**מעבדה 8 - סמפורים**

בעבודה עם סמפורים ממשק המשתמש של down במ"ה לינוקס ( CentOS ) הוא sem\_wait

וממשק המשתמש של up הוא sem\_post

הפונקציות UP ו-DOWN הן פונקציות פנימיות ונלמדו כמתודות כלליות (בחומר תיאורטי). באופן ספציפי, לכל מ"ה יש ממשק משתמש בשם שונה למתודות אלו.

To initialize a semaphore, use [sem\_init](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_init.html):

int [sem\_init](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_init.html)(sem\_t \*sem, int pshared, unsigned int value);

* sem points to a semaphore object to initialize
* pshared is a flag indicating whether or not the semaphore should be shared with fork()ed processes. In Linux only 0;
* value is an initial value to set the semaphore to

Example of use:  sem\_init(&sem\_name, 0, 10);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To wait on a semaphore, use [sem\_wait](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_wait.html):

int [sem\_wait](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_wait.html)(sem\_t \*sem);

Example of use:  sem\_wait(&sem\_name);

* If the value of the semaphore isn't positive, the calling process blocks; one of the blocked processes wakes up when another process calls sem\_post.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To increment the value of a semaphore, use [sem\_post](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_post.html):

int [sem\_post](http://pubs.opengroup.org/onlinepubs/009695399/functions/sem_post.html)(sem\_t \*sem);

Example of use:

sem\_post(&sem\_name);

* It increments the value of the semaphore and wakes up a blocked process waiting on the semaphore, if any.  
  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

לעבודה עם סמפורים נצטרך ספריית <semaphore.h>

מכריזים על משתנה סמפור כמשתנה מסוג sem\_t

למשל:

sem\_t s;   
sem\_init(&s, 0, 1);

**תרגיל 1.**

נתונה התכנית הבאה הממומשת תחת UNIX

#include <pthread.h>

#include <stdio.h>

#include <string.h>

void\* create\_message ( void\* str ){

int i = 0;

for ( i = 0; i < 10; i++ ) printf ( "I've wrote a message #%d. %s\n", i+1, (char\*) str );

}

int main (){

pthread\_t thread;

int i = 0;

pthread\_create (&thread, NULL, create\_message, (void\*)"Thread A" );

for ( i = 0; i < 10; i++ ) printf ( "The message #%d was printed. Thread B \n", i+1 );

 pthread\_join(thread, NULL);

}

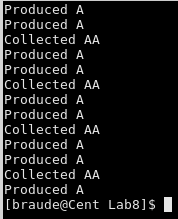
הפלט שלה:



הוסיפו לתכנית שימוש בסמפורים כך שיתאפשר הפלט הבא:

:

**תרגיל 2**

מכונה A מייצרת פריט כל 2 שניות. מכונה AA מרכיבה מוצר מ-2 פריטים שמכונה A מייצרת.

מכונה AA יכולה להרכיב מוצר כל שנייה, אבל היא צריכה 2 פריטים למוצר, לכן מכונה AA ממתינה ל-2 פריטים ממכונה A. למכונה AA אסור לבצע sleep.

כל מכונה עובדת בלולאה אין סופית, אבל המפעל נסגר אחרי 20 שניות ולא מחכה לסיום עבודת המכונות.

כתבו תכנית אשר מפעילה את המכנות כ-2 חוטים ומאפשרת למפעל ולמכונות שבו, עבודה מסודרת לפי דרישות המפעל. בפונקציות של החוטים printf לא יופיע יותר מפעם אחת. מותר להשתמש בסמפור אחד בלבד ואסור להשתמש ב-if.

**תרגיל 3**

עליכם לכתוב תכנית היוצרת 3 חוטים. חוט אחד מדפיס A בלולאה אינסופית. חוט שני מדפיס B בלולאה אינסופית וחוט שלישי מדפיס C בלולאה אינסופית. לכל חוט פונקציה משלו הרצה ב while (1) ומבצעת printf (פעם אחת בלבד בתוך הלולאה).

נדרש להוסיף סמפורים ולסנכרן את החוטים כך שהחוטים יעבדו בסדר הבא:

A  
A  
B  
C  
A  
A  
B  
C  
A  
A  
B  
C

כלומר, בתוכנית עליכם להגדיר main המאתחל סמפורים ומייצר 3 חוט. החוטים יפעילו את הפונקציות void\* A(void\*); void\* B(void\*); void\* C(void\*); אשר מבצעות הדפסה ו sem\_wait , sem\_post על מנת ליצור את הסדר הנ"ל. נדרש שהתוכנית תרוץ 5 שניות בלבד.