έξοδος) και ένα mutex `time_lock` για το κλείδωμα όταν τοποθετείται ο χρόνος της κάθε παραγγελίας σε δυναμικό πίνακα που έχουν πρόσβαση όλα τα νήματα. Ακολουθούν 2 ακόμα δηλώσεις `pthread_cond` όπου χρησιμοποιούνται ως conditions στα mutexes παρασκευαστών και φούρνων, καθώς και η δήλωση 2 μεταβλητών τύπου timespec struct(χρησιμοποιούνται στη συνάρτηση gettime για τον υπολογισμό του χρόνου της κάθε παραγγελίας) και του pointer του `F_times`(δυναμικός πίνακας με τους χρόνους όλων των παραγγελιών). Τέλος, το header file ολοκληρώνεται με τη δήλωση της ρουτίνας που πρέπει να υλοποιήσει το κάθε νήμα(συνάρτηση order), τη δήλωση της συνάρτησης isNumber (ελέγχει για την ορθότητα των command line arguments) και ενός struct με παραμέτρους id και number of pizzas όπου θα περάσουμε ως argument στην δημιουργία του νήματος. Χρησιμοποιούμε αυτήν την τεχνική γιατί θέλουμε να έχουμε πρόσβαση σε 2 μεταβλητές σε κάθε νήμα.

2 Ανάλυση main

Όσον αφορά το κύριο πρόγραμμα, σε πρώτη φάση γίνονται κατάλληλοι έλεγχοι για τον αριθμό των arguments (δεν πρέπει να ξεπερνάνε τα 3) και για το αν αποτελούν θετικούς αριθμούς (μέσω της isNumber). Αφού έχουμε βεβαιώσει ότι τα ορίσματα είναι σε κατάλληλη μορφή εκχωρούμε στη μεταβλητή Ncust το πρώτο όρισμα (που αποτελεί τον αριθμό των

παραγγελιών) ,στη μεταβλητή seed το δεύτερο όρισμα (που αποτελεί τον τυχαίο σπόρο), καθώς και εκχωρούμε δυναμικά μέσω της malloc , τον κατάλληλο χώρο για τους πίνακες id (θα περιέχουν τα ids των παραγγελιών), `F_times` (θα περιέχουν τους χρόνους των παραγγελιών) και pthreads (θα περιέχουν τα actual threads). Έπειτα πραγματοποιείται μέσω της init αρχικοποίηση όλων των mutexes και conditions που δηλώσαμε στο header file (`oven_lock`, `cook_lock`,`oven_cond`,`cook_cond`,`screen_lock`,`time_lock`). Στη συνέχεια με ένα for loop από 0 ως Ncust(αριθμός παραγγελιών) δημιουργούμε τα νήματα ως εξής:

- Περνάμε στον πίνακα id το id του συγκεκριμένου νήματος
- Φτιάχνουμε μεταβλητή x τύπου `pizzas_ids` που είναι το struct που έχουμε προαναφέρει.
- Εκχωρούμε στην πρώτη του παράμετρο(`number_of_pizzas`) τις συνολικές πίτσες που θα έχει η συγκεκριμένη παραγγελία με χρήση της `rand_r` η οποία θα έχει ως όρισμα τον αντίστοιχο σπόρο που έχει δοθεί και θα δίνει τιμή μεταξύ Norderlow και Norderhigh βάσει εκφώνησης.
- Εκχωρούμε στη δεύτερη παράμετρο το id.
- Δημιουργούμε το συγκεκριμένο νήμα μέσω της `pthread_create` όπου περνάμε ως ορίσματα τη θέση μνήμης που θα δημιουργηθεί το αντίστοιχο νήμα, τη ρουτίνα που θα πρέπει να ακολουθήσει (order) και τον pointer στη μεταβλητή x.
- Μόλις δημιουργηθεί το νήμα, αναμένουμε για χρόνο y μέσω της sleep, που θα βρίσκεται στο διάστημα Torderlow και Torderhigh (με χρήση και πάλι της `rand_r`), μέχρι να έρθει η επόμενη παραγγελία.
- Έπειτα χρησιμοποιείται ακόμα ένας βρόχος επανάληψης μέσα στον οποίο καλείται η `pthread_join` για να περιμένουμε το κάθε νήμα να τελειώσει τη ρουτίνα του.
- Στην τελική φάση της main καταστρέφουμε όλα τα mutexes, conditions μέσω της destroy , τυπώνουμε στην οθόνη το μέγιστο χρόνο και μέσο χρόνο της κάθε παραγγελίας (ανασύροντάς τα από τον πίνακα `F_times`) και αποδεσμεύουμε τη μνήμη όπου χρειάζεται μέσω της free.

3 Ανάλυση order

Στη συνάρτηση order έχουμε οργανώσει όλο το multithreading διαβεβαιώνοντας ότι δεν υπάρχουν επικαλύψεις στις κρίσιμες περιοχές της μνήμης του προγράμματός μας. Όπως προείπαμε περνάμε ως argument μεταβλητή τύπου `pizzas_ids` και χρησιμοποιούμε 2 local variables id,pizzas για να αποθηκεύσουμε τις τιμές του struct του νήματος. Ύστερα μαρκάρουμε τον χρόνο εκκίνησης του νήματος καθορίζοντας τον με την συνάρτηση gettime και αποθηκεύοντας τον στον πίνακα `F_times` .Εδώ χρησιμοποιείται `lock` για το mutex `time_lock` καθώς αλλάζουμε μνήμη που είναι κοινή για όλα τα νήματα και έτσι αποτελεί κρίσιμη περιοχή. Έπειτα κάνουμε το αντίστοιχο `mutex_unlock`. Τώρα το νήμα επιχειρεί να δεσμεύσει έναν παρασκευαστή για να ολοκληρωθεί, οπότε κλειδώνει το mutex `cook_lock` και τσεκάρει αν υπάρχουν διαθέσιμες θέσεις, αν όχι μπαίνει σε ένα while loop όπου το ΄κοιμίζει΄μέχρι να βρεθεί έστω ένας διαθέσιμος παρασκευαστής, μέσω της `cond_wait`. Αν δεν είναι αυτή η κατάσταση τότε απλά δεσμεύει έναν μάγειρα μειώνοντας τη μεταβλητή Ncook κατά 1, γίνεται `cook_unlock`(μιας και βγαίνουμε από την κρίσιμη περιοχή) και περιμένει τον αντίστοιχο χρόνο προετοιμασίας της παραγγελίας, που ισούται προφανώς με pizzas*Tprep.Μετά ξαναμπαίνουμε σε κρίσιμη περιοχή, μιας και πρέπει να ελέγξουμε για διαθέσιμους φούρνους. Επομένως ακολουθώντας την ίδια φιλοσοφία γίνεται το lock για τους φούρνους(`oven_lock` και ακριβώς με την ίδια λογική αν δεν υπάρχει διαθέσιμος φούρνος(δηλαδή Noven=0) το νήμα ξαναμπαίνει σε ένα while loop και κοιμάται και πάλι μέσω της `cond_wait` μέχρι να ελευθερωθεί έστω ένας φούρνος. Αν υπάρχει διαθέσιμος φούρνος μειώνει τη μεταβλητή Noven κατά 1, κάνει unlock και κοιμάται για όσο χρόνο απαιτεί το ψήσιμο δηλαδή Tbake. Αφού τελειώσει και το ψήσιμο η παραγγελία θεωρείται έτοιμη για διανομή και ελευθερώνεται ο κατασκευαστής και ο φούρνος που απασχολούσε. Γι αυτό το λόγο πρέπει να αυξήσουμε τις αντίστοιχες μεταβλητές τους οπότε εισερχόμαστε και πάλι σε μια κρίσιμη περιοχή. Πρώτα ελευθερώνεται ο φούρνος οπότε πραγματοποιείται mutex lock γι αυτόν, αυξάνεται η μεταβλητή Noven κατά 1, ξυπνάει όσα νήματα κοιμούνται στο while loop που αφορά τους φούρνους, μέσω της `cond_signal` και γίνεται το `oven_unlock`. Ακριβώς η ίδια διαδικασία ακολουθείται για τους παρασκευαστές μόνο που αντί για Noven έχουμε Ncook και αντί για `oven_lock` και `oven_cond`, έχουμε `cook_lock` και `cook_cond`. Ύστερα εξάγουμε τον χρόνο ολοκλήρωσης του νήματος κάνοντας την κατάλληλη αφαίρεση με την προηγούμενη τιμή του πινάκα `F_times` και αποθηκεύουμε το

χρόνο αυτό πάλι στον `F_times` με `mutex_lock` στο `time_lock` όπως προαναφέρθηκε. Σε αυτήν την φάση η παραγγελία μας είναι πλήρως ολοκληρωμένη και το μόνο που μένει είναι να τυπωθεί το μήνυμα στην οθόνη με το id της παραγγελίας, τον αριθμό των τεμαχίων και το χρόνο ολοκλήρωσης. Χρησιμοποιείται πάλι `mutex_lock` για να μην μπερδευτούν οι γραμμές εκτύπωσης μεταξύ τους. Μετά το unlock το νήμα τερματίζει μέσω της `pthread_exit`.

4 Περιορισμοί

Ο μόνος περιορισμός που υπάρχει στο πρόγραμμα πέρα από τον έλεγχο για command line arguments στην αρχή, είναι ότι μετά από κάθε εντολή `mutex_lock,mutex_unlock,cond_wait,cond_signal` κλπ γίνεται έλεγχος για το αν η συγκεκριμένη εντολή επιστρέφει αριθμό διαφορετικό του 0. Σε αυτή την περίπτωση έχει υπάρξει σφάλμα και τερματίζει το πρόγραμμα.