ΑΣΚΗΣΗ 3 – Πρόβλημα παραγωγών – καταναλωτών

Ζωγράφου Μαρία-Νίκη

ΑΜ: 1096060

Υλοποίηση με Threads

Στόχος: Υλοποίηση του προβλήματος του παραγωγού-καταναλωτή με χρήση threads, σεμαφόρων και mutex, επιτυγχάνοντας τον συγχρονισμό μεταξύ των threads παραγωγής και κατανάλωσης δεδομένων σε έναν κυκλικό buffer σταθερού μεγέθους.

Το πρόγραμμα περιλαμβάνει δύο τύπους threads: παραγωγούς (producers) και καταναλωτές (consumers). Οι παραγωγοί δημιουργούν δεδομένα και τα τοποθετούν στον κυκλικό buffer, ενώ οι καταναλωτές αφαιρούν δεδομένα από τον buffer. Η πρόσβαση στον buffer ελέγχεται με σεμαφόρους και mutex, ώστε να αποφεύγονται προβλήματα όπως ο ανταγωνισμός (race conditions) και οι ανεπιθύμητες συμπεριφορές.

Ο κυκλικός buffer, με σταθερό μέγεθος **BUFFER_SIZE**, επιτρέπει την αποθήκευση δεδομένων σε έναν πίνακα, όπου οι δείκτες εισαγωγής (in) και εξαγωγής (out) ενημερώνονται κυκλικά με την εντολή index = (index + 1) % **BUFFER_SIZE**. Με αυτόν τον τρόπο, όταν ο δείκτης φτάνει στο τέλος του πίνακα, επιστρέφει στην αρχή, εξασφαλίζοντας την επαναχρησιμοποίηση του διαθέσιμου χώρου.

Η λογική συγχρονισμού βασίζεται σε δύο σεμαφόρους:

Ο **empty** υποδεικνύει τον αριθμό των διαθέσιμων κενών θέσεων στον buffer. Αρχικά έχει τιμή **BUFFER_SIZE** (όλες οι θέσεις είναι κενές). Όταν ένας παραγωγός τοποθετεί δεδομένα, η τιμή του μειώνεται, ενώ όταν ένας καταναλωτής αφαιρεί δεδομένα, αυξάνεται.

Ο **full** υποδεικνύει τον αριθμό των γεμάτων θέσεων στον buffer. Ξεκινά από το 0 (ο buffer είναι άδειος). Αυξάνεται από τους παραγωγούς και μειώνεται από τους καταναλωτές.

Το mutex χρησιμοποιείται για να διασφαλιστεί ότι μόνο ένα thread τη φορά έχει πρόσβαση στον buffer, αποτρέποντας τον ταυτόχρονο ανταγωνισμό.

Παραγωγός (Producer):

Δημιουργεί ένα τυχαίο αντικείμενο και περιμένει αν δεν υπάρχουν διαθέσιμες κενές θέσεις (sem_wait(&empty)). Αν empty=0, η τιμή θα γίνει -1 οπότε θα βρεθεί σε ουρά αναμονής το νήμα του παραγωγού. Αλλιώς (empty>0), αποκτά αποκλειστική πρόσβαση στον buffer με pthread_mutex_lock(&mutex), τοποθετεί το αντικείμενο στην τρέχουσα θέση του δείκτη in, και ενημερώνει τη θέση in κυκλικά. Τέλος απελευθερώνει τον buffer (pthread_mutex_unlock(&mutex)) και αυξάνει τον σεμαφόρο full για να ειδοποιήσει ότι υπάρχει ένα νέο γεμάτο slot.

Καταναλωτής (Consumer):

Περιμένει αν δεν υπάρχουν διαθέσιμα γεμάτα slots (sem_wait(&full)). Αν full>=1 αποκτά αποκλειστική πρόσβαση στον buffer με pthread_mutex_lock(&mutex), αφαιρεί το αντικείμενο από την τρέχουσα θέση του δείκτη out, μηδενίζει τη θέση και ενημερώνει τη

θέση out κυκλικά. Απελευθερώνει τον buffer (pthread_mutex_unlock(&mutex)) και αυξάνει τον σεμαφόρο empty για να ειδοποιήσει ότι υπάρχει ένα νέο κενό slot.

Το πρόγραμμα εκτυπώνει κάθε ενέργεια των παραγωγών και των καταναλωτών, καθώς και την τρέχουσα κατάσταση του buffer μετά από κάθε εισαγωγή ή εξαγωγή (buffer_printer).

Το πρόγραμμα εκτελείται επ' αόριστον. Οι καθυστερήσεις (sleep(2)) προσομοιώνουν πραγματικές συνθήκες όπου η παραγωγή και η κατανάλωση απαιτούν χρόνο.

Αποτελέσματα

Τα αποτελέσματα δείχνουν τη σωστή λειτουργία του κυκλικού buffer, όπου οι παραγωγοί εισάγουν αντικείμενα και οι καταναλωτές τα αφαιρούν συντονισμένα. Ο buffer ποτέ δεν γεμίζει ή αδειάζει πέρα από τα όριά του, αποδεικνύοντας την αποτελεσματικότητα της συγχρονισμένης πρόσβασης με σεμαφόρους και mutex. Όπως και στην άσκηση 1 υπάρχει συνεργασία νημάτων πάνω σε κοινή μνήμη.

```
manya@debian:/media/sf_OS3$ gcc -o pthread ask3.c
manya@debian:/media/sf_OS3$ ./pthread
Producer 140024162698944: Inserted item 83 at 0
Buffer: 83 0 0 0 0
Consumer 140024145913536: Removed item 83 from 0
Buffer: 0 0 0 0 0
Producer 140024171091648: Inserted item 86 at 1
Buffer: 0 86 0 0 0
Consumer 140024154306240: Removed item 86 from 1
Buffer: 0 0 0 0 0
Producer 140024162698944: Inserted item 77 at 2
Buffer: 0 0 77 0 0
Consumer 140024145913536: Removed item 77 from 2
Buffer: 0 0 0 0 0
Producer 140024171091648: Inserted item 15 at 3
Buffer: 0 0 0 15 0
Consumer 140024154306240: Removed item 15 from 3
Buffer: 0 0 0 0 0
Producer 140024162698944: Inserted item 93 at 4
Buffer: 0 0 0 0 93
Producer 140024171091648: Inserted item 35 at 0
Buffer: 35 0 0 0 93
Consumer 140024154306240: Removed item 93 from 4
Buffer: 35 0 0 0 0
Consumer 140024145913536: Removed item 35 from 0
Buffer: 0 0 0 0 0
Producer 140024162698944: Inserted item 86 at 1
Buffer: 0 86 0 0 0
Producer 140024171091648: Inserted item 92 at 2
Buffer: 0 86 92 0 0
Consumer 140024145913536: Removed item 86 from 1
Buffer: 0 0 92 0 0
Consumer 140024154306240: Removed item 92 from 2
Buffer: 0 0 0 0 0
^C
manya@debian:/media/sf_OS3$
```

Υλοποίηση με Processes

Για τον συγχρονισμό των διεργασιών θα χρησιμοποιηθούν σεμαφόροι και μοιραζόμενη μνήμη.

Shared Memory

Η μοιραζόμενη μνήμη επιτρέπει στις διεργασίες να μοιράζονται δεδομένα, τα οποία είναι προσβάσιμα μέσω κοινών δεικτών (`buffer`, `in`, `out`). Η δημιουργία της μοιραζόμενης μνήμης πραγματοποιείται με την κλήση `shmget`, ενώ οι δείκτες συνδέονται σε αυτή μέσω `shmat`. Οι δείκτες `*in` και `*out` χρησιμοποιούνται για να παρακολουθούν τις θέσεις εισαγωγής και εξαγωγής στον buffer και φυλάνε τους ακεραίους που στην υλοποίηση με threads φυλούσαμε στις μεταβλητές in, out. Αρχικοποίησή των τιμώς τους γίνεται με την εντολή `*in = 0` και `*out = 0`.

Semaphores

Χρησιμοποιούνται τρεις σεμαφόροι για τον συγχρονισμό όπως και προηγουμένως:

- **`empty`**: Υποδεικνύει τον αριθμό των κενών θέσεων στον buffer και αρχικοποιείται στο μέγεθος του buffer.
- **`full`**: Υποδεικνύει τον αριθμό των γεμάτων θέσεων στον buffer και αρχικοποιείται στο 0.
- **`mutex`**: Διασφαλίζει αποκλειστική πρόσβαση στον buffer για να αποτρέπονται ταυτόχρονες τροποποιήσεις από διαφορετικές διεργασίες. Σεμαφόρος με αρχική τιμή 1.

Οι σεμαφόροι είναι **named semaphores**, που μπορούν να χρησιμοποιηθούν από διαφορετικές διεργασίες. Αρχικοποίηση γίνεται με την `sem_open`. Η `sem_wait` και η `sem_post` χρησιμοποιούνται για επεξεργασία των τιμών των σεμαφόρων.

Mutex

Καθώς έχει αρχική τιμή 1 μόνο μια διεργασία μπορεί να εκτελεί εκείνη τη στιγμή το κομμάτι κώδικα που περικλείει το Mutex – γίνεται δηλαδή αμοιβαίος αποκλεισμός της κρίσιμης περιοχής.

Παραγωγοί

Κάθε παραγωγός δημιουργεί ένα τυχαίο στοιχείο και το τοποθετεί στον buffer. Χρησιμοποιεί τον σεμαφόρο `empty` για να περιμένει αν δεν υπάρχουν κενές θέσεις και τον σεμαφόρο `mutex` για να αποκτήσει αποκλειστική πρόσβαση στον buffer. Αφού τοποθετήσει το στοιχείο, ενημερώνει τον δείκτη `*in` και αυξάνει τον σεμαφόρο `full`, ώστε να ειδοποιηθούν οι καταναλωτές ότι υπάρχει νέο στοιχείο.

Καταναλωτές

Κάθε καταναλωτής αφαιρεί ένα στοιχείο από τον buffer. Χρησιμοποιεί τον σεμαφόρο `full` για να περιμένει αν δεν υπάρχουν διαθέσιμα στοιχεία και τον σεμαφόρο `mutex` για να αποκτήσει αποκλειστική πρόσβαση στον buffer. Αφαιρεί το στοιχείο, ενημερώνει τον δείκτη `*out` και αυξάνει τον σεμαφόρο `empty`, ώστε να ειδοποιηθούν οι παραγωγοί ότι υπάρχει νέα κενή θέση.

Αποτελέσματα

Το πρόγραμμα χρησιμοποιεί διεργασίες και μοιραζόμενη μνήμη για την επίλυση του προβλήματος του παραγωγού-καταναλωτή. Σε αντίθεση με πριν έχει 3 named_semaphores και δεν μπορεί να χρησιμοποιήσει p_thread mutex. Επιπλέον τα νήματα έχουν ήδη πρόσβαση στην ίδια κοινή μνήμη μιας διεργασίας, ενώ τώρα πρέπει να την ορίσουμε εμείς την κοινή μνήμη με `shmget`.

Σαν αποτέλεσμα ο buffer παρουσιάζει την ίδια συμπεριφορά με πριν:

```
manya@debian:/media/sf_OS3$ gcc -o pthread ask3p.c
manya@debian:/media/sf_OS3$ ./pthread
Producer 3367: Inserted item 1625 at 0
Buffer: 1625 0 0 0 0
Consumer 3370: Removed item 1625 from 0
Buffer: 0 0 0 0 0
Producer 3366: Inserted item 2305 at 1
Buffer: 0 2305 0 0 0
Consumer 3369: Removed item 2305 from 1
Buffer: 0 0 0 0 0
Producer 3365: Inserted item 3303 at 2
Buffer: 0 0 3303 0 0
Consumer 3368: Removed item 3303 from 2
Buffer: 0 0 0 0 0
Producer 3364: Inserted item 1255 at 3
Buffer: 0 0 0 1255 0
Producer 3367: Inserted item 2440 at 4
Buffer: 0 0 0 1255 2440
Consumer 3370: Removed item 1255 from 3
Buffer: 0 0 0 0 2440
Producer 3366: Inserted item 1528 at 0
Buffer: 1528 0 0 0 2440
Consumer 3369: Removed item 2440 from 4
Buffer: 1528 0 0 0 0
Producer 3365: Inserted item 766 at 1
Buffer: 1528 766 0 0 0
Consumer 3368: Removed item 1528 from 0
Buffer: 0 766 0 0 0
Producer 3364: Inserted item 154 at 2
Buffer: 0 766 154 0 0
Producer 3367: Inserted item 856 at 3
Buffer: 0 766 154 856 0
Consumer 3370: Removed item 766 from 1
Buffer: 0 0 154 856 0
Producer 3366: Inserted item 2151 at 4
Buffer: 0 0 154 856 2151
Consumer 3369: Removed item 154 from 2
Buffer: 0 0 0 856 2151
Producer 3365: Inserted item 377 at 0
Buffer: 377 0 0 856 2151
Consumer 3368: Removed item 856 from 3
Buffer: 377 0 0 0 2151
```