ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

# Λειτουργικά Συστήματα

## 7ο εξάμηνο, Ακαδημαϊκή περίοδος 2010-2011

## Εξετάσεις Φεβρουαρίου 2011

### Διάρκεια: 2.5 ώρες

Οι εξετάσεις θα πραγματοποιηθούν ΧΩΡΙΣ την παρουσία βιβλίων, βοηθημάτων ή άλλου είδους σημειώσεων. Το μόνο που επιτρέπεται να έχετε μαζί σας είναι ένα φύλλο Α4, στο οποίο μπορείτε να έχετε γράψει ό,τι έχετε κρίνει πιο σημαντικό για το μάθημα και θέλετε να το έχετε ως βοήθημά σας. Απαγορεύεται η ανταλλαγή οποιουδήποτε αντικειμένου κατά την ώρα της εξέτασης, ούτε και των φύλλων Α4 που είναι ατομικά.

## Θέμα 1 (25%)

**α. (20%)** Δίνεται το ακόλουθο τμήμα κώδικα:

```
int fd[3][2], i, j, a, ret;

pipe(fd[0]); pipe(fd[1]); pipe(fd[2]);

for (i = 0; i < 3; i++) {
    ret = fork();
    if (ret != 0)
        continue;
    switch (i) {
        case 0:
            a = 3; write(fd[2][1], &a, sizeof(a)); break;
        case 1:
            a = 2; write(fd[1][1], &a, sizeof(a)); break;
        case 2:
            a = 4;
            read(fd[2][0], &a, sizeof(a)); write(fd[0][1], &a, sizeof(a));
    }
    read(fd[i][0], &a, sizeof(a));
```

```
    for (j = 0; j < a; j++) {
        ret = fork();
        if (ret == 0) {
            Fn(i, j); exit(0);
        }
    }
    for (j = 0; j < a; j++) wait(NULL);
}

for (i = 0; i < 3; i++) wait(NULL);

a = 1; write(fd[2][1], &a, sizeof(a));
```

Οι κλήσεις συστήματος δεν αποτυγχάνουν κι η `Fn()` δεν επιστρέφει ποτέ.
Σχεδιάστε το δέντρο διεργασιών που τελικά προκύπτει. Για κάθε κόμβο του δέντρου, γράψτε την κλήση συστήματος ή συνάρτηση που εκτελεί η αντίστοιχη διεργασία, μαζί με τα ορίσματά της. Συμπληρώστε το δέντρο διεργασιών ώστε να φαίνεται η διαδιεργασιακή επικοινωνία: για κάθε μεταφορά δεδομένων, σχεδιάστε ένα διακεκομμένο βέλος από τη διεργασία-αποστολέα στη διεργασία-παραλήπτη.

**β. (5%)** Σχεδιάστε ένα ενδεικτικό διάγραμμα μετάβασης καταστάσεων διεργασίας σε σύγχρονο ΛΣ. Έστω οι παρακάτω δύο συναρτήσεις, `Fna()` και `Fnb()`:

```
void Fna(void) {
    unsigned int i = 0;
    while (1) ++i;
}
```

```
void Fnb(void) {
    while (1) sleep(1);
}
```

Από ποιες καταστάσεις περνάει μια διεργασία όταν εκτελεί την `Fna()` κι από ποιες όταν εκτελεί την `Fnb()`; Ποιο γεγονός προκαλεί μετάβαση της διεργασίας από κατάσταση σε κατάσταση σε κάθε περίπτωση; Υποθέστε ότι υπάρχουν κι άλλες διεργασίες στο σύστημα κι ο αλγόριθμος χρονοδρομολόγησης είναι Round-Robin.

## Θέμα 2 (25%)

Δίνονται λειτουργίες δομής πίνακα κατακερματισμού (hash table), που αποθηκεύει σύνολο κλειδιών. Η δομή υλοποιείται με πίνακα `N` θέσεων. Σε κάθε θέση είναι αποθηκευμένη μια λίστα. Ο κάθε κόμβος της λίστας περιέχει ένα κλειδί (key).

```
list table[N];
```

```
void insert(int key){
  list a;
  a = table[key % N];
  append(a, key);
}
```

```
void remove(int key){
  list a;
  a = table[key % N];
  delete(a, key);
}
```

```
int exists(int key){
  list a;
  a = table[key % N];
  return contains(a, key);
}
```

**Σημείωση #1**: Οι κλήσεις χειρισμού της λίστας:

- `append()`: προσθήκη στοιχείου στη λίστα
- `delete()`: διαγραφή στοιχείου από τη λίστα
- `contains()`: έλεγχος αν ένα στοιχείο υπάρχει στη λίστα

**ΔΕΝ** περιέχουν κάποιου είδους συγχρονισμό. Σε περίπτωση αμφιβολίας για το πώς υλοποιούνται καλείστε να είστε συντηρητικοί!

**Σημείωση #2**: Για απλότητα θεωρούμε ότι δεν υπάρχει περίπτωση η `insert()` να κληθεί με τιμή κλειδιού που υπάρχει ήδη στον πίνακα και ότι δεν υπάρχει περίπτωση η `remove()` να κληθεί με τιμή κλειδιού που δεν υπάρχει στον πίνακα.

**α. (5%)** Υλοποιήστε σχήμα συγχρονισμού για (όλες) τις παραπάνω λειτουργίες χρησιμοποιώντας κεντρικό κλείδωμα.

**β. (10%)** Ποιό είναι το μειονέκτημα του κεντρικού κλειδώματος; Υλοποιήστε σχήμα συγχρονισμού πολλαπλών κλειδωμάτων που να το αντιμετωπίζει.

**γ. (10%)** Στο σχήμα συγχρονισμού του ερωτήματος **β** ζητείται να υλοποιηθεί συνάρτηση `insertremove(key1,key2)` που *ατομικά* εισάγει το `key1` και διαγράφει το `key2`.
Σωστή ατομική υλοποίηση της παραπάνω λειτουργίας συνεπάγεται ότι, δεδομένου ότι το `key2` υπάρχει στη δομή, η έκφραση `(!exists(key2)) && (!exists(key1))` δεν επιστρέφει *ποτέ* `true`. *ΠΡΟΣΟΧΗ:* Βεβαιωθείτε ότι η υλοποίησή σας δεν εμφανίζει deadlocks.

## Θέμα 3 (20%)

**α. (10%)** Έστω σύστημα με περισσότερους του ενός επεξεργαστές. Η χρονοδρομολόγηση υλοποιείται με μια ουρά χρονοδρομολόγησης ανά επεξεργαστή. Για ευκολία θεωρούμε ότι αρχικά κάθε ουρά περιέχει μια διεργασία.
Έστω δύο τακτικές χρονοδρομολόγησης:

α′ Κάθε διεργασία που δημιουργείται παραμένει μέχρι την ολοκλήρωσή της στην ουρά, στην οποία άνηκε ο γονιός της.

β′ Κατά την δημιουργία μιας διεργασίας ο χρονοδρομολογητής της ουράς του γονιού της ελέγχει όλες τις υπόλοιπες ουρές, και την τοποθετεί σε αυτή με το μικρότερο αριθμό διεργασίων.

Αναφέρατε ένα πλεονέκτημα της τακτικής (α') έναντι της (β'), και ένα πλεονέκτημα της τακτικής (β') έναντι της (α'). Μπορείτε, αν κρίνετε σκόπιμο, να επικαλεστείτε παραδείγματα.

**β. (10%)**

i. Αναφέρατε ένα πλεονέκτημα κι ένα μειονέκτημα της χρήσης μεγάλου μεγέθους μπλοκ δεδομένων σε ένα σύστημα αρχείων.

ii. Θεωρούμε ότι το FCB (i-node) ενός συστήματος αρχείων περιέχει ένα μετρητή (ακέραιο αριθμό) για την υλοποίηση *μη συμβολικών* συνδέσμων (hard links). Περιγράψτε συνοπτικά τι συνεπάγεται η εκτέλεση των παρακάτω λειτουργιών για τις δομές i-node που υπάρχουν και τις τιμές των μετρητών που περιέχουν.

- Δημιουργία νέου αρχείου
- Δημιουργία νέου hard link προς υπάρχον αρχείο
- Διαγραφή αρχείου (π.χ. `rm file1`).

## Θέμα 4 (30%)

**a. (15%)** Μια διεργασία διαθέτει τρία πλαίσια φυσικής μνήμης κι εκτελεί την παρακάτω ακολουθία αναφορών σε σελίδες εικονικής μνήμης:

**1, 3, 2, 1, 4, 2, 5, 3, 2, 1, 5**

Και τα τρία πλαίσια είναι αρχικά άδεια. Εκτελέστε τον αλγόριθμο αντικατάστασης σελίδας και βρείτε τον αριθμό των σφαλμάτων σελίδας που συμβαίνουν όταν η στρατηγική είναι:

i. First-In, First-Out (FIFO)

ii. Least Recently Used (LRU)

Ένας συμφοιτητής σας ισχυρίζεται ότι έχει βρει στρατηγική αντικατάστασης σελίδων η οποία εμφανίζει 5 σφάλματα σελίδας για τη συγκεκριμένη ακολουθία. Τι του απαντάτε;

**β. (5%)** Η υλοποίηση αλγορίθμων αντικατάστασης σελίδας με προσέγγιση LRU διευκολύνεται σημαντικά όταν η μηχανή διαθέτει bit αναφοράς (reference bit) για κάθε σελίδα εικονικής μνήμης. Έστω μηχανή η οποία δεν διαθέτει αυτή τη δυνατότητα. Σκιαγραφήστε στρατηγική με την οποία το ΛΣ μπορεί να προσομοιώσει τη λειτουργικότητα του reference bit και να γνωρίζει σε ποιες σελίδες έγινε αναφορά μέσα σε ένα συγκεκριμένο χρονικό διάστημα.

**γ. (10%)** Αναφέρατε πέντε χαρακτηριστικά ενός σύγχρονου Λειτουργικού Συστήματος τα οποία είτε βασίζονται για την υλοποίησή τους είτε επωφελούνται από συγκεκριμένους μηχανισμούς του Υλικού. Για κάθε ένα από τα χαρακτηριστικά αυτά μιλήστε *περιληπτικά*, με μία πρόταση, για τον τρόπο με τον οποίο υποστηρίζονται από τον αντίστοιχο μηχανισμό του Υλικού.