Λειτουργικά Συστήματα

2Η ΕΡΓΑΣΙΑ

p3140076 | p3140092 | p3140219

Δομή Αρχείων

Κάθε μέρος της εργασίας βρίσκεται στα καταλληλά ονόματα που ζητήθηκαν από την εκφώνηση. Τα headers περιέχουν τις μεθόδους που ζητήθηκαν από τον circular buffer. Το prodcons1 δουλεύει για αυθαίρετο αριθμό νημάτων το οποίο σημαίνει ότι δουλεύει και για έναν producer και για έναν consumer.

Υλοποίηση

Αρχίσαμε την υλοποίηση μας από το prodcons2. Το πρόγραμμα δέχεται τα arguments και εξετάζει αν είναι ορθά, δηλαδή αν το μέγεθος του buffer είναι μεγαλύτερο του 10, αν οι producer, οι consumer και το seed είναι μεγαλύτεροι του 0.

Έπειτα αρχικοποιεί όλα τα thread, δεν τα τρέχει. Δηλαδή ανάλογα με το input δημιουργει τα id για τους consumer και τους producers. Για τους producer χρησιμοποιούμε το producerParameters struct για να περάσουμε μέσα τις παραμέτρους, τα struct αρχικοποιούνται για κάθε producer.

Τα thread αρχικοποιούνται και αρχίζουν να τρέχουν. Πρώτα θα αρχίσουμε τους producers από την main και μετά τους consumers. Η producers έχουν καλούνται στην μέθοδο produce ενώ οι consumer στην μέθοδο consume. Αναλύουμε αυτές τις μεθόδους παρακάτω σαν λογική και θα πιο μετά σε λεπτομέρεια που χρησιμοποιήσαμε mutex και conditions. Τα prodcons1 και το prodcons2 έχουν ακριβώς τον ίδιο κώδικα. Το prodcons3 έχει τον μόνο διαφορετικό κώδικα. Οι διαφορές για το prodcons3 αναφερόνται παρακάτω:

- **Produce:** Με αυτήν την μέθοδο τρέχουν οι παραγωγοί. Η μέθοδός θα πάρει το struct που της περάσαμε και θα αρχικοποίηση τις παραμέτρους για κάθε thread producer. Ο κάθε producer θα τελειώσει όταν κάνει push στον buffer τις τιμές του. Για να ενθυλακώσουμε αυτή την λογική χρησιμοποιούμε for loop. Σε κάθε loop θα έχει κάνει push και έναν αριθμό. Μόνο στην prodcons3 ο παραγωγός περιμένει να τελειώσουν όλα τα thread. Αν όλα τα thread τελειώσουν τότε αρχίζει το printing. Το printing γίνεται με έναν counter. Για τους producer πρώτα θα ξεκινήσει ο 1, ο 2 κ.τ.λ. .
- **Consumer:** Με αυτή την μέθοδο τρέχουν οι καταναλωτές. Η μέθοδος απλά δέχεται ως όρισμα το id του. Οι consumers υλοποιούν while(true) διότι δεν μπορούμε να ξέρουμε από πριν πόσες φορές θα κάνει pop ένας consumer. Π.χ. μπορεί να τύχει ένας consumer να μην κάνει κανένα pop. Για να σπάσει από το while loop οι consumer πρέπει να έχουν καταναλώσει όλους τους αριθμούς που θα παραχθούν. Αυτόν τον αριθμό των ξέρουμε από πριν : Αριθμός Producer*Αριθμοί που Παράγονται από κάθε consumer. Επομένως οι consumer ενημερώνουν μια μεταβλητή, την popCount, οι οποία είναι κοινή για όλες τους consumers. Όταν αυτή η μεταβλητή είναι ίση με το σύνολο των αριθμών που θα παραχθούν τότε ο consumer βγαίνει από το while loop και σταματάει την εκτέλεση του. Μόνο στην prodcons3 ο consumer περιμένει να τελειώσουν όλα τα thread, αν αυτός είναι το

τελευταίο thread στέλνει μήνυμα ότι όλα τα thread τέλειωσαν. Αν όλα τα thread τελειώσουν τότε αρχίζει το printing. Το printing γίνεται με έναν counter. Για τους consumer πρώτα θα ξεκινήσει ο 1, ο 2 κ.τ.λ. . Οι consumer μοιράζονται το ίδιο printCount με τους καταναλωτές, για να ελέγχουμε αν είναι η σειρά των consumer τσεκάρουμε αν printCount = id + αριθμό producer. Δηλαδή για 4 producer και 4 consumer ο consumer 1 θα αντιστοιχεί στο printCount = 5

Ανάλυση Λειτουργιών Συχρόνισης των Thread

Παρακάτω αναλύουμε τις λειτουργίες που χρειαστήκαν χρονισμό και το πως τις υλοποιήσαμε:

- **Circular Buffer:** Είναι ο buffer που κρατάει τα δεδομένα. Γενικά όσο ένας producer ή ένας consumer κάνει ανάλογα push ή pop δεν θα πρέπει κάποιος άλλος να χρησιμοποιεί τον buffer. Ο buffer χρειάζεται και ενημερώνει το count και το current_size. Χρησιμοποιείται και σε producer και σε consumer. Οπότε βάζουμε το mutex κοινό για τα operation στον buffer. Όταν ο buffer θα είναι κενός ή άδειος αντίστοιχα θα περιμένει μέχρι να λάβει σήμα ότι δεν είναι κενός ή γεμάτος. Το σήμα ότι δεν είναι κενός ο buffer στέλνετε όταν ένας producer βάζει έναν αριθμό μέσα, οπότε θα ελευθερωθούν οι consumer. Οι consumer αν ο buffer είναι γεμάτος όταν καταναλώσουν έναν αριθμό θα στείλουν σήμα ότι δεν είναι γεμάτος ο buffer.
- **Write To File:** Εδώ έχουμε δύο διαφορετικούς mutex. Έναν για τους producers και έναν για τους consumers. Ο λόγος που χρησιμοποιούμε διαφορετικούς mutex είναι επειδή τα threads της κάθε κατηγορίας γράφουν σε διαφορετικό αρχείο. Το mutex αυτό κλειδώνει αφού ξεκλειδωθεί το mutex για τον buffer. Π.χ. Ένας producer κάνει lock το mutex του buffer κάνει push μια μεταβλητή και μετά ελευθερώνει το mutex. Έπειτα θα γράψει στο αρχείο. Και θα συνεχίσει.
- **Print με την σωστή σειρά:** Όταν κάθε νήμα έχει επιτελέσει τις απαραίτητες λειτουργίες θα περιμένει να τελειώσουν όλα τα υπόλοιπα. Αυτό γίνεται με ένα νέο mutex που γίνεται lock όταν ένα νήμα τελειώνει και unlock όταν περιμένει σήμα. Το τελευταίο σήμα θα στείλει broadcast το οποίο θα ελευθερώσει όλα τα νήματα. Ενώ θα μπορούσαμε να αρχίζουμε να κάνουμε print αν πχ ο producer 1 τελειώσει, χωρίς να περιμένω να τελειώσουν όλοι οι consumer η εκφώνηση δηλώνει στο τέλος της εκτέλεσης. Μετά χρησιμοποιούμε ένα άλλο mutex για το print. Το οποία σε συνδυασμό με σήματα βεβαιώνει ότι θα τυπώσει πρώτα ο producer 1 και του καθεξής. Αν ένα νήμα πάει να τυπώσει χωρίς να είναι η σειρά του απλά θα περιμένει για σήμα ότι κάποιο άλλο thread τελείωσε το print και θα ξαναδεί αν είναι η σειρά του, αν δεν είναι θα περιμένει ξανά.