ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ 2

Σπυρίδων Πάγκαλος, p3150133

Χρήστος Γκούμας, p3160026

ΑΝΑΛΥΣΗ ΕΡΩΤΗΜΑΤΩΝ:

prodcons1, prodcons2:

Ορίζουμε τις global variables, circ_buff, mutex, prod_condition, cons_condition, counter, cons, *in, *out. Όπου με τη σειρά αντιστοιχούν σε, circular queue, το mutex της ουρας, τα wait conditions των producers και των consumers, τον τελικο μετρητή για την κατανάλωση από όλα τα threads, το τελικό ποσό αριθμών που πρέπει να καταναλωθούν και τα αρχεία prod_in.txt, cons_out.txt.

main(main thread):

Αρχικά τσεκάρουμε τις μεταβλητές εισόδου και τις αρχικοποιούμε στις αντίστοιχες μεταβλητές, αλλιώς προβάλουμε κατάλληλο μήνυμα και τερματίζουμε το πρόγραμμα. Έπειτα αρχικοποιούμε πίνακες με p_thread, για όλους τους producers και όλους τους consumers, όπως επίσης και 2 πίνακες με τα αντίστοιχά τους id. Αρχικοποιούμε το counter = 0, και cons = number_of_producers*size_of_production. Αρχικοποιούμε τα 2 αρχεία όπως και τη circular queue. Αρχικοποιούμε επίσης το mutex, και τα 2 conditions. Μετά μπαίνουμε σε ένα for loop όπου αρχικοποιούμε τα thread των producers, στέλνοντας ένα struct με τις εξής πληροφορίες: id, μέγεθος παραγωγής και το σπόρο. Η ίδια ρουτίνα ακολουθείται για τους consumers, απλά περνάμε σα παράμετρο μόνο το id του consumer. Έπειτα κάνουμε join όλα τα threads σε διαφορετικά loops για τους producers και τους consumers. Τέλος κλείνουμε τα αρχεία, καταστρέφουμε το mutex, τα conditions για τους producers και τους consumers και απελευθερώνουμε τη μνήμη του circular queue και τερματίζουμε το πρόγραμμα.

producer(child thread):

Λαμβάνουμε τα arguments από το main thread μέσω struct και αρχικοποιούμε το πίνακα παραγωγής. Έπειτα μέσα σε ένα for loop για κ φορές(κ = size_of_production) ακολουθούμε την εξής ρουτίνα. Κλειδώνουμε την ουρά. Τσεκάρουμε ότι η ουρά είναι γεμάτη έτσι ώστε αν ισχύει να παγώσουμε τα thread των producers και να ξεκλειδώσουμε το πόρο για τους consumers. Μετά παίρνουμε ένα random αριθμό, τον εκτυπώνουμε στο αρχείο pron_in.txt με το id του producer, αποθηκεύουμε τον ίδιο αριθμό στον πίνακα παραγωγής και ωθούμε το στοιχείο στην ουρά. Έπειτα ξεμπλοκάρουμε τους producers, ξεμπλοκάρουμε τους consumers(broadcast) και ανοίγουμε τον πόρο. Τέλος κάνουμε sleep για 1 ms. Μετά από κ φορές

εκτυπώνουμε τα αποτελέσματα στη κονσόλα, απελευθερώνουμε τη μνήμη από τα arguments και τερματίζουμε το thread.

consumer(child thread):

Λαμβάνουμε το id του thread, αρχικοποιούμε ένα counter για τον αριθμό των δεδομένων που θα καταναλώσει ο consumer και αρχικοποιούμε τον πίνακα κατανάλωσης για 1 αριθμό. Έπειτα μπαίνουμε στην εξής ρουτίνα: Όσο δεν έχουμε φτάσει στο όριο των μέγιστων αριθμών(number_of_producers*size_of_producers), κλειδώνουμε την ουρά, αν η ουρά είναι άδεια τόσο παγώνουμε τα threads των consumers και ξεκλειδώνουμε τον πόρο για τους producers. Έπειτα κάνουμε pop από την ουρά, εκτυπώνουμε τον αριθμό στο αρχείο cons_out.txt με το id του consumer και αποθηκεύουμε τον αριθμό στον πίνακα κατανάλωσης και αυξάνουμε τον thr_count κατά 1. Έπειτα κάνουμε realloc για ένα αριθμό παραπάνω. Ξεκλειδώνουμε τα threads των consumers, ξεκλειδώνουμε τα threads των producers και ξεκλειδώνουμε τον πόρο. Τέλος κάνουμε sleep για 1 ms. Μετά από ν φορές(thr_count), εκτυπώνουμε τα αποτελέσματα στην κονσόλα και τερματίζουμε το thread.

prodcons3:

Ορίζουμε τις global variables, circ_buff, mutex, prod_condition, cons_condition, counter, cons, *in, *out. Όπου με τη σειρά αντιστοιχούν σε, circular queue, το mutex της ουρας, τα wait conditions των producers και των consumers, τον τελικο μετρητή για την κατανάλωση από όλα τα threads, το τελικό ποσό αριθμών που πρέπει να καταναλωθούν και τα αρχεία prod_in.txt, cons_out.txt.

main(main thread):

Αρχικά τσεκάρουμε τις μεταβλητές εισόδου και τις αρχικοποιούμε στις αντίστοιχες μεταβλητές, αλλιώς προβάλουμε κατάλληλο μήνυμα και τερματίζουμε το πρόγραμμα. Έπειτα αρχικοποιούμε πίνακες με p_thread, για όλους τους producers και όλους τους consumers, όπως επίσης και 2 πίνακες με τα αντίστοιχά τους id. Αρχικοποιούμε το counter = 0, και cons = number_of_producers*size_of_production. Αρχικοποιούμε τα 2 αρχεία όπως και τη circular queue. Αρχικοποιούμε επίσης το mutex, και τα 2 conditions. Μετά μπαίνουμε σε ένα for loop όπου αρχικοποιούμε τα thread των producers, στέλνοντας ένα struct με τις εξής πληροφορίες: id, μέγεθος παραγωγής και το σπόρο. Η ίδια ρουτίνα ακολουθείται για τους consumers, απλά περνάμε σα παράμετρο μόνο το id του consumer. Έπειτα κάνουμε join όλα τα threads σε διαφορετικά loops για τους producers και τους consumers. Μετά λαμβάνουμε από τα threads τα αποτελέσματά τους μέσω ενός struct που περιέχει το id του thread, πόσα items περιέχει ο πίνακας επιστροφής και τον πίνακα επιστροφής. Έπειτα ταξινομούμε τους producers και τους consumers κατά αύξουσα σειρά και εκτυπώνουμε στη κονσόλα τα αποτελέσματα πρώτα από τους producers και μετά από τους consumers. Τέλος κλείνουμε τα αρχεία, καταστρέφουμε το mutex, τα

conditions για τους producers και τους consumers και απελευθερώνουμε τη μνήμη του circular queue και τερματίζουμε το πρόγραμμα.

producer(child thread):

Λαμβάνουμε τα arguments από το main thread μέσω struct και αρχικοποιούμε το πίνακα παραγωγής. Έπειτα μέσα σε ένα for loop για κ φορές(κ = size_of_production) ακολουθούμε την εξής ρουτίνα. Κλειδώνουμε την ουρά. Τσεκάρουμε ότι η ουρά είναι γεμάτη έτσι ώστε αν ισχύει να παγώσουμε τα thread των producers και να ξεκλειδώσουμε το πόρο για τους consumers. Μετά παίρνουμε ένα random αριθμό, τον εκτυπώνουμε στο αρχείο pron_in.txt με το id του producer, αποθηκεύουμε τον ίδιο αριθμό στον πίνακα παραγωγής και ωθούμε το στοιχείο στην ουρά. Έπειτα ξεμπλοκάρουμε τους producers, ξεμπλοκάρουμε τους consumers(broadcast) και ανοίγουμε τον πόρο. Τέλος κάνουμε sleep για 1 ms. Μετά από κ φορές εκτυπώνουμε τα αποτελέσματα στη κονσόλα, απελευθερώνουμε τη μνήμη από τα arguments, δημιουργούμε το struct επιστροφής και αποθηκεύουμε σε αυτό το id το μέγεθος του πίνακα παραγωγής και το πίνακα παραγωγής και επιστρέφουμε το struct αυτό και τερματίζουμε το thread.

consumer(child thread):

Λαμβάνουμε το id του thread, αρχικοποιούμε ένα counter για τον αριθμό των δεδομένων που θα καταναλώσει ο consumer και αρχικοποιούμε τον πίνακα κατανάλωσης για 1 αριθμό. Έπειτα μπαίνουμε στην εξής ρουτίνα: Όσο δεν έχουμε φτάσει στο όριο των μέγιστων αριθμών(number_of_producers*size_of_producers), κλειδώνουμε την ουρά, αν η ουρά είναι άδεια τόσο παγώνουμε τα threads των consumers και ξεκλειδώνουμε τον πόρο για τους producers. Έπειτα κάνουμε pop από την ουρά, εκτυπώνουμε τον αριθμό στο αρχείο cons_out.txt με το id του consumer και αποθηκεύουμε τον αριθμό στον πίνακα κατανάλωσης και αυξάνουμε τον thr_count κατά 1. Έπειτα κάνουμε realloc για ένα αριθμό παραπάνω. Ξεκλειδώνουμε τα threads των consumers, ξεκλειδώνουμε τα threads των producers και ξεκλειδώνουμε τον πόρο. Τέλος κάνουμε sleep για 1 ms. Μετά από ν φορές(thr_count), δημιουργούμε το struct επιστροφής και αποθηκεύουμε σε αυτό το id το μέγεθος του πίνακα κατανάλωσης και το πίνακα κατανάλωσης και επιστρέφουμε το struct αυτό και εκτυπώνουμε τα αποτελέσματα στην κονσόλα και τερματίζουμε το thread.