Κλήση Συστήματος semget()

int semget(key_t key, int nsems, int semflag)

key: οποιοσδήποτε ακέραιος (απαιτείται όμως casting στον τύπο key_t)
nsems: το πλήθος των σημαφόρων που θέλουμε να δημιουργήσουμε
semflag: δικαιώματα και πράξεις πρόσβασης (οι δυνατές πράξεις είναι IPC_CREAT ή IPC_EXCL). Συνήθως θέτουμε αυτό το flag στη τιμή IPC_CREAT | 0660

Έτσι μια τυπική κλήση της semget είναι:

```
sid = semget( (key_t) 1234, 1, IPC_CREAT | 0660))
```

Κλήση συστήματος semop()

int semop(int semid, struct sembuf *opstr, int nops)

semid: το id που επέστρεψε η semget
opstr: η διεύθυνση ενός πίνακα που περιέχει τις πράξεις που θα εφαρμοστούν στους σημαφόρους
nops: το μέγεθος του προηγούμενου πίνακα

Ο πίνακας που χρησιμοποιείται στο δεύτερο όρισμα αποτελείται από τις παρακάτω δομές. Κάθε μια από αυτές τις δομές καθορίζει μια πράξη πάνω σε ένα σημαφόρο:

```
struct sembuf {
        short sem_num;
        short sem_op;
        short sem_flg;
};
```

sem_num: καθορίζει σε ποιον σημαφόρο από αυτούς που ορίσαμε στην semget θα εφαρμοστεί η πράξη (άρα μπορεί να πάρει τιμές από 0 έως nsems-1)
sem_op: το θέτουμε <0 (συνήθως -1) για δέσμευση ενός σημαφόρου και >0 (συνήθως 1) για αποδέσμευση
sem_flg: συνήθως το θέτουμε ίσο με 0

Κλήση συστήματος semctl

int semctl (int semid, int semnum, int cmd, union semun arg)

semid: το id που επέστρεψε η semget

semnum: ο σημαφόρος του συνόλου στον οποίο θα εφαρμοστεί η πράξη (0 έως nsems-1)
cmd: η πράξη που θέλουμε να εφαρμόσουμε στον σημαφόρο (πιθανές πράξεις είναι οι SETVAL, GETVAL, SETALL, GETALL, IPC_RMID).
arg: χρησιμοποιείται σε συνδυασμό με την πράξη. Ουσιαστικά περιέχει τα ορίσματα της πράξης. Οι δύο πρώτες πράξεις (που είναι και οι πιο συνηθισμένες μαζί με την τελευταία) χρησιμοποιούν το πεδίο val της union senum, η οποία ορίζεται παρακάτω (ο ορισμός της βρίσκεται στο `linux/sem.h`. Αν δεν χρησιμοποιούμε Linux μπορούμε να την ορίσουμε εμείς οι ίδιοι στο κώδικά μας όπως φαίνεται και παρακάτω):

```
union senum {
        int val;
        struct semid ds *buff;
        unsigned short *array;
};
```

Μια συνηθισμένη πράξη της semctl είναι η αρχικοποίηση των σημαφόρων.
Για να αρχικοποιήσουμε το 3^{ο} σημαφόρο ενός συνόλου σημαφόρων, που δημιουργήσαμε με την semget, στην τιμή 1 θα χρησιμοποιούσαμε τις εντολές:

```
union senum arg;
arg.val=1;
semctl(semid, 2, SETVAL, arg)
```

Βέβαια δεν είναι υποχρεωτικό να αρχικοποιηθεί κάποιος σημαφόρος. Τότε όμως τι τιμή θα έχει; (αφήνεται ως άσκηση)

Για να διαγράψουμε ένα σημαφόρο θα χρησιμοποιούσαμε την εντολή:

```
semctl(semid, 0, IPC_RMID,0)
```

Προσοχή: Πάντα να ελέγχονται οι τιμές που επιστρέφουν οι κλήσεις συστήματος και τα τυχόν σφάλματα καλό είναι να εμφανίζονται στην οθόνη με την συνάρτηση perror(char* errorString)

Ένα παράδειγμα χρήσης των παραπάνω κλήσεων είναι το παρακάτω. Το πρόγραμμα αυτό δεν είναι πλήρες και δεν συμπεριλαμβάνεται έλεγχος λαθών για συντομία και απλότητα.

```
/*απαραίτητες δηλώσεις επικεφαλίδων*/
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

/*δημιουργία ενός νέου σημαφόρου που θεωρούμε ότι θα χρησιμοποιηθεί για δέσμευση διαμοιραζόμενης μνήμης */
int semid = semget((key_t) 333, 1, IPC_CREAT | 0600);

/*αρχικοποίησή του στην τιμή 1, η μνήμη δεν είναι δεσμευμένη*/
union senum arg;
arg.val=1;
semctl(semid, 0, SETVAL, arg);

/*κάνουμε down στον σημαφόρο για να δεσμεύσουμε τη μνήμη*/
struct sembuf semopr;
semopr.sem_num = 0;
semopr.sem_op = -1;
semopr.sem_flg = 0;
semop(semid, &semopr, 1);

/*Σε αυτό το σημείο γράφουμε ή διαβάζουμε κάτι από τη διαμοιραζόμενη μνήμη */

/*Στη συνέχεια αποδεσμεύουμε τη διαμοιραζόμενη μνήμη, δηλαδή κάνουμε up στο σημαφόρο*/
semopr.sem_num = 0;
semopr.sem_op = 1;
semopr.sem_flg = 0;
semop(semid, &semopr, 1);

/*Τέλος διαγράφουμε το σημαφόρο*/
semctl(semid, 0, IPC_RMID,0)
```