Akmal Ikhsan 1301184059

Jurnal Praktikum
Sistem Operasi (CII-2H3)

Modul 5: Deadlock

Tujuan

1. Mahasiswa mampu mengatasi deadlock dengan ordering
2. Mahasiswa menguasai banker's algorithm

Catatan

1. Praktikan wajib untuk screenshot setiap langkah yang dikerjakan hingga tampilan output akhir
2. Untuk soal source code, kumpulkan SS-nya saja
3. Praktikan wajib untuk melakukan screenshot lengkap dengan nama root. Contoh : root@username
4 Berikan identitas nama - nim dalam bentuk comment di Source Code
5.Harap kerjakan secara mandiri, jika tidak paham silahkan bertanya kepada Asisten Praktikum masing-masing. Dilarang mengcopy jawaban dan source code dari teman!

Jurnal Praktikum

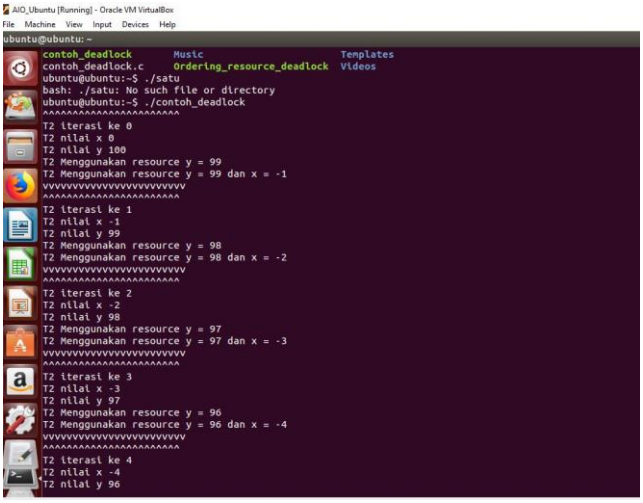
1. Contoh Deadlock
    a. Copy paste source code contoh_deadlock.c
    done
    b. Beri komentar pada setiap baris kode!

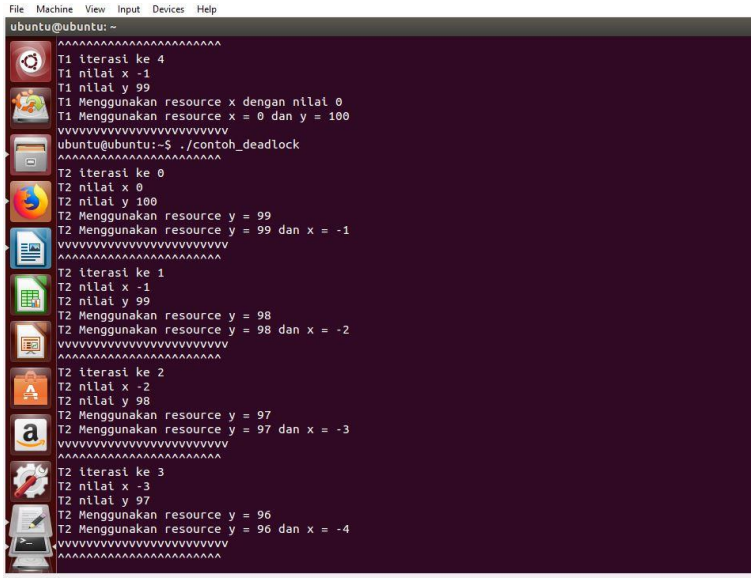
```
int i;
for (i = 0; i < 5; i++) {  //  looping i < 5  increase
  pthread_mutex_lock( & mutex_a);
  printf("^^^^^^^^^^^^^^^^^^^^^^^^\n");
  printf("T1 iterasi ke %d\n", i);
  printf("T1 nilai x %d\n", shared_x); //akan mengoutput nilai x
  printf("T1 nilai y %d\n", shared_y); // output nilai y
  shared_x++;
  printf("T1 Menggunakan resource x dengan nilai %d\n", shared_x);
  pthread_mutex_lock( & mutex_b);
  shared_y++; // increase nilai y
  printf("T1 Menggunakan resource x = %d dan y = %d\n", shared_x, shared_y);
  printf("vvvvvvvvvvvvvvvvvvvvvvvv\n");
  pthread_mutex_unlock( & mutex_b);
  pthread_mutex_unlock( & mutex_a);
 }
}

void * World(void * input) { // prosedure world input
 int i;
 for (i = 0; i < 5; i++) { // looping i < 5 increase
       pthread_mutex_lock( & mutex_b);
       printf("^^^^^^^^^^^^^^^^^^^^^^^^\n");
  printf("T2 iterasi ke %d\n", i);
  printf("T2 nilai x %d\n", shared_x);
  printf("T2 nilai y %d\n", shared_y);
  shared_y--;
  printf("T2 Menggunakan resource y = %d\n", shared_y); // output nilai y
  pthread_mutex_lock( & mutex_a);
  shared_x--;
  printf("T2 Menggunakan resource y = %d dan x = %d\n", shared_y, shared_x); // output resource y dan x
  printf("vvvvvvvvvvvvvvvvvvvvvvvv\n");
```

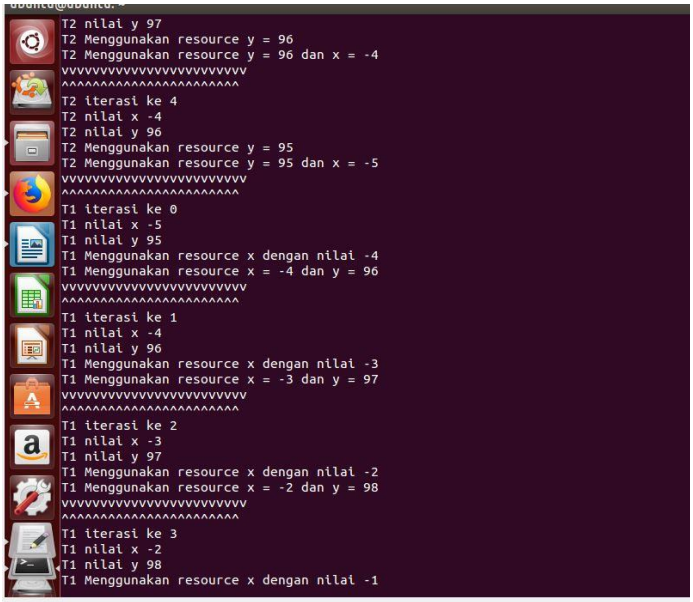
    c. Apa yang dilakukan oleh thread Hello?
    Thread Hello melakukan increament pada nilai x dan nilai y (shared_x dan Shared_y)

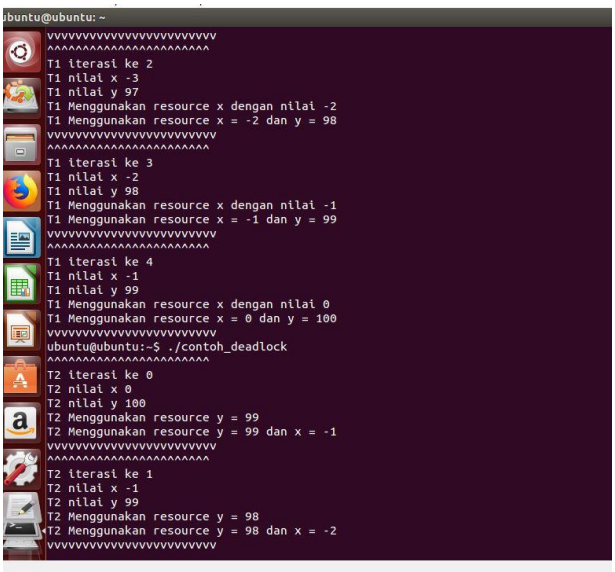
d. Apa yang dilakukan oleh thread World?
   Melakukan Descreament pada nilai x dan nilai y (shared_x dan shared_y)
e. Mutex apa yang melindungi resource shared_x?
   Mutex a
f. Mutex apa yang melindungi resource shared_y?
   Mutex b
g. Compile kode tersebut! Jalankan program tersebut minimal 10 kali! Amati hasilnya!

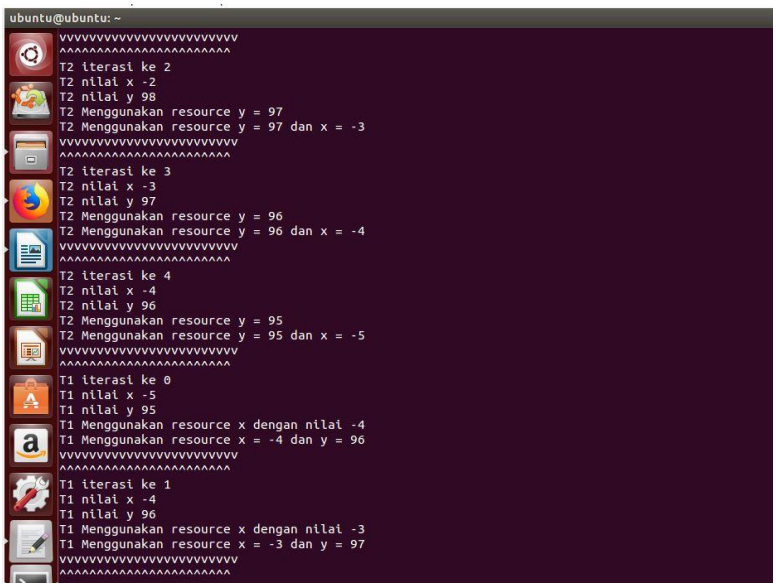


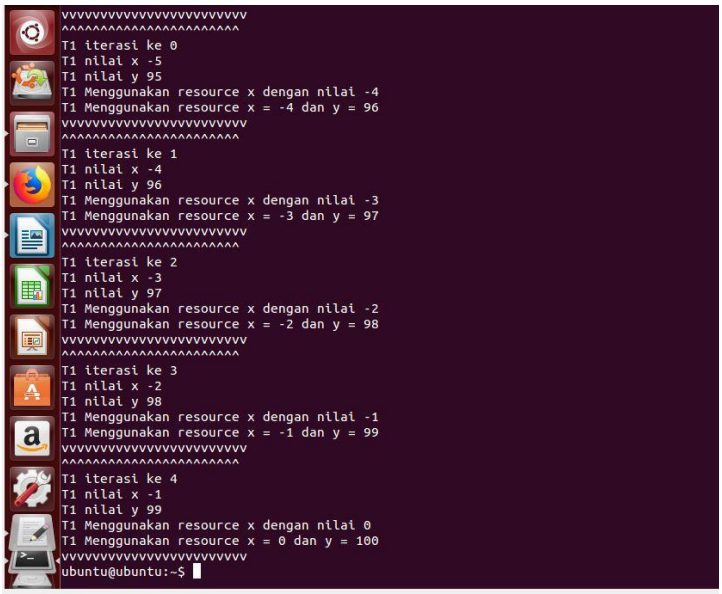


Akmal Ikhsan 1301184059



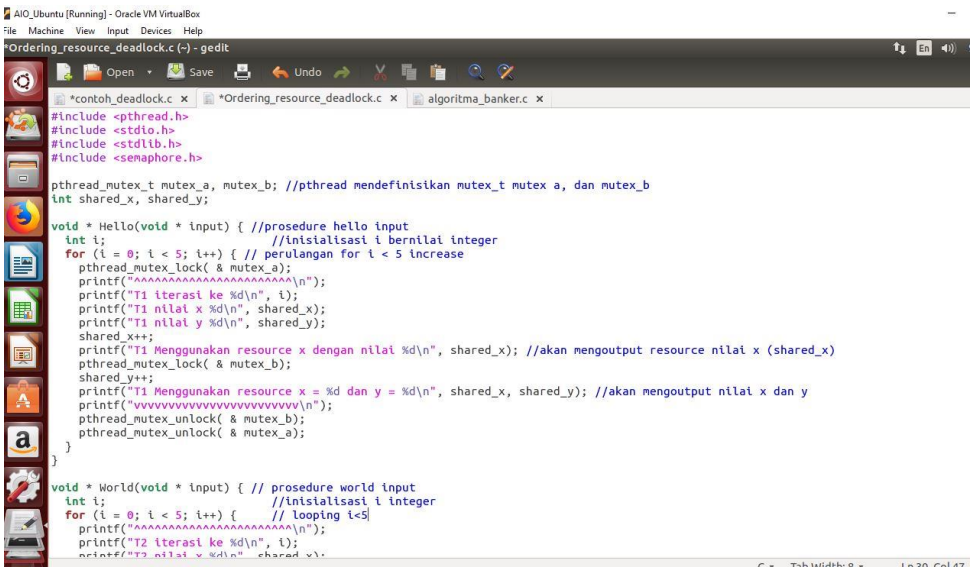






h. Mengapa program mengalami deadlock? Jelaskan!
   Karena ketika T2 sedang menggunakan resource y di saat yang bersamaan T1 juga mengeksekusi resource x, tidak bisa ditambahkan sebab x nya lagi di pakai di T1

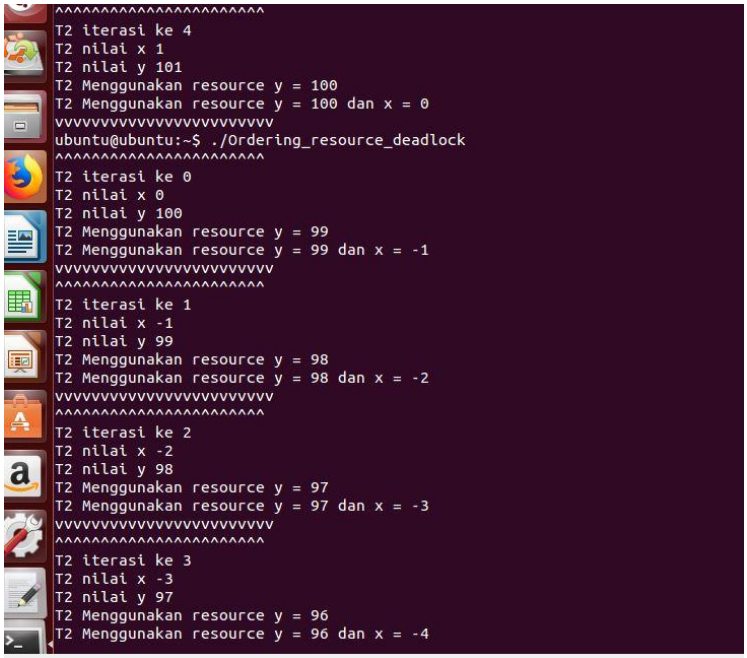
2. Solusi deadlock dengan ordering resource
   a. Copy paste source code ordering_resource_deadlock.c
      Done
   b. Beri komentar pada setiap baris kode!

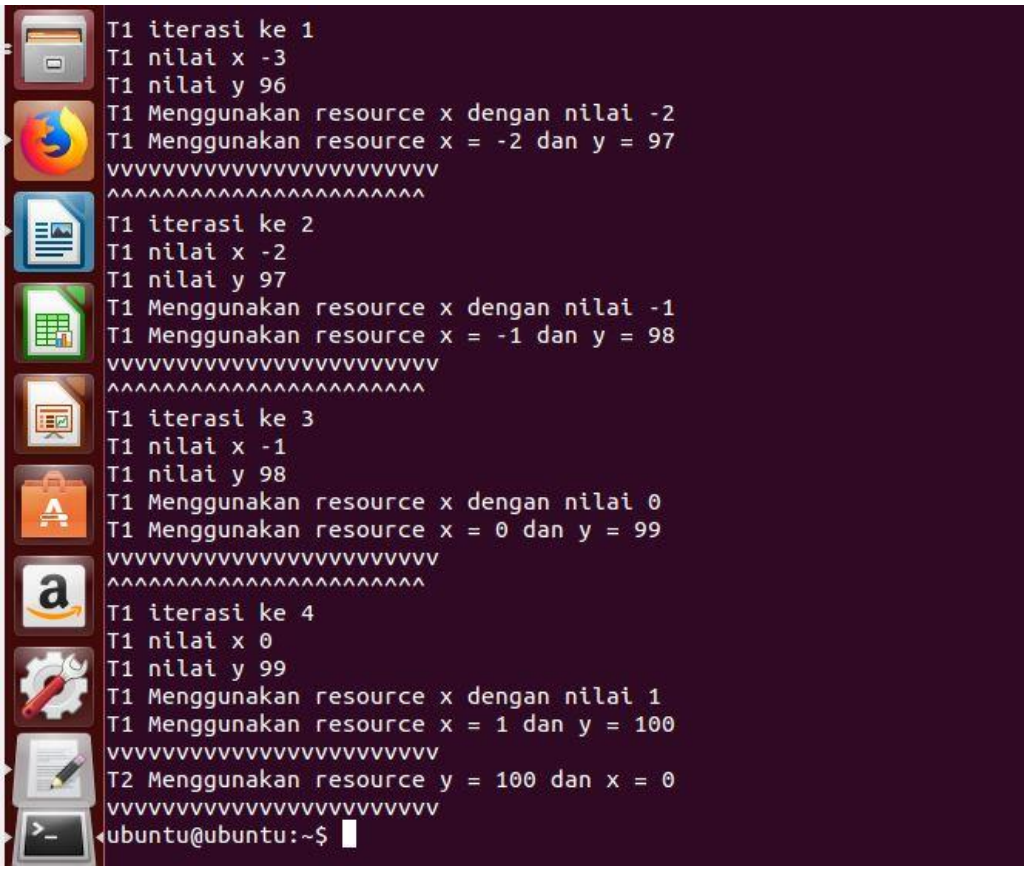


c. Jelaskan yang dilakukan kode tersebut!

- Thread Hello melakukan increament pada nilai x dan nilai y (shared_x dan Shared_y)
- Melakukan Descreament pada nilai x dan nilai y (shared_x dan shared_y)
- Mutex a melindungi resource shared_x
- Dan mutex b melindungi resource shared_y

d. Compile kode tersebut! Jalankan program tersebut minimal 10 kali! Amati hasilnya!





e. Jelaskan mengapa program tidak mengalami deadlock? Teknik apa yang dilakukan?

Dengan teknik ordering resource, memindahkan keteurutan dari mutex dan menghapus lock b, jadi T2 walaupun mengakses tapi tidak mengunci resource tersebut, diurutkan

resource yg akan dilindungi jika mengakses resource yang dilindungi maka di unlock terlebih dahulu.

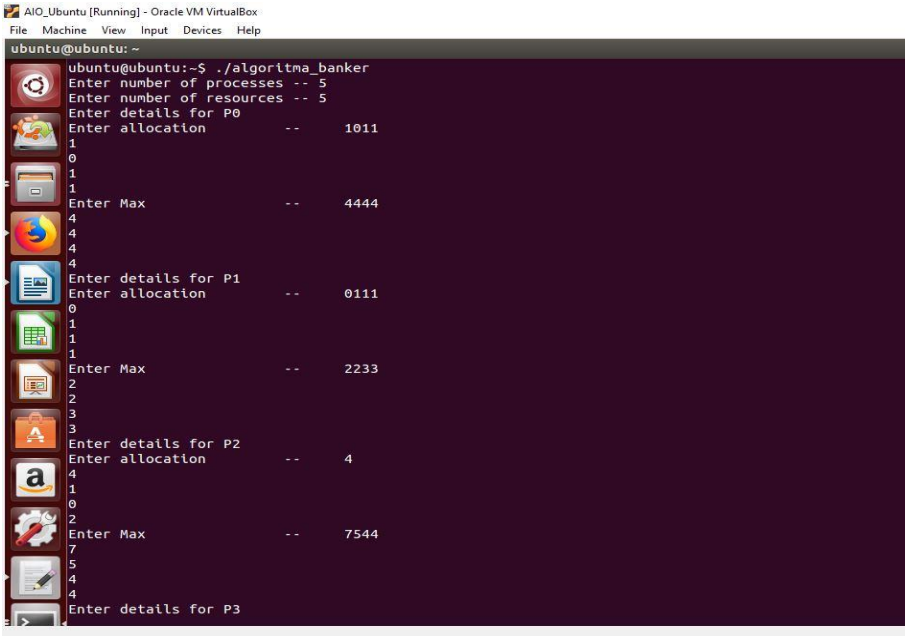
3. Perhitungan Algortima Banker
    a. Diketahui sistem sebagai berikut:

Available			
A	B	C	D
6	3	5	4

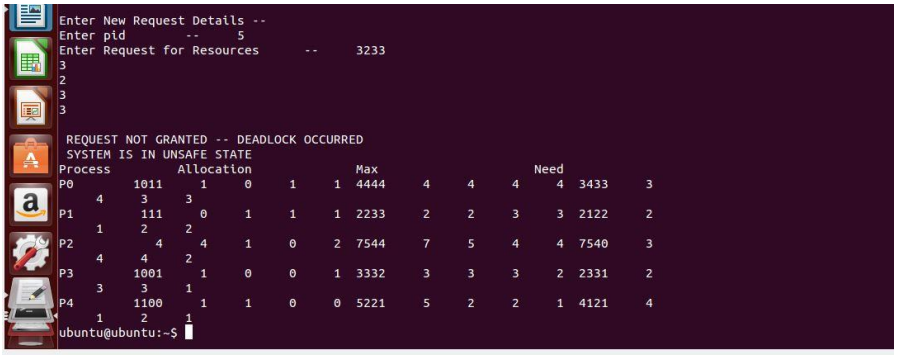
Process	Current allocation				Maximum demand			
	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

    b. Hitunglah matrik need!
    c. Apakah sistem ini safe? Jika safe sebutkan urutan terminasi proses, jika tidak safe apa yang menyebabkan!
    Safe dan bisa di eksekusi urutan terminasinya p1,p2,p0,p3,p4,p5.
    d. Jika P5 meminta resource [3, 2, 3, 3] apakah akan diberikan? Jelaskan!
    Selama kasusnya di awal maka akan diberikan, selama bank tersebut punya sumber daya yg dapat dipakai maka itu boleh, bisa dibilang jika dapat dipenuhi maka bisa.

4. Program Algoritma Banker
    a. Copy paste source code algortima_banker.c
    done
    b. Compile source code tersebut! Jalankan program!



c. Tunjukkan bahwa hasil perhitungan Anda pada soal 3 sama dengan program algoritma banker. Jika ada perbedaan jelaskan alasannya!



Source code

1.contoh_deadlock.c

==========

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>

pthread_mutex_t mutex_a, mutex_b;
int shared_x, shared_y;

void * Hello(void * input) {
```

```
  int i;
  for (i = 0; i < 5; i++) {
    pthread_mutex_lock( & mutex_a);
    printf("^^^^^^^^^^^^^^^^^^^^^^^^\n");
    printf("T1 iterasi ke %d\n", i);
    printf("T1 nilai x %d\n", shared_x);
    printf("T1 nilai y %d\n", shared_y);
    shared_x++;
    printf("T1 Menggunakan resource x dengan nilai %d\n", shared_x);
    pthread_mutex_lock( & mutex_b);
    shared_y++;
    printf("T1 Menggunakan resource x = %d dan y = %d\n", shared_x, shared_y);
    printf("vvvvvvvvvvvvvvvvvvvvvvvv\n");
    pthread_mutex_unlock( & mutex_b);
    pthread_mutex_unlock( & mutex_a);
  }
}

void * World(void * input) {
  int i;
  for (i = 0; i < 5; i++) {
          pthread_mutex_lock( & mutex_b);
          printf("^^^^^^^^^^^^^^^^^^^^^^^^\n");
    printf("T2 iterasi ke %d\n", i);
    printf("T2 nilai x %d\n", shared_x);
    printf("T2 nilai y %d\n", shared_y);
    shared_y--;
    printf("T2 Menggunakan resource y = %d\n", shared_y);
    pthread_mutex_lock( & mutex_a);
    shared_x--;
    printf("T2 Menggunakan resource y = %d dan x = %d\n", shared_y, shared_x);
    printf("vvvvvvvvvvvvvvvvvvvvvvvv\n");
    pthread_mutex_unlock( & mutex_a);
          pthread_mutex_unlock( & mutex_b);
  }
}

int main(int argc, char * argv[]) {

  pthread_t thread_1, thread_2;
```

```
  shared_x = 0;
  shared_y = 100;

  pthread_create( & thread_1, NULL, Hello, NULL);
  pthread_create( & thread_2, NULL, World, NULL);

  pthread_join(thread_1, NULL);
  pthread_join(thread_2, NULL);

}
```

==========

2. Ordering_resource_deadlock.c

==========

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>

pthread_mutex_t mutex_a, mutex_b;
int shared_x, shared_y;

void * Hello(void * input) {
  int i;
  for (i = 0; i < 5; i++) {
    pthread_mutex_lock( & mutex_a);
    printf("^^^^^^^^^^^^^^^^^^^^^^^^^\n");
    printf("T1 iterasi ke %d\n", i);
    printf("T1 nilai x %d\n", shared_x);
    printf("T1 nilai y %d\n", shared_y);
    shared_x++;
    printf("T1 Menggunakan resource x dengan nilai %d\n", shared_x);
    pthread_mutex_lock( & mutex_b);
    shared_y++;
    printf("T1 Menggunakan resource x = %d dan y = %d\n", shared_x, shared_y);
    printf("vvvvvvvvvvvvvvvvvvvvvvv\n");
```

```
    pthread_mutex_unlock( & mutex_b);
    pthread_mutex_unlock( & mutex_a);
  }
}

void * World(void * input) {
  int i;
  for (i = 0; i < 5; i++) {
    printf("^^^^^^^^^^^^^^^^^^^^^^^^\n");
    printf("T2 iterasi ke %d\n", i);
    printf("T2 nilai x %d\n", shared_x);
    printf("T2 nilai y %d\n", shared_y);
    shared_y--;
    printf("T2 Menggunakan resource y = %d\n", shared_y);
    pthread_mutex_unlock( & mutex_b);
    pthread_mutex_lock( & mutex_a);
    shared_x--;
    printf("T2 Menggunakan resource y = %d dan x = %d\n", shared_y, shared_x);
    printf("vvvvvvvvvvvvvvvvvvvvvvv\n");
    pthread_mutex_unlock( & mutex_a);
  }
}

int main(int argc, char * argv[]) {

  pthread_t thread_1, thread_2;
  shared_x = 0;
  shared_y = 100;

  pthread_create( & thread_1, NULL, Hello, NULL);
  pthread_create( & thread_2, NULL, World, NULL);

  pthread_join(thread_1, NULL);
  pthread_join(thread_2, NULL);

}
```

==========

4. algortima_banker.c

==========

```
#include <stdio.h>
  struct file {
    int all[10];
    int max[10];
    int need[10];
    int flag;
  };
void main() {
  struct file f[10];
  int fl;
  int i, j, k, p, b, n, r, g, cnt = 0, id, newr;
  int avail[10], seq[10];

  printf("Enter number of processes -- ");
  scanf("%d", & n);
  printf("Enter number of resources -- ");
  scanf("%d", & r);
  for (i = 0; i < n; i++) {
    printf("Enter details for P%d", i);
    printf("\nEnter allocation\t -- \t");
    for (j = 0; j < r; j++)
      scanf("%d", & f[i].all[j]);
    printf("Enter Max\t\t -- \t");
    for (j = 0; j < r; j++)
      scanf("%d", & f[i].max[j]);
    f[i].flag = 0;
  }
  printf("\nEnter Available Resources\t -- \t");
  for (i = 0; i < r; i++)
    scanf("%d", & avail[i]);
  printf("\nEnter New Request Details -- ");
  printf("\nEnter pid \t -- \t");
  scanf("%d", & id);
  printf("Enter Request for Resources \t -- \t");
  for (i = 0; i < r; i++) {
```

```
    scanf("%d", & newr);
    f[id].all[i] += newr;
    avail[i] = avail[i] - newr;
  }
  for (i = 0; i < n; i++) {
    for (j = 0; j < r; j++) {
      f[i].need[j] = f[i].max[j] - f[i].all[j];
      if (f[i].need[j] < 0)
        f[i].need[j] = 0;
    }
  }
  cnt = 0;
  fl = 0;
  while (cnt != n) {
    g = 0;
    for (j = 0; j < n; j++) {
      if (f[j].flag == 0) {
        b = 0;
        for (p = 0; p < r; p++) {
          if (avail[p] >= f[j].need[p])
            b = b + 1;
          else
            b = b - 1;
        }
        if (b == r) {
          printf("\nP%d is visited", j);
          seq[fl++] = j;
          f[j].flag = 1;
          for (k = 0; k < r; k++)
            avail[k] = avail[k] + f[j].all[k];
          cnt = cnt + 1;
          printf("(");
          for (k = 0; k < r; k++)
            printf("%3d", avail[k]);
          printf(")");
          g = 1;
        }
      }
    }
    if (g == 0) {
```

```
      printf("\n REQUEST NOT GRANTED -- DEADLOCK OCCURRED");
      printf("\n SYSTEM IS IN UNSAFE STATE");
      goto y;
    }
  }
  printf("\nSYSTEM IS IN SAFE STATE");
  printf("\nThe Safe Sequence is -- (");
  for (i = 0; i < fl; i++)
    printf("P%d ", seq[i]);
  printf(")");
  y: printf("\nProcess\t\tAllocation\t\tMax\t\t\tNeed\n");
  for (i = 0; i < n; i++) {
    printf("P%d\t", i);
    for (j = 0; j < r; j++)
      printf("%6d", f[i].all[j]);
    for (j = 0; j < r; j++)
      printf("%6d", f[i].max[j]);
    for (j = 0; j < r; j++)
      printf("%6d", f[i].need[j]);
    printf("\n");
  }

}
```

==========