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:

```
int sem_init(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:

```
int sem_wait(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:

```
int sem_post(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:

```
The message #1 was printed. Thread B
The message #2 was printed. Thread B
The message #3 was printed. Thread B
The message #4 was printed. Thread B
The message #5 was printed. Thread B
The message #6 was printed. Thread B
The message #7 was printed. Thread B
The message #8 was printed. Thread B
The message #9 was printed. Thread B
The message #10 was printed. Thread B
I've wrote a message#1. Thread A
I've wrote a message#2. Thread A
I've wrote a message#3. Thread A
I've wrote a message#4. Thread A
I've wrote a message#5. Thread A
I've wrote a message#6. Thread A
I've wrote a message#7. Thread A
I've wrote a message#8. Thread A
I've wrote a message#9. Thread A
I've wrote a message#10. Thread A
[braude@Cent Lab7]$
```

הוסיפו לתכנית שימוש בסמפורים כך שיתאפשר הפלט הבא:

```
I've wrote a message #1. Thread A
The message #1 was printed. Thread B

I've wrote a message #2. Thread A
The message #2 was printed. Thread B

I've wrote a message #3. Thread A
The message #3 was printed. Thread B

I've wrote a message #4. Thread A
The message #4 was printed. Thread B

I've wrote a message #5. Thread A
The message #5 was printed. Thread B

I've wrote a message #6. Thread A
The message #6 was printed. Thread B

I've wrote a message #7. Thread A
The message #7 was printed. Thread B

I've wrote a message #8. Thread A
The message #8 was printed. Thread B

I've wrote a message #9. Thread A
The message #9 was printed. Thread B

I've wrote a message #10. Thread A
The message #10 was printed. Thread B
[braude@Cent Lab7]$
```

:

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.

```
Produced A
Produced A
Collected AA
Produced A
Produced A
Collected AA
Produced A
Produced A
Collected AA
Produced A
Produced A
Collected AA
Produced A
[braude@Cent Lab8]$
```