Ιωάννης Τσάμπρας 1066584 5οετης
2η εργασία Λειτουργηκά Συστήματα

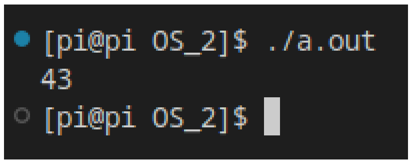
Code:

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

//setting thread ids arrays
int myid[100];
pthread_t tid[100];

//setting semaphore global vars
sem_t *x_sem;
sem_t *y_sem;
sem_t *z_sem;
sem_t *c1_sem;
sem_t *c2_sem;
sem_t *a2_sem;
sem_t *b1_sem;

//setting shared resources gloabal vars
int x,y,z,c1,c2,a2,b1;




void *t1(void *arg){
    int a1; //init non-shared var
    //implement given pseudo-code
    a1=10;
    a2=11;
    sem_post(a2_sem);
    sem_wait(c1_sem);
    y=a1+c1;
    sem_post(y_sem);
    sem_wait(x_sem);
    printf("%d\n",x);

}
```

```c
void *t2(void *arg){
    int b2,w; //init non-shared vars
    //implement given pseudo-code
    b1=20;
    b2=21;
    sem_post(b1_sem);
    sem_wait(c2_sem);
    w=b2+c2;
    sem_wait(y_sem);
    sem_wait(z_sem);
    x=z-y+w;
    sem_post(x_sem);

}

void *t3(void *arg){
    //implement given pseudo-code
    c1=10;
    c2=11;
    sem_post(c1_sem);
    sem_post(c2_sem);
    sem_wait(a2_sem);
    sem_wait(b1_sem);
    z=a2+b1;
    sem_post(z_sem);

}

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

//setting semaphores for each shared resource
    x_sem= sem_open("/semaphore_x", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_x");

    y_sem= sem_open("/semaphore_y", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_y");

    z_sem= sem_open("/semaphore_z", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_z");
    c1_sem= sem_open("/semaphore_c1", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_c1");


    c2_sem= sem_open("/semaphore_c2", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_c2");


    a2_sem= sem_open("/semaphore_a2", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_a2");


    b1_sem= sem_open("/semaphore_b1", O_CREAT, 0644, 0);
    sem_unlink("/semaphore_b1");





//starting the 3 threads
    if(pthread_create(&tid[1-1], NULL, &t1, &myid[1-1]) < 0){
            printf("thread failed\n");
    }
    if(pthread_create(&tid[2-1], NULL, &t2, &myid[2-1]) < 0){
            printf("thread failed\n");
    }
    if(pthread_create(&tid[3-1], NULL, &t3, &myid[3-1]) < 0){
            printf("thread failed\n");
    }
//joining the 3 threads
    pthread_join(tid[1-1], NULL);
    pthread_join(tid[2-1], NULL);
    pthread_join(tid[3-1], NULL);

}
```

Also on github: GitHub

Results:



```
[pi@pi OS_2]$ ./a.out
43
[pi@pi OS_2]$
```