ΣΗΜΑΦΟΡΟΥΣ (SEMAPHORES)

Λειτουργικά Συστήματα - Εργαστήριο

Πάνος Παπαδόπουλος

Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Πανεπιστήμιο Θεσσαλίας

ΘΕΩΡΙΑ

ΛΕΙΤΟΥΡΓΙΑ ΣΗΜΑΦΟΡΩΝ

- **Μηχανισμός συγχρονισμού** threads και processes
- Επίλυση προβλημάτων **ραντεβού**
- Επίλυση προβλημάτων **αμοιβαίου αποκλεισμού (mutual exclusion)**
- Καθορισμός / υλοποίηση **κρίσιμης περιοχής (critical section)**

ΤΥΠΟΙ ΣΗΜΑΦΟΡΩΝ

- **Δυαδικός σημαφόρος (binary semaphore)**: τιμές που μπορεί να λάβει **0 / 1**

- **Γενικός / μετρητής σημαφόρος (general / counting semaphore)**: τιμές που μπορεί να λάβει **>= 0**

ΒΑΣΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

- **sem_init()**: αρχικοποίηση σημαφόρου
- **sem_wait() / sem_trywait()**: κλείδωμα σημαφόρου
- **sem_post()**: ξεκλείδωμα σημαφόρου
- **sem_destroy()**: καταστροφή σημαφόρου

ΑΡΧΙΚΟΠΟΙΗΣΗ ΣΗΜΑΦΟΡΟΥ - SEM_INIT() - (1/2)

- #include <semaphore.h>
- Prototype: int sem_init(sem_t *sem, int pshared, unsigned int value)
- **Αρχικοποίηση** μιας ανώνυμης σημαφόρου στην οποία μπορούμε να αναφερόμαστε χρησιμοποιώντας την παράμετρο **sem**.
- Αν η παράμετρος **pshared** έχει **τιμή διαφορετική του 0**, τότε η σημαφόρος είναι **κοινή** για όλα τα **processes**.
- Αν η παράμετρος **pshared** έχει **τιμή 0**, τότε η σημαφόρος είναι **κοινή** για όλα τα **threads** ενός process.

ΑΡΧΙΚΟΠΟΙΗΣΗ ΣΗΜΑΦΟΡΟΥ - SEM_INIT() - (2/2)

- Η παράμετρος **value** καθορίζει την αρχική τιμή της σημαφόρου.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.
- **Προσπάθεια αρχικοποίησης μιας ήδη αρχικοποιημένης σημαφόρου οδηγεί σε απροσδιόριστη συμπεριφορά.**

ΚΛΕΙΔΩΜΑ ΣΗΜΑΦΟΡΟΥ - SEM_WAIT()

- #include <semaphore.h>
- int sem_wait(sem_t *sem)
- **”Κλείδωμα”** σημαφόρου στην οποία μπορούμε να αναφερόμαστε χρησιμοποιώντας την παράμετρο **sem**.
- **Μειώνεται η τιμή της σημαφόρου ή μπλοκάρει το thread/process**.
- Αν η **τιμή είναι μεγαλύτερη του 0**, τότε το thread ή process **την μειώνει** και **συνεχίζει κανονικά την εκτέλεσή του**.
- Αν η **τιμή είναι 0**, τότε το thread ή process **μπλοκάρει**.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.

ΞΕΚΛΕΙΔΩΜΑ ΣΗΜΑΦΟΡΟΥ - SEM_POST()

- #include <semaphore.h>
- int sem_post(sem_t *sem)
- **”Ξεκλείδωμα”** σημαφόρου στην οποία μπορούμε να αναφερόμαστε χρησιμοποιώντας την παράμετρο **sem**.
- **Αυξάνεται η τιμή της σημαφόρου**.
- Αν η **τιμή (μετά τη αύξηση) γίνει μεγαλύτερη του 0**, τότε **κάποιο άλλο thread ή process** που έχει μπλοκάρει στην sem_wait(), **ξεμπλοκάρει** και μπορεί να ”κλειδώσει” αυτό τη σημαφόρο.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.

ΚΑΤΑΣΤΡΟΦΗ ΣΗΜΑΦΟΡΟΥ - SEM_DESTROY()

- #include <semaphore.h>
- int sem_destroy(sem_t *sem)
- **Καταστροφή** μιας ανώνυμης σημαφόρου η οποια βρίσκεται στην διεύθυνση μνήμης που δείχνει η παράμετρος **sem**.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.
- **Μόνο** σημαφόροι που έχουν **αρχικοποιηθεί** χρησιμοποιώντας την συνάρτηση **sem_init()** πρέπει να καταστρέφονται χρησιμοποιώντας την sem_destroy().
- **Καταστροφή σημαφόρου που χρησιμοποιείται είτε από threads είτε από processes οδηγεί σε απροσδιόριστη συμπεριφορά**.

ΣΗΜΑΦΟΡΟΙ ΣΕ THREADS (1/3)

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

void *thread_func(void* arg);

sem_t sem;

int main (int argc, char *argv[])
{
    pthread_t t1, t2;

    sem_init(&sem, 0, 1);

    pthread_create(&t1, NULL, thread_func, NULL);
    sleep(2);
    pthread_create(&t2, NULL, thread_func, NULL);
```

ΣΗΜΑΦΟΡΟΙ ΣΕ THREADS (2/3)

```
20      pthread_join(t1, NULL);
21      pthread_join(t2, NULL);
22
23      sem_destroy(&sem);
24
25      return 0;
26  }
```

ΣΗΜΑΦΟΡΟΙ ΣΕ THREADS (3/3)

```
27 void* thread_func(void* arg)
28 {
29     sem_wait(&sem);
30     printf("\nentered critical section ...\n");
31
32     // do something
33     sleep(4);
34
35     printf("\nexiting critical section ...\n");
36     sem_post(&sem);
37
38     return NULL;
39 }
```