Λειτουργικά Συστήματα
2η Εργασία

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

Περίληψη

Η δεύτερη εργασία αποτελεί επέκταση της πρώτης. Ο κορμός του κώδικα είναι ίδιος κι απλώς έχουν προστεθεί οι απαιτούμενες αλλαγές που χρειάζονται στο header και στο source file αντίστοιχα. Οι κύριες αλλαγές που χρειάστηκαν ήταν στη συνάρτηση order. 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 ζητούμενο της συγκεκριμένης εργασίας. Τα πιο τεχνικά κομμάτια αναλύονται στα σχόλια του κώδικα.

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), Ndeliverer(διαθέσιμοι διανομείς),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, Tprep(χρόνος παρασκευής μίας πίτσας),Tbake(χρόνος ψησίματος), καθώς και Tlow και Thigh που αποτελούν το εύρος στο οποίο θα βρίσκεται ο χρόνος διανομής της παραγγελίας. Περιέχονται επίσης οι δηλώσεις των κατάλληλων mutexes που είναι το `oven_lock`(για τους φούρνους), το `cook_lock`(για τους παρασκευαστές), το `deliverer_lock`(για τους διανομείς), το `screen_lock`(για το κλείδωμα της οθόνης όταν τυπώνεται η έξοδος), `time_lock` (για το κλείδωμα όταν τοποθετείται ο χρόνος παραλαβής της κάθε παραγγελίας στον αντίστοιχο πίνακα) και `time2_lock`(για το κλείδωμα όταν τοποθετείται ο χρόνος κρυώματος της κάθε παραγγελίας σε δυναμικό πίνακα) Ακολουθούν 3 ακόμα δηλώσεις `pthread_cond` όπου χρησιμοποιούνται ως conditions στα mutexes παρασκευαστών, φούρνων και διανομέων, καθώς και η δήλωση 1 μεταβλητής τύπου timespec struct(χρησιμοποιείται στη συνάρτηση gettime για τον υπολογισμό του χρόνου παραλαβής και κρυώματος κάθε παραγγελίας) αλλά και οι δηλώσεις των δυναμικών πινάκων `F_times` (θα περιέχει όλους τους χρόνους παραλαβής της κάθε παραγγελίας) και `Cold_times` (θα περιέχει όλους τους χρόνους κρυώματος). 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),τη δήλωση της συνάρτησης memory_check (ελέγχει για πιθανά memory leaks), τη δήλωση της συνάρτησης rc_check (ελέγχει για το αν όλες οι εντολές που αφορούν pthread είναι σωστές) και ενός struct με παραμέτρους id ,number of pizzas και `del_time` (χρόνος διανομής) όπου θα περάσουμε ως argument στην δημιουργία του νήματος. Χρησιμοποιούμε αυτήν την τεχνική γιατί θέλουμε να έχουμε πρόσβαση σε 3 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 μαζί με το όνομα του προγράμματος) 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), `Cold_times` (θα περιέχουν τους χρόνους κρυώματος των παραγγελιών) και pthreads (θα περιέχουν τα actual threads),όπου γίνονται και οι κατάλληλοι έλεγχοι για memory leaks μέσω της `memory_check` . Έπειτα πραγματοποιείται μέσω της init αρχικοποίηση όλων των mutexes και conditions που δηλώσαμε στο header file (`oven_lock`, `cook_lock`, deliverer_lock, `oven_cond`, `cook_cond`, `deliverer_cond`, `screen_lock`, `time_lock`, `time2_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.
- Εκχωρούμε στην τρίτη παράμετρο(`del_time`) το χρόνο διανομής που θα έχει η συγκεκριμένη παραγγελία με χρήση της `rand_r`, και θα δίνει τιμή μεταξύ Tlow και Thigh.
- 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 παράδοσης της κάθε παραγγελίας (ανασύροντάς τα από τον πίνακα `F_times`), καθώς και το μέγιστο και μέσο χρόνο κρυώματος με αντίστοιχο τρόπο(μέσω του πίνακα `Cold_times`) και αποδεσμεύουμε τη μνήμη όπου χρειάζεται μέσω της free.

3 Ανάλυση order

Στη συνάρτηση order έχουμε διαβεβαιώσει ότι το multithreading στο πρόγραμμά μας λειτουργεί με σωστό τρόπο εξασφαλίζοντας ότι δεν υπάρχουν επικαλύψεις στις κρίσιμες περιοχές της μνήμης του προγράμματός μας. Όπως προείπαμε περνάμε ως argument μεταβλητή τύπου `pizzas_ids` και χρησιμοποιούμε 3 local variables id, pizzas, `deliv_time` 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 και κάνει unlock . Σε αυτό το σημείο είναι που λαμβάνει μέρος η πρώτη βασική αλλαγή στον κώδικα σε σχέση με την πρώτη εργασία. Οι παρασκευαστές πλέον γίνονται διαθέσιμοι με το που βάλουν την παραγγελία στο φούρνο και δε χρειάζεται να την περιμένουν να ψηθεί. Επομένως βάσει αυτού, μετά το `unlock` για τους φούρνους πρέπει να γίνει το αντίστοιχο `cook_lock` ώστε να αυξηθεί ο παρασκευαστής κατά 1 μιας και είναι πλεόν διαθέσιμος, και να δοθεί το αντίστοιχο σήμα σε όσα threads περίμεναν για διαθέσιμο παρασκευαστή, μέσω της `cond_signal`. Έπειτα γίνεται το αντίστοιχο unlock για τους παρασκευαστές και η παραγγελία ψήνεται για χρόνο Tbake μέσω της sleep. Αφού έχουν ψηθεί οι πίτσες, αρχίζουν να κρυώνουν μέχρι να φτάσουν στο σημείο παραλαβής. Γι αυτό το λόγο γίνεται το αντίστοιχο `mutex_lock(time2_lock)` για να σώσουμε τον ακριβή χρόνο που αρχίζουν να κρυώνουν οι πίτσες, στον πίνακα `Cold_times` μέσω της gettime. Γίνεται το αντίστοιχο unlock στο συγκεκριμένο mutex. Μετά το ψήσιμο πρέπει να ελέγξουμε επίσης αν υπάρχουν διαθέσιμοι διανομείς. Με ολόιδια διαδικασία με παραπάνω γίνεται το αντίστοιχο lock στο `deliverer_lock` και ελέγχουμε αν υπάρχει διαθέσιμος διανομέας (αν δεν υπάρχει η παραγγελία περιμένει μέχρι να υπάρξει μέσω της `cond_wait`). Αν υπάρχει τότε οι διανομείς μειώνονται κατά 1(Ndeliverer–) και ο διανομέας παίρνει τις πίτσες της παραγγελίας από το φούρνο οπότε απελευθερώνεται ο συγκεκριμένος φούρνος. Γίνεται unlock στο `deliverer_lock`. Εφόσον ο φούρνος απελευθερώθηκε γίνεται `oven_lock` για να αυξηθούν οι φούρνοι κατά 1(Noven++) και να δοθεί σήμα σε όσες παραγγελίες περίμεναν για διαθέσιμο φούρνο μέσω της `cond_signal`. Ο διανομέας τώρα ξεκινάει από το κατάστημα για να παραδώσει την παραγγελία στο σημείο παραλαβής και μέσω της sleep κάνει χρόνο ίσο με `deliv_time` ο οποίος θα είναι μεταξύ 5 και 15 λεπτών βάσει εκφώνησης. Με το πέρας αυτού του σταδίου η παραγγελία έχει φτάσει επιτυχώς στο σημείο παραλαβής. Γι αυτό το λόγο θα πρέπει να κάνουμε τα αντίστοιχα locks στα `mutexes time2_lock, time_lock` για να αποθηκεύσουμε στους πίνακες `F_times, Cold_times` το συνολικό χρόνο παράδοσης της παραγγελίας και το χρόνο κρυώματος μέσω της συνάρτησης gettime και πάλι. Αφού συμβεί αυτό είμαστε σε θέση να εκτυπώσουμε το

απαιτούμενο μήνυμα με το id, το χρόνο παράδοσης αλλά και κρυώματος του συγκεκριμένου thread. Για το σκοπό αυτό χρησιμοποιούμε το `mutex screen_lock`. Τώρα το μόνο που μένει για να ολοκληρωθεί η ρουτίνα του thread , είναι να επιστρέψει ο διανομέας στο κατάστημα όπου κάνει και πάλι χρόνο `deliv_time` μέσω της sleep. Αφού επιστρέψει, είναι πλέον διαθέσιμος για διανομή επόμενης παραγγελίας οπότε γίνεται lock στο `mutex deliverer_lock` για να αυξηθεί η μεταβλητή κατά 1(Ndeliverer++) και να δοθεί σήμα και πάλι μέσω της `cond_signal` σε όσα threads περίμεναν για διαθέσιμο διανομέα. Το thread τελείωσε τη δουλειά του και κάνει exit.

4 Περιορισμοί

Οι μόνοι περιορισμοί που υπάρχουν στο πρόγραμμα πέρα από τον έλεγχο για τα command line arguments στην αρχή, είναι ότι μετά από κάθε εντολή malloc γίνεται έλεγχος για memory leaks μέσω της `memory_check`, όπως κι επίσης μετά από κάθε εντολή `mutex_lock, mutex_unlock, cond_wait, cond_signal` κλπ γίνεται έλεγχος μέσω της rc_check για το αν οι συγκεκριμένες εντολές επιστρέφουν αριθμό διαφορετικό του 0. Σε αυτή την περίπτωση έχει υπάρξει σφάλμα και τερματίζει το πρόγραμμα.