Λειτουργικά Συστήματα 6ο εξάμηνο ΣΗΜΜΥ ΕΜΠ Ακ. Έτος 2021-2022

Εργαστηριακή Άσκηση 4
Συγχρονισμός

Εργαστηριακή Άσκηση 4

Οι σημαφόροι (**semaphores**) είναι ένας ΑΤΔ, και χρησιμοποιούνται για τον έλεγχο πρόσβασης πολλαπλών προγραμμάτων/νημάτων σε έναν κοινό πόρο (resource).

Στόχος της άσκησης είναι η δημιουργία μίας βιβλιοθήκης που να υλοποιεί την δομή του σημαφόρου, καθώς και η δοκιμή της ώστε να εξασφαλίζεται ότι η υλοποίηση είναι πλήρης και ορθή.

Η υλοποίηση (προφανώς) **δεν πρέπει** να χρησιμοποιεί την δομή σημαφόρου που παρέχεται από το UNIX (semaphore.h). Αντίθετα, ζητείται ο σημαφόρος να υλοποιηθεί εσωτερικά με την χρήση σωληνώσεων (pipes).

Synchronization

Απαιτείται όταν πολλές διεργασίες/νήματα θέλουν να έχουν ταυτόχρονη πρόσβαση σε έναν κοινό πόρο (resource, critical section).

Γίνεται με την χρήση ειδικών δομών συγχρονισμού (κλειδωμάτων), π.χ. Spinlocks, mutexes, semaphores, κλπ.

Semaphore

Οι σημαφόροι είναι δομή που χρησιμοποιείται για τον συγχρονισμό μίας ή περισσότερων διεργασιών/νημάτων. Περιέχουν ως state έναν ακέραιο αριθμό, και υλοποιούν τις 2 παρακάτω λειτουργίες:

1. **P(X) / post / signal**
   Αυξάνει την τιμή του σημαφόρου κατά X (*ατομικά*).

2. **V(X) / wait / release**
   Μειώνει την τιμή του σημαφόρου κατά X (*ατομικά*). Η λειτουργία αυτή μπλοκάρει σε περίπτωση που η τιμή του σημαφόρου γίνεται αρνητική.

Semaphore (2)

Οι σημαφόροι εφαρμόζονται σε προβλήματα τύπου παραγωγού / καταναλωτή.

Οι παραγωγοί εισάγουν κάποια κομμάτι εργασίας σε μία ουρά και αυξάνουν την τιμή του σημαφόρου (λειτουργία P). Η λειτουργία P δεν μπλοκάρει ποτέ.

Οι καταναλωτές *περιμένουν* μέχρι να υπάρχει κάποιο διαθέσιμο κομμάτι εργασίας, μειώνουν την τιμή του σημαφόρου (λειτουργια V) και το αφαιρούν από την ουρά. Αν η ουρά είναι άδεια, τότε μπλοκάρουν μέχρι κάποιος παραγωγός να προσθέσει κάτι.

Semaphore (3)

Παράδειγμα: Βιβλία σε ράφι

1. Παραγωγός 1: Πρόσθεσε 3 βιβλία (OK)
2. Καταναλωτής 1: Πάρε 2 βιβλία (OK)
3. Καταναλωτής 2: Πάρε 1 βιβλίο (OK)
4. Παραγωγός 2: Πρόσθεσε 1 βιβλίο (OK)
5. Καταναλωτής 3: Πάρε 2 βιβλία (blocked) (*)
6. Παραγωγός 3: Πρόσθεσε 1 βιβλίο (OK)
7. Καταναλωτής 3: Πάρε 2 βιβλία (unblocked, OK)

Shared Libraries

Οι βιβλιοθήκες υλοποιούν συναρτήσεις, τύπους δεδομένων, αλγορίθμους, …

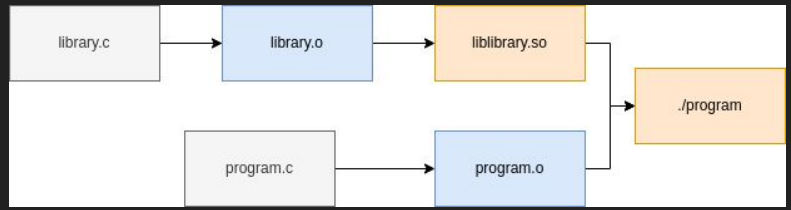
Γράφονται με τέτοιο τρόπο ώστε να μπορούν να χρησιμοποιούνται από πολλά προγράμματα.

```c
// library.h
void foo();
```

```c
// library.c
#include "library.h"

void foo() {
    printf("Hello World\n");
}
```

```c
// program.c
#include "library.h"

int main() {
    foo();
}
```

```
$ gcc library.c -fPIC -shared -o liblibrary.so
$ gcc program.c -lmylib -L. -o program

$ ./program
```



Implementation

Σας δίνεται το αρχείο δηλώσεων **sem.h**. Περιέχει αναλυτικά comments για τις παραμέτρους και τα return values των συναρτήσεων που πρέπει να υλοποιήσετε.

```c
#ifndef _SEM_H_INCLUDED
#define _SEM_H_INCLUDED

You, 57 minutes ago | 1 author (You)
struct pipe_semaphore_t {
    // TODO: member fields for internal state of semaphore ...
};

typedef struct pipe_semaphore_t pipe_semaphore_t;

/**…
pipe_semaphore_t *new_semaphore();

/**…
int semaphore_wait(pipe_semaphore_t *sem, int val);

/**…
int semaphore_signal(pipe_semaphore_t *sem, int val);

#endif /* _SEM_H_INCLUDED */
```

Implementation (2)

Σας δίνεται επίσης ο σκελετός **sem.c** της υλοποίησης της βιβλιοθήκης.

Αυτό που καλείστε να κάνετε είναι να συμπληρώσετε κατάλληλα την υλοποίηση ώστε να ικανοποιούνται τα semantics της δομής του σημαφόρου.

```c
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

#include "sem.h"

pipe_semaphore_t *new_semaphore() {
  // TODO: initialize semaphore
  return NULL;
}

int semaphore_wait(pipe_semaphore_t *s, int val) {
  // TODO: implement semaphore wait operation
  return -1;
}

int semaphore_signal(pipe_semaphore_t *s, int val) {
  // TODO: implement semaphore signal operation
  return -1;
}
```

Implementation (3)

Σας δίνεται επίσης μια σειρά απο tests (στο αρχείο tests.c) τα οποία ελέγχουν την ορθότητα της υλοποίησης.

Η υλοποίηση πρέπει να περνάει όλα τα tests για να θεωρηθεί σωστή.

```
61 > /**…
67   void test_single_process() {
68     pipe_semaphore_t *s = new_semaphore();
69
70     semaphore_signal(s, 1);
71     semaphore_wait(s, 1);
72
73     exit(0);
74   }
```

```
ubuntu@wayfarer ~/D/d/n/2/LAB4 (master) $ make test
./tests test_single_process                ... SUCCESS
./tests test_multi_process                 ... SUCCESS
./tests test_many_signals                  ... SUCCESS
./tests test_liveness                      ... SUCCESS
./tests test_loop                          ... SUCCESS
./tests test_wait_without_signal           ... SUCCESS
./tests test_wait_without_enough_signal ... SUCCESS
```

```
ubuntu@wayfarer ~/D/d/n/2/LAB4 (master) $ make test
./tests test_single_process                ... SUCCESS
./tests test_multi_process                 ... SUCCESS
./tests test_many_signals                  ... SUCCESS
./tests test_liveness                      ... SUCCESS
./tests test_loop                          ... FAILED (TIMED OUT)
./tests test_wait_without_signal           ... SUCCESS
./tests test_wait_without_enough_signal ... FAILED

One or more tests failed. Lookup the test descriptions in tests.c
and update your semaphore implementation accordingly, then re-run:

    make test
```

Implementation (4)

Για ευκολία σας, δίνεται επίσης makefile που χτίζει την βιβλιοθήκη, και εκτελεί τους ελέγχους.

$ make test
$ make clean

```
1  ##################################################
2  # SEMAPHORE shared library
3  sem.o: sem.c sem.h
4      $(CC) -c sem.c -fPIC -o sem.o -Wall -Wextra
5  libsem.so: sem.o
6      $(CC) sem.o -shared -o libsem.so -Wall -Wextra
7
8  ##################################################
9  # SEMAPHORE library tests
10 tests: tests.c libsem.so
11     $(CC) tests.c -L. -lsem -lpthread -o tests -Wall
12 test: tests
13     @./run_test.sh
14
15 ##################################################
16 # make clean
17 .PHONY = clean
18 clean:
19     rm -rfv sem.o libsem.so tests
```

Definition Of Done

1. Ποια είναι τα χαρακτηριστικά μίας δομής συγχρονισμού; (Liveness,
2. Υλοποίηση της βιβλιοθήκης sem.c η οποία περνάει όλα τα tests.
3. Γνώση του τι είναι τα shared libraries, πώς χρησιμοποιούνται, πώς μπορούμε να γράψουμε ένα shared library.
4. Τι ειναι το LD_LIBRARY_PATH

References

1. man 2 pipe, read, write
2. man sem_overview
3. man ldd
4. man ld
5. https://linuxhint.com/what-is-ld-library-path/
6. https://www.tutorialspoint.com/makefile/makefile_quick_guide.htm

Q&A