Ταυτόχρονος Προγραμματισμός - Εργασία 3

Κυριακίδης Χρήστος 3029
Κεστελίδης Φίλιππος 3060

3.1 Αναγνώριση πρώτων αριθμών (με ελεγκτή)

- Στην εργασία αυτή πλέον χειριζόμαστε το διάλογο μεταξύ main και worker με τη χρήση mutexes και conditions
- Οι conditionals συναρτήσεις της βιβλιοθήκης pthread είναι signal-continue (μέχρι το επόμενο mutex unlock), όμως για τα δεδομένα της εργασίας μας καλούμε τη signal και έπειτα κατευθείαν τη wait, υλοποιώντας έμμεσα μια signal-block λογική

Απλή περιγραφή προγράμματος

- Η main διαβάζει έναν αριθμό και περιμένει, σε ένα predicate loop, σήμα από ένα worker ότι είναι διαθέσιμος
- Μόλις ξυπνήσει στέλνει σήμα στο worker ότι έχει διαβάσει τη τιμή και περιμένει ξανά μέχρι να ολοκληρώσει ο worker
- Ο worker παράλληλα, σε επανάληψη στέλνει ένα σήμα ότι είναι διαθέσιμος και περιμένει μέχρι να λάβει σήμα από τη main ότι υπάρχει διαθέσιμη τιμή
- Τότε υπολογίζει αν ο αριθμός είναι πρώτος, στέλνει σήμα στη main ότι ολοκλήρωσε και ξανά από την αρχή

Ψευδοκώδικας

```
Main
while (there's input) {
        while (predicate == false) {
                wait(available);
        }
        signal(read);
        wait(calculate);
}
…
```

```
…
mainDone = true;
for i < number of workers {
        signal(read);
        wait(finish);
}
/* read conditional acts as a way to break out of
the worker while loop */
```

Ψευδοκώδικας

```
Worker
while (1) {
        lock(mutex);
        signal(available);
        predicate = true;
        wait(read);
        predicate = false;
…
        if (mainDone) break;
        calculate prime and print result
        signal(calculate);
        unlock(mutex);
}
signal(finish);
unlock(mutex);
```

3.2 Μονόδρομη γέφυρα με αυτοκίνητα

• Στο πρόγραμμα καλούμαστε να συγχρονίσουμε αυτοκίνητα που θέλουν να περάσουν μια γέφυρα διπλής κατεύθυνσης (αλλα μιας λωρίδας) με την βοήθεια ενός ελεγκτή

• Ο συγχρονισμός αυτός γίνεται με την χρήση δυο conditional variables (left, right) και ενός mutex

• Καθώς όλοι οι enter / exit codes βρίσκονται ανάμεσα σε mutex lock και unlock, δεν υπάρχει καμία περίπτωση για race conditions

• Και η main με την σειρά της περιμένει (με χρήση του δικού της conditional mainEnd) να περάσουν όλα τα αυτοκίνητα για να τερματίσει το πρόγραμμα

Απλή περιγραφή προγράμματος

• Η main κανει generate αυτοκίνητα από τις 2 πλευρές με διάφορα delay

• Όποιο αυτοκίνητο έρθει πρώτο, μπαίνει στην γέφυρα και το ακολουθούν αυτοκίνητα της ίδιας κατεύθυνσης μέχρι να γεμίσει η γέφυρα ή να μην περιμένουν άλλα αυτοκίνητα από την συγκεκριμένη κατεύθυνση **

• Μόλις περάσει και το τελευταίο αυτοκίνητο, δίνεται η προτεραιότητα στην απέναντι μεριά (αν υπάρχουν αυτοκίνητα που περιμένουν αλλιώς στην ίδια κατεύθυνση) και επαναλαμβάνεται μέχρι να τελειώσει το input και έχουν περάσει όλα τα αυτοκίνητα

• Άμα απο καμία πλευρά δεν περιμένουν αυτοκίνητα, αλλά ξέρουμε ότι δεν έχει τελειώσει το input, το πρόγραμμα περιμένει και δίνει την προτεραιότητα στο αυτοκίνητο που θα καταθέσει πρώτο

** στην συγκεκριμένη περίπτωση, άμα καταφθάσουν άλλα αυτοκίνητα της ίδιας κατεύθυνσης, περιμένουν, αφού από την απέναντι πλευρά τα αυτοκίνητα περιμένουν περισσότερη ώρα

Ψευδοκώδικας (Ο κώδικας είναι συμμετρικός αφού πάντα αμα περιμένει από την απέναντι πλευρά, δίνουμε προτεραιότητα για να μην υπάρξει λιμοκτονία κανενός) (το mySide αντικαθιστάται με left ή right, αναλόγως την πλευρά)

```
void car_enter() {
        lock(mtx);
        ++waitingMySide;
        while (bridge full or not my direction) wait(mySide);
        ++carsOnBridge;
        --waitingMySide;
        if (bridge not full) signal(mySide);
        unlock(mtx);
}
```

```
void car_thread() {
        if (not direction) direction = mySide;
        car_enter();
        //car passing and printing enter / exit messages
        car_exit();
}
```

```
void car_exit() {
        lock(mtx);
        –carsOnBridge;
        if (bridge empty and waitingOtherSide > 0) {
                direction = otherSide;
                signal(otherSide);
        }
        else if (bridge empty and waitingMySide > 0) signal(mySidet);
        else if (bridge empty and not waiting any cars and input ended) signal(mainEnd);
        else if (bridge empty and not waiting any cars)   direction = none;
        unlock(mtx);
}
```

3.3 Τρενάκι με επιβάτες

• Στο συγκεκριμένο πρόγραμμα καλούμαστε να συγχρονίσουμε τα threads-επιβάτες με το thread-τρενάκι, προσομοιώνοντας την λογική του τραίνου του λούνα πάρκ

• Με την χρήση ενός monitor με τα κατάλληλα conditional variables (passenger, train, pass_exit, exiting) και ενός mutex, οι επιβάτες «ανεβαίνουν» στο τραινάκι και όταν γεμίσει, τρέχει (οι υπόλοιποι επιβάτες περιμένουν), κατεβαίνουν και μετά το τρενάκι επιτρέπει στους επόμενους επιβάτες να ανέβουν

• Δεν δημιουργούνται race conditions γιατί σε κάθε στιγμή εκτέλεσης του προγράμματος, εκτελείται ο κώδικας είτε ενός από τους επιβάτες είτε του τραίνου χάρη στο monitor

• Όλοι οι επιβάτες κάνουν wait στην αρχή και το τραίνο ξυπνάει σε loop MAX_PASSENGERS

Απλή περιγραφή προγράμματος

• Όταν ξεκινάει την εκτέλεση ένα thread επιβάτη, αμέσως περιμένει (wait στο passengers). Άμα γίνει signal αλλά εκείνη την ώρα “βγαίνουν” από το τρενάκι άλλοι επιβάτες, πάλι περιμενουν (wait στο exiting)

• Το τραίνο κάθε φορά που είναι άδειο (τρέχει από την αρχή το while) κάνει signal MAX_PASSENGERS επιβάτες για να μπουν στο τρένο

• Όταν ένας passenger μπαίνει στο τραίνο, αυξάνει τον αριθμό των επιβατών που είναι μέσα, ελέγχει αν είναι ο τελευταίος που χωράει και περιμένει (wait στο pass_exit). Άν είναι ο τελευταίος, κάνει signal το τρενάκι, το οποίο τρέχει και στην συνέχεια με την σειρά του ξυπνάει τους επιβάτες που περιμένουν να βγουν

• Όταν βγουν όλοι οι επιβάτες, ο τελευταίος ξυπνάει αυτούς που περίμεναν επειδή πήγαν να μπούν όταν άλλοι επιβάτες έβγαιναν από το τρενάκι (signal exiting). Μετά από αυτό τρέχει από την αρχή το while loop του τρένου για να ξυπνήσει επιβάτες που περιμένουν να μπουν στο τρενάκι (signal passenger), και συνεχίζεται το ίδιο για πάντα (σύμφωνα με συγκεκριμένες οδηγίες)

Ψευδοκώδικας

```
void trainThread() {

    int i;

    while (1) {

       lock(mtx);

        for (i = 0; i < t->max_passengers;
i++) signal(passenger);

        wait(train, mtx);

            //running train

        isExiting = 1; //signal that
passengers are exiting

        for (i = 0; i < t->current_passengers;
i++) signal(pass_exit);

       unlock(mtx);

}
```

```
void passengerThread() {

    //enter

    lock(mtx);

    wait(passenger, mtx);

    if (isExiting) { waitingToEnter++;
wait(exiting, mtx); }

    //main

    current_passengers++;

    if (current_passengers ==
max_passengers) signal(train);

    wait(pass_exit, mtx);

    unlock(mtx); [...]
```

```
 [...] //exit

    lock(mtx);

    current_passengers--;

    int i;

    if (current_passengers == 0) {

        isExiting = 0;

        for (i = 0; i < waitingToEnter;
i++) signal(exiting);

        t->waitingToEnter = 0;

    }

    unlock(mtx);

}
```

3.4 Κοινόχρηστες τουαλέτες

• Στο πρόγραμμα καλούμαστε να συγχρονίσουμε ανθρώπους και των δύο φύλων που θέλουν να μπουν σε μια κοινόχρηστη τουαλέτα, χωρίς να είναι άτομα του ίδιου φύλου ταυτόχρονα μέσα στην τουαλέτα

• Ο συγχρονισμός αυτός γίνεται με την χρήση δυο conditional variables (woman, man) και ενός mutex

• Καθώς όλοι οι enter / exit codes βρίσκονται ανάμεσα σε mutex lock και unlock, δεν υπάρχει καμία περίπτωση για race conditions

• Και η main με την σειρά της περιμένει (με χρήση του δικού της conditional mainEnd) να τελειώσουν όλοι οι άνθρωποι για να τερματίσει το πρόγραμμα

Απλή περιγραφή προγράμματος

- Η main κανει generate ανθρώπους των δύο φύλων με διάφορα delay
- Όποιος έρθει πρώτος (άνδρας / γυναίκα), μπαίνει στην τουαλέτα και τον ακολουθούν άτομα του ίδιου φύλου μέχρι να γεμίσει η τουαλέτα ή να μην περιμένουν άλλα άτομα του συγκεκριμένου φύλου
- Μόλις βγει και ο τελευταίος άνδρας, δίνεται η προτεραιότητα ΠΑΝΤΑ στις γυναίκες (αν υπάρχουν, αλλιώς συνεχίζουν να μπαίνουν οι άνδρες)
- Μόλις βγει η τελευταία γυναίκα, επιτρέπει σε άλλες γυναίκες να μπουν και ΜΟΝΟ ΑΝ δεν υπάρχουν άλλες γυναίκες που περιμένουν, παραχωρείται η τουαλέτα στους άνδρες
- Άμα δεν περιμένει κανένας, αλλά ξέρουμε ότι δεν έχει τελειώσει το input, το πρόγραμμα περιμένει και δίνει την προτεραιότητα στο φύλο του πρώτου ανθρώπου που θα καταφθάσει
- Άμα δεν περιμένει κανένας και έχει τελειώσει και το input, το πρόγραμμα τερματίζει

Ψευδοκώδικας (Γυναίκες)

```
void womanEnter() {
    lock(mtx);
    ++waitingWomen;
    while (WC full or men in wc) wait(women, mtx);
    ++peopleInWC;
    --waitingWomen;
    if (WC not full) signal(women);
    unlock(mtx);
}
```

```
void womanThread() {
    if (not any gender in wc) give WC to women;
    womanEnter();
// printing entering and exiting messages.
    womanExit();
}
```

```
void womanExit() {
    lock(mtx);
    --peopleInWC;
    if (WC empty and no women waiting and men waiting) { Give WC to men; signal(men);  }
    else if (WC empty and women waiting) signal(women);
    else if (WC empty and no people waiting and input ended) signal(&mainEnd);
    else if (WC empty and no people waiting)
        give WC to first person to come
    unlock(mtx);
}
```

Ψευδοκώδικας (Άνδρες)

```
void manEnter() {
    lock(mtx);
    ++waitingMen;
    while (WC full or women in wc) wait(men, mtx);
    ++peopleInWC;
    --waitingMen;
    if (WC not full) signal(men);
    unlock(mtx);
}
```

```
void manThread() {
    if (not any gender in wc) give WC to men;
    manEnter();
// printing entering and exiting messages.
    manExit();
}
```

```
void manExit() {
    lock(mtx);
    --peopleInWC;
    if (WC empty and women waiting) { Give WC to women; signal(women);  }
    else if (WC empty and men waiting and no women waiting) signal(men);
    else if (WC empty and no people waiting and input ended) signal(mainEnd);
    else if (WC empty and no people waiting)
        give WC to first person to come;
    unlock(mtx);
}
```