Τρίτη Εργασία Ταυτόχρονου Προγραμματισμού

Ομάδα: 11η

Μέλη:

Κωφόπουλος – Λυμπέρης Σταμάτης 2548

Παυλίδης Παναγιώτης 2608

Άσκηση 3.1 : Πρώτοι Αριθμοί

Main:

```
Create worker threads;
While(i<numbers){
  lock(mutex);
  if(wait_workers != num_threads){
    wait_main++; cond_wait(main_cond);}
  for(j<num_threads){
    if(workers[j].flag == 0  && j < num && i < num){
      assign job; flag == -1;
      if(wait_workers != 0){ cond_signal(workers_cond);}
    }
  }
  unlock(mutex);
}
lock(mutex);
if(wait_main != 69) {cond_wait(main_cond);}
for(all workers){workers.flag = 666;}
cond_broadcast(workers_cond);
cond_wait(main_cond); unlock(mutex);
destroy mutex and conditions;
```

Primesearch:

```
lock(mutex);
If(last worker && wait_main != 0) {
  wait_main--; cond_signal(main_cond); }
cond_wait(workers_cond);
While(1){
  unlock(mutex);
  if(workers.flag != 0){
    if(terminate){ lock(mutex);
      if(last_worker){ cond_signal(main_cond);}
    unlock(mutex); }
    check if number is prime;}
  else{ lock(mutex); cond_signal(workers_cond);
    unlock(mutex);}
  lock(mutex);
  if(wait_main != 0){ wair_main--; cond_signal(main_cond);}
  if(all workers blocked && all nums finished){
    wait_main = 69; cond_wait(main_cond);}
  cond_wait(workers_cond);
}
```

Άσκηση 3.2 : Στενή Γέφυρα

Global variables
colour_same_move = -1;
colour_opp_move = -1;

Main:
Creates sems;
Randomly assigns the color while creates car-thread;
lock(mutex); cond_wait(main_cond); unlock(mutex);
Destroy conditions and mutex;

Threads_func:

```
lock(mutex);
If(colour == 1) { car_in_move = blue; }
else { car_in_move = red; }
unlock(mutex);
//Depending on color :
Before bridge CS
lock(mutex);
If(colour_op_move != -1) { wait_same_colour++;
  cond_wait(same_colour); }
If(bridge_counter >= specific amount of cars) {
 bridge_counter++; colour_op_move = 0; wait_same_colour++;
 cond_wait(same_colour); colour_op_move = -1; }
Bridge_counter++;  same_in_bridge++;
If(same_in_bridge > bridge_space) { cond_wait(bridge_limit_cond); }
unlock(mutex);
```

```
After bridge Cs
Lock(mutex);
If(same_in_bridge > bridge_space){ cond_signal(bridge_limit_cond);}
Same_in_bridge--;
If(same_in_bridge == 0){
  same_in_move = -1; opp_in_move = 0; bridge_counter = 0;
  if(wait_counter_same != 0 && wait_counter_opp == 0){
    same_in_move = 0; opp_in_move = -1;
    for(i< wait_counter_same && i < CARS_POP){
      cond_signal(same_colour);}
    wait_same_colour = wait_same_colour – i; }
  for(i< wait_opp_colour && i< CARS_POP){cond_signal(opp_colour);}
  wait_opp_colour = wait_opp_colour – i;
}
Unlock(mutex);
//after car exits bridge
Lock(mutex); if (last car) { cond_signal(main_cond); } unlock(mutex);
```

Άσκηση 3.3 : Τρενάκι

Main:

```
Create train and passengers;
cond_wait(main_cond);
Destroy conditions and mutex;
```

Thread_train:

```
While(num_passengers != 0) {
  train_func();
}
```

Train_func:

```
lock(mutex);
If(train_flag == 0) { cond_wait(train_cond); }
train_flag = 0;
Unlock(mutex); sleep(1); lock(mutex);
for(i < train_limit) { cond_signal(passengers_cond);
num_passengers--; }
Unlock(mutex);
```

Thread_passengers:

```
passenger_func();
If(he is the last passenger) {
  signal(main_cond);
}
```

Passengers_func:

```
lock(mutex);
If(passenger_in_train > train_limit) {
  cond_wait(passengers_to_wait_cond); }
waiting_passengers_on_train++;
if(train is full) { train_flag = 1; cond_signal(train_cond);  }
cond_wait(passengers_cond);
waiting_passengers_on_train--; passengers_in_train--;
If(last passenger on the ride) {
 wake_passengers = train_limit/waiting_to_board;
 waiting_to_board = waiting_to_board – train_limit/0;
  for(wake_passengers) { cond_singal(passengers_to_wait);
  unlock(mutex); }
}
```

Άσκηση 3.4 : CCR.h

```
#define CCR_DECLARE(label)
    mutex_##label, mutex_q_##label;
    cond_q1_##label, cond_q2_##label;
    q1_counter_##label, q2_counter_##label;
```

```
#define CCR_ INIT(label)
    mutex_init(mutex_##label); mutex_init(mutex_q_##label);
    cond_init(cond_q1_##label); cond_init(cond_q2_##label);
    q1_counter_##label = 0;
    q2_counter_##label = 0;
```

```
#define CCR_EXEC(label, cond, body)
    mutex_lock(mutex_##label);
    mutex_lock(mutex_q_##label);
    while(!cond){
        q1_counter_##label++;
        if(q2 _counter_##label > 0){ q2 _counter_##label --;
            cond_signal(cond_q2 _##label);}
        else {mutex_unlock(mutex_##label);}
        cond_wait(cond_q1 _##label);
        if(q1 _counter_##label > 0){ q1 _counter_##label--;
            cond_signal(cond_q1_##label);}
        else if(q2 _counter_##label > 0){ q2 _counter_##label--;
            cond_signal(cond_q2 _##label);}
        else {mutex_unlock(mutex _##label);}
        q2 _counter_##label++;
        cond_wait(cond_q2 _##label);
    }
```

```
    body;
    If(q1 _counter_##label > 0){q1 _counter_##label--;
        cond_wait(cond_q1 _##label);}
    else if(q2 _counter_##label > 0){q2 _counter_##label--;
        cond_wait(cond_q2 _##label);}
    else{mutex_unlock(mutex_##label);}
    mutex_unlock(mutex_q _##label);
```

Άσκηση 3.4.1 : Πρώτοι αριθμοί

Main:

```
Create worker threads;
While(i<numbers){
  CCR_EXEC(R, ((wait_workers == num_threads) && (start_main == 1)),
    for(j<num_threads){
      if(workers[j].flag == 0  && j < num && i < num){
        assign job; flag == -1;
      }
    }
    start_main = 0;
  });
CCR_EXEC(R, (wait_main == 69),
  for(all workers){ workers.flag = 666; });
CCR_EXEC(R, start_main == -1, print("Main will terminate")););
```

Primesearch:

```
CCR_EXEC(R, 1, wait_workers ++; if(last worker){start_main = 1;});
While(1){
  CCR_EXEC(R, check_number->flag != 0, wait_workers --;
    if(check_number->flag == 666){ if(last worker){ start_main = -1;}} );
  if(check_number->flag == 666){ return(NULL);}
  check if number is prime;
  CCR_EXEC(R, 1, wait_workers++;
    check_numbe->flag = 0; finished_numbers++;
    if(wait_workers == num_threads) && (finished_numbers == numbers)
      { wait_main = 69;}
    start_main = 1;);
}
```

Άσκηση 3.4.3 : Τρενάκι

Main:

```
Create train and passengers;
CCR_EXEC(R, main_flag == 1,
printf("Terminate"));
```

Thread_train:

```
While(num_passengers != 0) {
  train_func();
}
```

Thread_passengers:

```
passenger_func();
CCR_EXEC(R, ((num_pass == 0)&&
  (pass_on_train == 0), main_flag = 1; );
```

Train_func:

```
CCR_EXEC(R, ((pass_on_train == train_limit)&&(train_flag == 0)),
  do the course;  num_pass -= train_limit; train_flag =1;);
```

Passengers_func:

```
CCR_EXEC(R, ((pass_on_train < train_limit)&&(train_flag == 0)),
  pass_on_train++; );
CCR_EXEC(R, train_flag == 1, pass_on_train--;
  if(pass_on_train == 0){ train_flag = 0;});
```