

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ 2

Σπυρίδων Πάγκαλος, p3150133

Χρήστος Γκούμας, p3160026

ΑΝΑΛΥΣΗ ΕΡΩΤΗΜΑΤΩΝ:

prodcons1, prodcons2:

Ορίζουμε τις global variables, circ_buff, mutex, p_mut, prod_condition, cons_condition, counter, cons, *in, *out. Όπου με τη σειρά αντιστοιχούν σε, circular queue, το mutex της ουρας, το mutex για το print στη κονσόλα, 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 προβάλλουμε 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 τα mutexes, 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 πόρο. 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 τα mutexes, 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 prod_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 με αμοιβαίο αποκλεισμό ελέγχοντας ότι ο αριθμός των consumers που έχουν εκτυπώσει να είναι NULL και από την πλευρά των consumers ότι έχουν εκτυπώσει όλοι οι producers πρώτα, 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($\text{number_of_producers} * \text{size_of_producers}$), 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 ελέγχοντας ότι έχουν εκτυπώσει όλοι οι producers πρώτα και τερματίζουμε το thread.

prodcons3:

Ορίζουμε τις global variables, circ_buff, mutex, p_mut, prod_condition, cons_condition, counter, cons, *in, *out. Όπου με τη σειρά αντιστοιχούν σε, circular queue, το mutex της ουρας, το mutex που εκτυπώνει τα αποτελέσματα στη κονσόλα, τα wait conditions των producers και των consumers, τον τελικό 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 προβάλλουμε 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 = $\text{number_of_producers} * \text{size_of_production}$. Αρχικοποιούμε τα 2 αρχεία όπως και τη circular queue. Αρχικοποιούμε επίσης τα mutexes, 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 τα mutexes, 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 ($\kappa = \text{size_of_production}$) 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 prod_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 με αμοιβαίο αποκλεισμό ελέγχοντας ότι ο αριθμός των consumers που έχουν εκτυπώσει να είναι NULL και από την πλευρά των consumers ότι έχουν εκτυπώσει όλοι οι producers πρώτα, έπειτα ελέγχουμε το id του producer με το p_p(producer print counter) για να εκτυπώσουμε κατά αύξων αριθμό id, μετά 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 ($\text{number_of_producers} * \text{size_of_producers}$), 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 ελέγχοντας ότι έχουν εκτυπώσει όλοι οι producers πρώτα και έπειτα ότι το id του consumer ισούται με το c_p(consumer print counter) για να εκτυπώσουμε τα αποτελέσματα κατά αύξων αριθμό consumer id και τέλος τερματίζουμε το thread.