Σετ εργασιών 3
Ομάδα 6

Γιαννούκκος Παναγιώτης 228
Χωροπανίτης Πασχάλης 2453

Άσκηση 3.1 - Σκέψη

- Αρχικά, δημιουργίσαμε ένα struct worker_t όπου αποθηκεύει πληροφορία για την κατάσταση του κάθε worker thread.
- Το main thread φτιάχνει ένα πίνακα με τέτοια structs όσα είναι και τα thread workers και κατα τη δημιουργία του κάθε worker, του δίνουμε σαν παράμετρο το index του σε αυτόν τον πίνακα.
- Ο κάθε worker έχει ένα δικό του condition variable για να μπορεί η main να κάνει signal τον κάθε worker όταν του δίνει δουλειά.

Άσκηση 3.1 - Υλοποίηση

```
mutex mtx; cond main_wait;
```

main:

```
Create workers
while (job exists) {
  lock(&mtx);

  while(worker not available) {
    cond_wait(&main_wait, &mtx);
  }

  /* set worker parameters */
  cond_signal(&worker[pos].wait);
  unlock(&mtx);

}
```

worker:

```
while (1) {
  /* I am available */
  lock(&mtx);
  while (don't have job) {
    cond_wait(&self.wait,&mtx);
  }
  unlock(&mtx);
  if signaled by main, break;
  process assigned job
  cond_signal(&main_wait);
}
cond_signal(&main_wait);
```

Άσκηση 3.2 - Σκέψη

- Αρχικά φτιάξαμε ένα struct για την κατάσταση της γέφυρας και ένα για του αυτοκινήτου.
- Η main δημιουργεί τα αυτοκίνητα και έπειτα περιμένει μέχρι να περάσουν από την γέφυρα.
- Η συνάρτηση car thread καλεί την bridge enter, κοιμάται όσο έχει οριστεί το drive time του αυτοκινήτου και έπειτα καλέι την bridge leave.
- Η συνάρτηση bridge enter ελέγχει από ποια κατεύθυνση έρχονται τα αυτοκίνητα και τα αφήνει να περάσουν ή τα μπλοκάρει ανάλογα με την τρέχουσα κατεύθυνση ή εαν η γέφυρα είναι γεμάτη.
- Η συνάρτηση bridge leave αφυπνίζει τα μπλοκαρισμένα αυτοκίνητα και έαν έχουν περάσει όλα αφυπνίζει τη main για τερματισμό.

Άσκηση 3.2 - Υλοποίηση

```
mutex enter_mtx,leave_mtx,main_mtx; cond blue_q,red_q,main_wait;
```

bridge_enter:

```
lock(&enter_mtx);

while (/*can't enter*/) {
  if(red_dir)
    cond_wait(&red_q,&enter_mtx);
  else
    cond_wait(&blue_q,&enter_mtx);
}

/*enter bridge*/

unlock(&enter_mtx);

}
```

bridge_leave:

```
lock(&leave_mtx);

/* leave bridge and signal
   signal someone from a queue */

if (/*no more cars */) {
  cond_signal(&main_wait);
}

unlock(&leave_mtx);
```

Άσκηση 3.3 - Σκέψη

- Αρχικά φτιάξαμε ένα struct για την κατάσταση του train.
- Η main δημιουργεί το τραίνο και τους passengers. Επειτα περιμένει μέχρι να περάσουν όλοι οι passengers από το τραίνο.
- Η συνάρτηση passenger thread βάζει passengers στο τραίνο και έαν έχει γεμίσει το σηματοδοτεί για να ξεκινήσει μια βόλτα.
- Η συνάρτηση train thread ξεκινά μια βόλτα όταν γεμίσουν οι θέσεις της. Εαν έχουν εξυπηρετηθεί όλοι οι επισκέπτες τότε σηματοδοτεί την main για τερματισμό.

Άσκηση 3.3 - Υλοποίηση

```
mutex mtx,train_mtx;cond queue,on_board,train_wait,main_wait;
```

passenger:

```
lock(&mtx);
while (/*can't enter*/) {
  cond_wait(&queue,&mtx);
}
if(/*train is full) {
  cond_signal(&train_wait);
}
cond_signal(&queue);
cond_wait(&on_board,&mtx);
cond_wait(&on_board,&mtx);
if(passengers)cond_signal(&on_board)
else cond_signal(&queue);
if(!visitors) cond_signal(&train);
unlock(&mtx);
```

train:

```
while(1)
  lock(&train_mtx);
  cond_wait(&train_wait,&train_mtx)
  If(!visitors) break;
  /*Go for a ride*/
  cond_signal(&on_board);
  unlock(&train_mtx);
}
cond_signal(&main_wait);
unlock(&train_mtx);
```

Άσκηση 3.4 - CCR

```
init(R_mtx);
init(R_q1); R_n1 = 0; init(R_q2); R_n2 = 0;
```

```
lock(R_mtx);
while (!lexpr) {
  R_n1++;
  if (R_n2 > 0) { R_n2--; cond_signal(R_q2); }
  cond_wait(&R_q1, &R_mtx);
  R_n2++;
  if (R_n1 > 0) { R_n1--; cond_signal(R_q1); }
  else if (R_n2 >= 2) {R_n2--; cond_signal(R_q2); }
  cond_wait(&R_q2, &R_mtx);
}
body;
if (R_n1 > 0) { R_n1--; cond_signal(R_q1); }
else if (R_n2 > 0) { R_n2--; cond_signal(R_q2); }
unlock(&R_mtx);
```