Λειτουργικά Συστήματα

Εργασία 3η: Producer – Consumer Problem

Ονοματεπώνυμο: Στεφανιώρος Μιχαήλ

ΑΜ: 1072774

Έτος: 4ο

Περίληψη:

Το πρόβλημα παραγωγού-καταναλωτή είναι ένα κλασικό παράδειγμα ενός προβλήματος συγχρονισμού πολλαπλών διεργασιών. Περιλαμβάνει δύο διεργασίες, τον παραγωγό και τον καταναλωτή, που μοιράζονται έναν κοινό buffer που χρησιμοποιείται για την αποθήκευση ενός συγκεκριμένου αριθμού στοιχείων. Ο παραγωγός παράγει δεδομένα και τα βάζει στον buffer, ενώ ο καταναλωτής παίρνει αυτά τα δεδομένα από τον buffer και τα «καταναλώνει».

Υπάρχουν πολλά θέματα που πρέπει να αντιμετωπιστούν κατά την εφαρμογή μιας λύσης στο πρόβλημα παραγωγού-καταναλωτή. Από την μια, είναι ο τρόπος συγχρονισμού της πρόσβασης στον κοινό buffer, έτσι ώστε ο παραγωγός και ο καταναλωτής να μην προσπαθούν να έχουν πρόσβαση ταυτόχρονα. Από την άλλη, είναι το πώς να διασφαλιστεί ότι ο παραγωγός δεν προσπαθεί να προσθέσει δεδομένα στον buffer όταν είναι γεμάτος και πώς να διασφαλιστεί ότι ο καταναλωτής δεν προσπαθεί να πάρει δεδομένα από τον buffer όταν είναι άδειος. Στην συγκεκριμένη περίπτωση, για την επίλυση των προβλημάτων αυτών χρησιμοποιώ δύο semaphores και ένα mutex.

Επεξήγηση κώδικα:

Χρησιμοποιώντας ως βάση τον κώδικα του φροντιστηρίου και έπειτα τον αλγόριθμο της εκφώνησης, κατάφερα να εντοπίσω τα σημεία που πρέπει να γίνονται τα wait() και signal() των semaphores καθώς και τα lock() και unlock() του mutex. Ο πρώτος semaphore (full) χρησιμοποιείται για την παρακολούθηση του αριθμού των στοιχείων στον buffer. Ο δεύτερος semaphore (empty) χρησιμοποιείται για την παρακολούθηση του αριθμού των κενών υποδοχών του buffer. Ο mutex χρησιμοποιείται για την προστασία του κοινόχρηστου πόρου (του buffer) από την ταυτόχρονη πρόσβαση των νημάτων του παραγωγού και του καταναλωτή.

Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

1. Το νήμα παραγωγού κάνει wait τον empty μέχρι να υπάρξει τουλάχιστον μία κενή θέση στον buffer.
2. Το νήμα παραγωγού κάνει lock τον mutex, προσθέτει ένα αντικείμενο στον buffer και μετά κάνει unlock τον mutex.
3. Το νήμα παραγωγού κάνει post τον full για να υποδείξει ότι υπάρχει τώρα ένα ακόμη στοιχείο στον buffer.
4. Το νήμα καταναλωτή κάνει wait τον full έως ότου υπάρξει τουλάχιστον ένα στοιχείο στον buffer.
5. Το νήμα καταναλωτή κάνει lock τον mutex, αφαιρεί ένα αντικείμενο από τον buffer και μετά κάνει unlock τον mutex.
6. Το νήμα καταναλωτή κάνει post τον empty για να υποδείξει ότι υπάρχει τώρα μια ακόμη κενή υποδοχή στον buffer.

Αυτή η προσέγγιση διασφαλίζει ότι τα νήματα παραγωγού και καταναλωτή δεν παρεμβαίνουν μεταξύ τους κατά την πρόσβαση στον κοινόχρηστο πόρο. Οι semaphores και ο mutex χρησιμοποιούνται για να συγχρονίσουν την πρόσβαση στον buffer και να αποτρέψουν τυχόν race conditions.

```
62  // producer
63  void *producer(void* rank) {
64      for (int i = 0; i < 10; i++) {
65          do_sleep(2);
66
67          sem_wait(&empty);
68          pthread_mutex_lock(&mut);
69
70          // produce next item
71          item it = produce_item();
72          printf("produced --> item %3d, %s\n", it.num, it.str);
73
74          // save item
75          set_item(in, it);
76          in = (in + 1) % BUFFER_SIZE;
77          pthread_mutex_unlock(&mut);
78          sem_post(&full);
79      }
80
81      return NULL;
82  }
```

```
84
85  // consumer
86  void *consumer(void* rank) {
87      for (int i = 0; i < 10; i++) {
88          do_sleep(2);
89
90          // get item
91          sem_wait(&full);
92          pthread_mutex_lock(&mut);
93          item it = get_item(out);
94          out = (out + 1) % BUFFER_SIZE;
95          pthread_mutex_unlock(&mut);
96          sem_post(&empty);
97
98          // consume item
99          printf("-------- consumed -------> item %3d, %s\n", it
100     }
101
102     return NULL;
103 }
```