1.

משתנים גלובליים:

Sem\_Binary – הצהרה על סמאפור בינארי

Global – משתנה גלובלי אותו נרצה לעדכן

פונקציות:

Binary – פונקציה שמאתחלת את Sem\_Binary, לוקחת אתו (חוסמת גישה באמצעותו) ומייצרת 2 taskים.

taskOne – רצה בלולאה בה הסמאפור נלקח, מדפיסה פלט ומעדכנת את global ונותנת את הסמאפור חזרה

taskTwo – מחזירה את הסמאפור ונכנסת ללולאה בה הסמאפור נלקח, מודפס פלט ומעודכן global והסמאפור ניתן חזרה.

ציפיות מהקוד:

נגדיר את הentry point שלנו כפונקציה binary. אם כך, אנחנו מצפים שהסמאפור כבר ילקח לפני יצירת הtaskים t1 וt2 וכשניגש לtaskOne מיצירת t1 בכניסה ללולאה ניעצר כי הסמאפור יהיה תפוס והוא יכנס לתור. נמשיך לtaskTwo, הסמאפור יחזור להיות זמין וכשנגיע ללולאה של taskTwo ונרצה לקחת את הסמאפור t2 יכנס לתור ההמתנה ואחרי t1 שעכשיו יקח את הסמאפור ויבצע איטרציה עד שיתן את הסמאפור ויכנס לתור בחזרה.  
כלומר התוצאה צפויה להיות: הדפסה של t1 וביצוע global++, הדפסה של t2 וביצוע global—לסירוגין עד שיסתיימו האיטרציות

2.

כאשר נמחק את הקודים תחת note1 וnote2 נצפה לקבל תוצאה בה נגיע ליצירת t1 ושם הוא יקח ראשון את הסמאפור בלולאה ויחזיר אותו בצורה בה הוא יוכל לבצע את כל האיטרציות שלו ברצף ורק כשיסיים ניגש לt2 שיבצע את האיטרציות שלו באותו אופן בדיוק.

3.

ניצור במקום הסמאפור הבינארי סמפור מנייה בה הספירה היא עד 1.

לשם כך ניקח את הקוד הרשום למעלה ונשנה את השורה:

semBinary = semBCreate(SEM\_Q\_FIFO, SEM\_FULL);

להיות:

semBinary = semCCreate(SEM\_Q\_FIFO, 1);

4.

בשביל לשנות את עדיפות התור להיות לפי priority במקום לפי FIFO נרצה לשנות את השורה

SemBCreate(SEM\_Q\_FIFO, SEM\_FULL);

להיות:

SemBCreate(SEM\_Q\_PRIORITY, SEM\_FULL);

כפי שציפינו כאשר t1 יגיע עם עדיפות גבוהה מt2 נקבל הרצה של t1 עד סוף האיטרציות שלו (global=10) ורק לאחר מכן יגיע t2 ויחזיר את global=0. כמובן שכאשר העדיפות תהיה הפוכה התוצאה תהיה הפוכה.

שאלות:

1. סמאפור בינארי יכול לשמש בין taskים או לסנכרון או למניעה הדדית בין taskים שונים שלא יגשו למשתנה גלובאלי. לסנכרון בין taskים נשתמש בסמאפור בהגדרה לפי עדיפות המשימות וכך כל משימה נותנת ולוקחת את הסמאפור והתור מנוהל לפי העדיפות.

לשם מניעה הדדית ניתן לסמאפור ניהול תור לפי סדר הגעה (FIFO), משימה שרוצה לגשת למשתנה הגלובלי תנסה לקחת את הסמאפור ואם הוא לא קיים תיכנס לתור במיקום הראשון שפנוי ותמתין עד שתוכל לגשת.

// לקרוא בדוקומנטציות בבית תשובות ל2 ו3

1. סמאפור בינארי הוא חסר בעלות, הוא יכול להלקח בפונקציה x ולחזור בפונקציה y. בקוד הנתון למשל, הסמאפור הבינארי נלקח בbinary וחוזר רק בtaskTwo. הסמפור יכול להיות בשימוש על ידי אינטרפטס רק לשם give ולא לשם take.
2. סמאפור mutex הוא בבעלות היוצר ולכן לא ניתן לשימוש בידי המשימות. אי אפשר לתת או לקחת סמפור זה באמעצות interrupt

**חלק 2:**

1. המשימות המחזוריות הן:

T1,T2,T3

המשימה הספורדית היא:

T4

1. קוד (מצורף גם כקובץ c):

#include "vxWorks.h"

#include "taskLib.h"

#include "semLib.h"

#include "stdio.h"

#define ITER 10

#define STACK\_SIZE 2000

/\* function prototypes \*/

void taskOne(int);

void taskTwo(int);

void taskThree(int);

void taskFour();

/\* globals \*/

SEM\_ID semBinary;

SEM\_ID semFour;

int var = 0;

void binary(void)

{

int taskIdOne, taskIdTwo, taskIdThree, taskIdFour;

int ticks=15,ticks3=20;

/\* create semaphore with semaphore available and queue tasks on PRIORITY basis \*/

semBinary = semBCreate(SEM\_Q\_PRIORITY, SEM\_EMPTY);

semFour = semBCreate(SEM\_Q\_PRIORITY, SEM\_FULL);

/\* spawn the two tasks \*/

printf("Start\n");

taskIdOne = taskSpawn("t1",105,0x100,STACK\_SIZE,(FUNCPTR)taskOne,ticks,0,0,0,0,0,0,0,0,0);

printf("First Task spawned\n");

taskIdTwo = taskSpawn("t2",105,0x100,STACK\_SIZE,(FUNCPTR)taskTwo,ticks,0,0,0,0,0,0,0,0,0);

printf("Second task spawned\n");

taskIdThree = taskSpawn("t3",102,0x100,STACK\_SIZE,(FUNCPTR)taskThree,ticks3,0,0,0,0,0,0,0,0,0);

printf("Third task Spawned\n");

taskIdFour = taskSpawn("t4",101,0x100,STACK\_SIZE,(FUNCPTR)taskFour,0,0,0,0,0,0,0,0,0,0);

}

void taskOne(int ticks)

{

int i;

for (i=0; i < ITER; i++)

{

taskDelay(ticks);

semTake(semBinary,WAIT\_FOREVER);

var+=3;

printf("Task 1. VAR = %d\n",var);

semGive(semBinary);

}

}

void taskTwo(int ticks)

{

int i;

for (i=0; i < ITER; i++)

{

taskDelay(ticks);

semTake(semBinary,WAIT\_FOREVER);

var+=5;

printf("Task 2. VAR = %d\n",var);

semGive(semBinary);

}

semTake(semFour,WAIT\_FOREVER);

}

void taskThree(int ticks3)

{

int i;

for(i=0;i<ITER;i++)

{

taskDelay(ticks3);

semTake(semBinary, WAIT\_FOREVER);

printf("Task 3\n");

if(var%3==0 || var%5==0){

semGive(semFour);

}

semGive(semBinary);

}

}

void taskFour (){

int i;

semGive(semBinary);

for(i=0;i<ITER;i++){

printf("Task 4\n");

semTake(semFour,WAIT\_FOREVER);

if(var%3==0){

printf("VAR is divided by 3\n");

}

if(var%5==0){

printf("VAR is divided by 5\n");

}

}

}