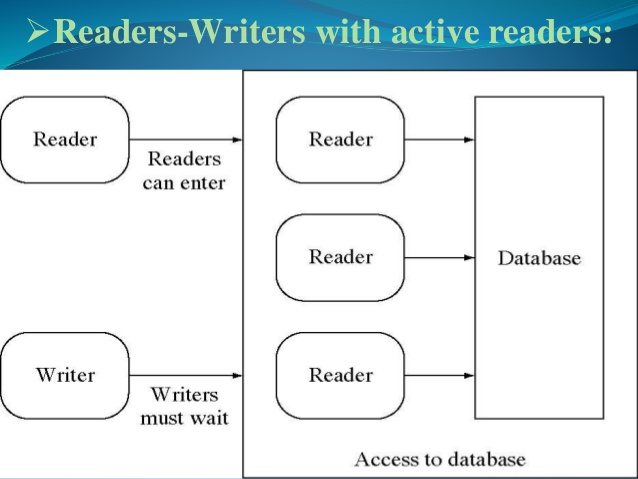
5/6/2019

|  |
| --- |
| Καθηγητής κ.Γκόγκος Χρήστος | Μπουραντά Μαρία ΑΜ:15369 |

|  |  |
| --- | --- |
| τμημα πληροφορικης και τηλεπικοινωνιων | εργασια για το μαθημα κατανεμημενα και παραλληλα συστηματα |

Πρόβλημα αναγνωστών-συγγραφέων | (Εισαγωγή και Λύση Προτιμήσεων Αναγνώρισης)

Θεωρούμε ένα σενάριο όπου έχουμε ένα αρχείο που μοιράζεται μεταξύ πολλών διεργασιών.

Αν ένας από μια διεργασία προσπαθήσει να επεξεργαστεί (να γράψει) το αρχείο, καμία άλλη διεργασία δεν πρέπει να διαβάζει ή να γράφει ταυτόχρονα, διαφορετικά οι αλλαγές δεν θα είναι ορατές ή τα δεδομένα θα είναι εσφαλμένα.

Ωστόσο, εάν κάποια διεργασία (process) διαβάζει το αρχείο, τότε μπορούν να το διαβάσουν και άλλες ταυτόχρονα.

Στο επίπεδο του λειτουργικό συστήματος (Operating System) το πρόβλημα αυτό είναι γνωστό ως “το πρόβλημα των αναγνωστών-συγγραφέων”.

Ένα σύνολο δεδομένων μοιράζεται μεταξύ πολλών διεργασιών, κάποιες από τις οποίες διαβάζουν τα δεδομένα (διεργασίες-αναγνώστες) και κάποιες γράφουν τα δεδομένα (διεργασίες-συγγραφείς).

Μόλις μια διεργασία-συγγραφέας είναι έτοιμη, εκτελεί την εγγραφή της. Μόνο ένας συγγραφέας μπορεί να γράψει κάθε φορά. Καθώς μια διεργασία-συγγραφέας γράφει, καμία άλλη διεργασία δεν μπορεί να διαβάσει.

Αντίστοιχα, αν τουλάχιστον μία διεργασία-αναγνώστης διαβάζει, καμία άλλη διεργασία-συγγραφέας δεν μπορεί να γράψει. Εντούτοις, μπορούν να διαβάζουν κι άλλες διεργασίες-αναγνώστες να διαβάζουν τα δεδομένα ταυτόχρονα. Σημείωση: οι διεργασίες-αναγνώστες δεν επιτρέπεται να γράφουν δεδομένα (εξορισμού).

Λύση:

Καθώς πολλαπλές διεργασίες-αναγνώστες μπορούν να διαβάζουν ταυτόχρονα, η διεργασία-αναγνώστης έχει προτεραιότητα έναντι της διεργασίας-συγγραφέα.

Εδώ η προτεραιότητα σημαίνει, ότι καμία διεργασία-αναγνώστης δεν χρειάζεται να περιμένει για να προχωρήσει στην ανάγνωση του αρχείου.

Για την υλοποίηση χρησιμοποιούμε δύο σημαφόρους (semaphores), έστω mutex και wrt, και μια μεταβλητή τύπου ακεραίου (integer), έστω readcnt.

* semaphore mutex, wrt; // Το mutex του semaphore χρησιμοποιείται για την εξασφάλιση αμοιβαίου αποκλεισμού όταν ενημερώνεται το readcnt δηλ. όταν οποιοσδήποτε αναγνώστης εισέρχεται ή εξέρχεται από την κρίσιμη ενότητα και το semaphore wrt χρησιμοποιείται από τους αναγνώστες και τους συγγραφείς
* int readcnt; // readcnt δηλώνει τον αριθμό των διαδικασιών που εκτελούν την ανάγνωση στην κρίσιμη ενότητα, αρχικά 0

Λειτουργίες για semaphore:

- wait (): μειώνει την τιμή του σηματοφόρου.

- σήμα (): αυξάνει την τιμή του σηματοφόρου.

Διαδικασία συγγραφέα:

Ο συγγραφέας ζητά την εισαγωγή στην κρίσιμη ενότητα.

Αν επιτρέπεται, δηλαδή wait (), δίνει μια πραγματική τιμή, εισάγει και εκτελεί την εγγραφή. Εάν δεν επιτρέπεται, συνεχίζει να περιμένει.

Βγαίνει από το κρίσιμο τμήμα.

Do{

// αιτήσεις συγγραφέα για κρίσιμη ενότητα

wait(wrt);

// εκτελεί την εγγραφή

    // αφήνει το κρίσιμο τμήμα

    σήμα (wrt);

signal(wrt);

} while(true);

Διαδικασία ανάγνωσης:

Ο αναγνώστης ζητά την εισαγωγή στην κρίσιμη ενότητα.

Αν επιτρέπεται:

αυξάνει τον αριθμό του αριθμού των αναγνωστών στο εσωτερικό του κρίσιμου τμήματος. Εάν αυτός ο αναγνώστης είναι ο πρώτος αναγνώστης που εισέρχεται, κλειδώνει το wrt semaphore για να περιορίσει την είσοδο των συγγραφέων αν υπάρχει κάποιος αναγνώστης μέσα.

Στη συνέχεια, σηματοδοτεί το mutex καθώς οποιοσδήποτε άλλος αναγνώστης μπορεί να εισέλθει ενώ άλλοι διαβάζουν ήδη.

Μετά την ανάγνωση, εξέρχεται από το κρίσιμο τμήμα. Όταν βγαίνει, ελέγχει εάν δεν υπάρχει κανένας αναγνώστης μέσα, σηματοδοτεί το σηματοφόρο "wrt" όπως τώρα, ο συγγραφέας μπορεί να εισέλθει στο κρίσιμο τμήμα.

Εάν δεν επιτρέπεται, συνεχίζει να περιμένει.

do {

// ο συγγραφές θέλει αν εισέλθει στο κρίσιμο τμήμα

wait(mutex);

// ο αριθμός των χρηστών θα ανεβεί κατά ένα

readcnt++;

// υπάρχει τουλάχιστον ένας αναγνώστης στο κρίσιμο τμήμα

   // αυτό εξασφαλίζει ότι κανένας συγγραφέας δεν μπορεί να εισέλθει εάν υπάρχει ακόμη και ένας αναγνώστης

   // έτσι προτιμούμε τους αναγνώστες εδώ

if (readcnt==1)

wait(wrt);

// άλλοι αναγνώστες μπορούν να εισέλθουν ενώ αυτός ο τρέχων αναγνώστης είναι μέσα

   // το κρίσιμο τμήμα

signal(mutex);

// ο τρέχων αναγνώστης πραγματοποιεί την ανάγνωση εδώ

wait(mutex);   // ο αναγνώστης θέλει να φύγει

readcnt--;

// δηλαδή, δεν υπάρχει κανένας αναγνώστης στην κρίσιμη ενότητα,

if (readcnt == 0)

signal(wrt);         //οι συγγραφείς μπορεί αν εισέλθουν

signal(mutex); // οι αναγνώστες φεύγουν

} while(true);

Έτσι, το σηματοφόρο «wrt» τίθεται σε αναμονή τόσο για τους αναγνώστες όσο και για τους συγγραφείς με τέτοιο τρόπο ώστε να δίνεται προτίμηση στους αναγνώστες, αν υπάρχουν συγγραφείς. Έτσι, κανένας αναγνώστης δεν περιμένει απλά επειδή ένας συγγραφέας έχει ζητήσει να εισέλθει στο κρίσιμο τμήμα.

Άρθρο που συνεισέφερε η Ekta Goel. Παρακαλούμε γράψτε σχόλια αν βρείτε κάτι λανθασμένο ή θέλετε να μοιραστείτε περισσότερες πληροφορίες σχετικά με το θέμα που συζητήσαμε παραπάνω.

Το πρόβλημα των αναγνωστών και των συγγραφέων είναι χρήσιμο για διαδικασίες μοντελοποίησης που ανταγωνίζονται για έναν περιορισμένο κοινόχρηστο πόρο. Ένα πρακτικό παράδειγμα ενός προβλήματος αναγνωστών και συγγραφέων είναι ένα σύστημα κράτησης αεροπορικών εταιρειών που αποτελείται από μια τεράστια βάση δεδομένων με πολλές διαδικασίες που διαβάζουν και γράφουν τα δεδομένα. Η ανάγνωση πληροφοριών από τη βάση δεδομένων δεν θα προκαλέσει πρόβλημα, αφού δεν έχουν αλλάξει δεδομένα. Το πρόβλημα έγκειται στη σύνταξη πληροφοριών στη βάση δεδομένων. Εάν δεν υπάρχουν περιορισμοί στην πρόσβαση στη βάση δεδομένων, τα δεδομένα ενδέχεται να αλλάξουν ανά πάσα στιγμή. Μέχρι τη στιγμή που μια διαδικασία ανάγνωσης εμφανίζει το αποτέλεσμα μιας αίτησης για πληροφορίες στον χρήστη, τα πραγματικά δεδομένα στη βάση δεδομένων μπορεί να έχουν αλλάξει. Τι γίνεται αν, για παράδειγμα, μια διαδικασία διαβάζει τον αριθμό των διαθέσιμων θέσεων σε μια πτήση, βρίσκει αξία μία και την αναφέρει στον πελάτη. Πριν ο πελάτης έχει την ευκαιρία να κάνει την κράτησή του, μια άλλη διαδικασία κάνει κράτηση για έναν άλλο πελάτη, αλλάζοντας τον αριθμό των διαθέσιμων θέσεων στο μηδέν.

Στις επόμενες σελίδες, θα εξετάσουμε λύσεις σε αυτό το πρόβλημα χρησιμοποιώντας σηματοφόρα, οθόνες και μηνύματα.

Στις επόμενες τρεις ενότητες, την Orange Line, θα συζητήσουμε λύσεις σε αυτό το πρόβλημα χρησιμοποιώντας καθεμία από τις τρεις μεθόδους που αναφέρονται στην ενότητα Τύποι λύσεων:

Semaphores

Monitors

Message Passing

Σηματοφόρα μπορούν να χρησιμοποιηθούν για να περιορίσουν την πρόσβαση στη βάση δεδομένων υπό ορισμένες προϋποθέσεις. Σε αυτό το παράδειγμα, τα σηματοφόρα χρησιμοποιούνται για να εμποδίσουν τις διαδικασίες γραφής να αλλάζουν πληροφορίες στη βάση δεδομένων ενώ άλλες διαδικασίες διαβάζουν από τη βάση δεδομένων.

**Semaphore** mutex = 1; // Ελέγχει την πρόσβαση στην καταμέτρηση των αναγνωστών

**Semaphore** db = 1; // Ελέγχει την πρόσβαση στη βάση δεδομένων

**int** reader\_count; // Ο αριθμός των διαδικασιών ανάγνωσης που έχουν πρόσβαση στα δεδομένα

Reader ()

{

**While** (TRUE) { // loop

Down (&mutex); // πρόσβαση στο reader count

reader\_count = reader\_count + 1; // αύξηση του αναγνωριστικού\_στοιχείου

If (reader\_count == 1)

Down (&db); // αν αυτή είναι η πρώτη διαδικασία για να διαβάσετε τη βάση δεδομένων,

// μέσω μια διαδικασίας γραφής

 (& mutex)? // επιτρέπουν σε άλλες διεργασίες να έχουν πρόσβαση στο reader\_count

     read\_db (); // διαβάστε τη βάση δεδομένων

     Down (& mutex); // αποκτήστε πρόσβαση στο reader\_count

     reader\_count = νούμερο αναγνώστη - 1; // decrement reader\_count

     if (reader\_count == 0)

         up (& db); // εάν δεν υπάρχουν περισσότερες διαδικασίες που διαβάζονται από το

                                            // βάση δεδομένων, επιτρέπουν τη διαδικασία εγγραφής για την πρόσβαση στα δεδομένα

     (& mutex)? // επιτρέπουν σε άλλες διεργασίες να αποκτούν πρόσβαση σε reader\_countuse\_data ();

                                            // χρησιμοποιήστε τα δεδομένα που διαβάζονται από τη βάση δεδομένων (μη κρίσιμα)

}}

Writer ()

{

  While (TRUE) {// των για πάντα

     create\_data (); // Δημιουργία δεδομένων για εισαγωγή σε βάση δεδομένων (μη κρίσιμη)

     down(& db)? // αποκτήστε πρόσβαση στη βάση δεδομένων

     write\_db (); // γράψτε πληροφορίες στη βάση δεδομένων

     up (& db); // απελευθέρωση της αποκλειστικής πρόσβασης στη βάση δεδομένων

}}

Οι οθόνες μπορούν να χρησιμοποιηθούν για να περιορίσουν την πρόσβαση στη βάση δεδομένων. Σε αυτό το παράδειγμα, οι λειτουργίες ανάγνωσης και εγγραφής που χρησιμοποιούνται από διαδικασίες που έχουν πρόσβαση στη βάση δεδομένων βρίσκονται σε μια οθόνη που ονομάζεται ReadersWriters. Εάν μια διαδικασία θέλει να γράψει στη βάση δεδομένων, πρέπει να καλέσει τη λειτουργία writeDatabase. Εάν μια διαδικασία θέλει να διαβάσει από τη βάση δεδομένων, πρέπει να καλέσει τη λειτουργία readDatabase.

Θυμηθείτε ότι οι οθόνες χρησιμοποιούν τα αρχέτυπα Wait and Signal για να θέσουν τις διαδικασίες σε ύπνο και να ξυπνήσουν ξανά. Στο writeDatabase, η διαδικασία κλήσης θα τεθεί σε ύπνο αν ο αριθμός των διαδικασιών ανάγνωσης, που αποθηκεύονται στον αριθμό μεταβλητών, δεν είναι μηδέν. Μετά την έξοδο από τη λειτουργία readDatabase, οι διαδικασίες ανάγνωσης ελέγχουν αν θα ξυπνήσουν μια διαδικασία ύπνου.

  monitor ReadersWriters

    condition OKtoWrite, OKtoRead;

    int ReaderCount = 0;

    Boolean busy = false.

  procedure StartRead ()

  {

    if (busy) // αν η βάση δεδομένων δεν είναι ελεύθερη, μπλοκάρει

      OKtoRead.wait;

    ReaderCount ++; // reader Reader Reader

    OKtoRead.signal ();

  }}

  procedure EndRead ()

  {

     ReaderCount--; // ανάγνωση Reader Reader

     if (ReaderCount == 0)

        OKtoWrite.signal ();

  }}

  procedure StartWrite ()

  {

    if (busy || ReaderCount! = 0)

       OKtoWrite.wait ();

    busy = true;

  }}

  procedure EndWrite ()

  {

    busy = false;

    if (OKtoRead.Queue)

      OKtoRead.signal ();

    else

      OKtoWrite.signal ();

   }}

  Reader()

  {

     while (TRUE) // loop για πάντα

     {

        ReadersWriters.StartRead ();

  readDatabase (); // κλήση λειτουργίας readDatabase στην οθόνη

        ReadersWriters.EndRead ();

     }}

  }}

  Writer()

  {

     while (TRUE) // loop για πάντα

     {

 make\_data (& info); // δημιουργήστε δεδομένα για εγγραφή

        ReaderWriters.StartWrite ();

writeDatabase (); // κλήση writeDatabase λειτουργία στην οθόνη

        ReadersWriters.EndWrite ();

     }}

  }}

Σε αυτή τη λύση στο πρόβλημα Readers / Writers, το Passing Message χρησιμοποιείται για να εμποδίσει τη διαδικασία εγγραφής από τη γραφή στη βάση δεδομένων ενώ η άλλη διαδικασία διαβάζει ή γράφει. Η ακόλουθη λύση υποθέτει ότι οι αναγνώστες και οι συγγραφείς στέλνουν τα αιτήματά τους σε μια διαδικασία διακομιστή βάσης δεδομένων που διασφαλίζει την κατοχή της παραπάνω ιδιότητας.

Reader()

{

**while** (TRUE) {

send (server, ReadRequest);

receive (server, value);

display (value);

}

}

Writer()

{

**while** (TRUE) {

create\_data (&value);

send (server, WriteRequest, value);

receive (server, WriteOk);

}

}

Βιβλιογραφία

<http://denninginstitute.com/modules/ipc/aqua/readers.html>

<https://www.geeksforgeeks.org/readers-writers-problem-set-1-introduction-and-readers-preference-solution/>