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

Επαναληπτική Εξέταση ακ. έτους 2010-2011
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%)

Δίνεται το ακόλουθο τμήμα κώδικα:

```
int i, j, n, fd[3][2];
pid_t p[3];

for (i = 0; i < 3; i++) pipe(fd[i]);

for (i = 0; i < 3; i++) {
    p[i] = fork();
    if (p[i] == 0) {
        for (j = i; j < 3; j++)
            close(fd[j][1]);
        sleep(5);

        n = 3 - i;
        if (i > 0)
            write(fd[i - 1][1], &n, sizeof(n));

        read(fd[i][0], &n, sizeof(n));
        for (j = 0; j < n; j++)
```

```
            if (fork() == 0)
                Fn(i, j);

        wait(NULL);
        Fn(i, 5);
    }
}

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

kill(p[2], SIGKILL);
close(fd[1][1]);
/* A */
wait(NULL);
Fn(i, n);
```

Οι κλήσεις συστήματος δεν αποτυγχάνουν κι η `Fn()` δεν επιστρέφει ποτέ. Θεωρήστε ότι η αρχική διεργασία φτάνει στο σημείο `/* A */` σε πολύ λιγότερο από 5 δευτερόλεπτα. Δεδομένου ότι η `Fn()` δεν επιστρέφει ποτέ, τελικά οι διεργασίες που δημιουργεί το παραπάνω πρόγραμμα έρχονται σε μια *μόνιμη* κατάσταση: το δέντρο διεργασιών μένει σταθερό για πάντα, και κάθε διεργασία εκτελεί συγκεκριμένη συνάρτηση από τον κώδικά της.

α. (12%) Για την *τελική* κατάσταση: σχεδιάστε το δέντρο διεργασιών που προκύπτει. Εξηγήστε *συνοπτικά* το σκεπτικό πίσω από τη μορφή του.

β. (4%) Για κάθε κόμβο του δέντρου, γράψτε την κλήση συστήματος ή συνάρτηση που εκτελεί η αντίστοιχη διεργασία, μαζί με τα ορίσματά της.

γ. (4%) Συμπληρώστε το δέντρο διεργασιών ώστε να φαίνεται η διαδιεργασιακή επικοινωνία: για κάθε μεταφορά δεδομένων που συνέβη, σχεδιάστε ένα διακεκομμένο βέλος από τη διεργασία-αποστολέα στη διεργασία-παραλήπτη. Πάνω στο βέλος γράψτε την τιμή ή τιμές που μεταφέρονται κάθε φορά.

δ. (5%) Απαντήστε *συνοπτικά*, όχι πάνω από δύο γραμμές, στα ακόλουθα:

i. Τι κάνει η κλήση συστήματος `wait()`;

ii. Πότε μια διεργασία γίνεται zombie; Τι πρέπει να συμβεί για να εξαφανιστεί από το σύστημα;

iii. Τι συμβαίνει στα ανοιχτά αρχεία μιας διεργασίας όταν αυτή πεθάνει κανονικά κι όταν σκοτωθεί;

iv. Τι επιστρέφει η `read()` σε κάποιον που διαβάζει από ένα pipe όταν κανείς δεν έχει πλέον ανοιχτό το άκρο εγγραφής;

Θέμα 2 (25%)

α. (5%) Δύο φίλοι πηγαίνουν για ψάρεμα με μια βάρκα, έστω ότι αναπαρίστανται από τις διεργασίες `friend_A` και `friend_B`. Καθένας από τους δύο κάνει τα εξής: περπατά μέχρι την αποβάθρα (συνάρτηση `walk_to_dock()`), αν ο άλλος δεν είναι ήδη εκεί τον περιμένει (σημείο `...rendezvous...`), μπαίνει στη βάρκα (`board()`) και φεύγουν μαζί.

```
void friend_A()
{
    walk_to_dock();
    ...rendezvous...
    board();
}
```

```
void friend_B()
{
    walk_to_dock();
    ...rendezvous...
    board();
}
```

Στα σημεία που υποδηλώνονται (`...rendezvous...`), σας ζητείται να υλοποιήσετε σχήμα συγχρονισμού, μόνο με χρήση `signal` και `wait()` σε κατάλληλα αρχικοποιημένους σημαφόρους, έτσι ώστε οι φίλοι να μπαίνουν στη βάρκα μόνο όταν κι οι δύο έχουν φτάσει στην αποβάθρα.

β. (10%) Έστω ότι τρεις φίλοι `friend_A`, `friend_B`, `friend_C` πηγαίνουν για ψάρεμα με βάρκα που χωράει τρία άτομα. Ομοίως με το προηγούμενο ερώτημα, υλοποιήστε κατάλληλο σχήμα συγχρονισμού *μόνο* με κλήσεις `signal()` και `wait()` σε σημαφόρους, ώστε να μπαίνουν στη βάρκα μόνο όταν κι οι τρεις έχουν φτάσει στην αποβάθρα.

γ. (10%) Έστω ότι N φίλοι πηγαίνουν για ψάρεμα, με μία βάρκα που χωράει N επιβάτες. Μπαίνουν στη βάρκα *μόνο* όταν *όλοι* έχουν φτάσει στην αποβάθρα. Ένας από αυτούς ορίζεται καπετάνιος· αφού μπει στη βάρκα, την οδηγεί (`steer_boat()`).
Όλοι εκτελούν ακριβώς την ίδια συνάρτηση:

```
shared int N;

void friend()
{
    bool captain = false;

    walk_to_dock();
    ...rendezvous...
    board();
    if (captain)
        steer_boat();
}
```

Σας ζητείται η υλοποίηση σχήματος συγχρονισμού στο σημείο που υποδεικνύεται, κοινό για όλες τις συναρτήσεις, έτσι ώστε να τηρούνται οι κανόνες τις εκδρομής. Ο κώδικάς σας πρέπει να θέτει τη σημαία `captain` *μόνο* σε μία από τις διεργασίες, της δικής σας επιλογής. Μπορείτε να χρησιμοποιήσετε κλήσεις `signal()` και `wait()` σε κατάλληλα αρχικοποιημένους σημαφόρους, καθώς και μεταβλητές μοιραζόμενες ανάμεσα στις διεργασίες. Το πλήθος των φίλων σας δίνεται στη μεταβλητή `N`.
Υπόδειξη: Οι φίλοι κρατούν σε κατάλληλα αρχικοποιημένη μοιραζόμενη μεταβλητή πόσοι έχουν φτάσει μέχρι τώρα στην αποβάθρα.

Θέμα 3 (25%)

α. (5%) Τι είναι εναλλαγή περιβάλλοντος (context switch); Τι πληροφορίες αποθηκεύει και επαναφέρει το ΛΣ για ένα context switch; Πώς διαφέρει η εναλλαγή περιβάλλοντος ανάμεσα σε διεργασίες από την εναλλαγή ανάμεσα σε νήματα της ίδιας διεργασίας;

β. (5%) Αναφέρατε τέσσερα κριτήρια αξιολόγησης για έναν αλγόριθμο χρονοδρομολόγησης. Περιγράψτε *πολύ* συνοπτικά πώς ορίζεται το καθένα.

γ. (10%) Σχολιάστε την επίδραση του μεγέθους του κβάντου χρόνου που χρησιμοποιείται για χρονοδρομολόγηση στα εξής:

i. Αποκρισιμότητα του συστήματος

ii. Επιβάρυνση της ΚΜΕ με λειτουργίες χρονοδρομολόγησης

iii. Hit rate της κρυφής μνήμης (cache)

Αν θέλετε μπορείτε να χρησιμοποιήσετε παραδείγματα.

δ. (5%) Πότε μια διεργασία λέγεται I/O-bound και πότε CPU-bound; Πού κατατάσσονται οι εξής διεργασίες και γιατί; (α′) Ο vi όταν επεξεργάζεστε ένα κείμενο (β′) Το Photoshop όταν η εικόνα είναι μικρότερη της φυσικής μνήμης (γ′) Το 3D Studio όταν υπολογίζει μια φωτορεαλιστική σκηνή (δ′) Το Photoshop όταν η εικόνα δεν χωράει στη φυσική μνήμη (ε′) Μια μηχανή σκακιού

Θέμα 4 (25%)

α. (10%) Απαντήστε *συνοπτικά*, όχι πάνω από δύο-τρεις γραμμές:

i. Με ποιον τρόπο βοηθά η υποστήριξη modify bit από το υλικό την υλοποίηση αποδοτικού μηχανισμού για σελιδοποίηση κατ'απαίτηση από το ΛΣ;

ii. Τι θα γινόταν αν μπορούσε μια διεργασία χρήστη να ελέγξει τον χρονιστή (timer) του συστήματος; πώς αποτρέπεται αυτό το ενδεχόμενο;

iii. Υπάρχει περίπτωση μια εικονική διεύθυνση να είναι έγκυρη διεύθυνση για μια διεργασία (π.χ. έχει επιστραφεί από τη `malloc()`), και παρ'όλα αυτά η αντίστοιχη σελίδα να μην διαθέτει έγκυρη απεικόνιση στον πίνακα σελίδων; Τι συμβαίνει αν η διεργασία κάνει αναφορά στη συγκεκριμένη σελίδα;

β. (10%)

i. Σχεδιάστε ένα ενδεικτικό διάγραμμα μετάβασης καταστάσεων διεργασίας σε ΛΣ.

ii. Από ποια σε ποια κατάσταση μεταβαίνει μια διεργασία όταν εκτελεί `sleep()`; Πότε φεύγει από την κατάσταση στην οποία μπαίνει με `sleep()` και σε ποια πηγαίνει;

iii. Μια διεργασία είναι "Υπό Εκτέλεση" στον επεξεργαστή και πηγαίνει σε κατάσταση "Έτοιμη" χωρίς να εκτελέσει κλήση συστήματος. Τι συνέβη; Πότε θα φύγει από εκεί;

iv. Μια διεργασία είναι "Υπό Εκτέλεση" και γίνεται "Σε Αναμονή" χωρίς να εκτελέσει κλήση συστήματος. Τι συνέβη; Πότε θα φύγει από εκεί;

γ. (5%) Έστω συγκεκριμένος υπολογισμός που εκτελείται με δεδομένο αριθμό πλαισίων F σε σύστημα εικονικής μνήμης και προκαλεί p σφάλματα σελίδας. Αν αυξήσουμε τον αριθμό των πλαισίων σε $F' > F$, υπάρχει περίπτωση ο νέος αριθμός σφαλμάτων σελίδας να είναι $p' > p$; θεωρήστε στρατηγική αντικατάστασης σελίδας

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

ii. Least Recently Used (LRU)

Δικαιολογήστε *συνοπτικά* την απάντησή σας.