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, Ακαδημαϊκό Έτος 2011-2012

Κανονική Εξέταση

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 πρόγραμμα C σε περιβάλλον UNIX που ακολουθεί μετά τα ζητούμενα.
Θεωρήστε ότι οι κλήσεις συστήματος δεν αποτυγχάνουν, η `Fn()` δεν επιστρέφει ποτέ, η `Fn()` δεν εκτελεί κλήσεις συστήματος, όταν μια διεργασία δημιουργείται κληρονομεί ακριβώς τον τρόπο χειρισμού σημάτων του πατέρα της και τέλος ότι οι κλήσεις συστήματος διακόπτονται από εισερχόμενα σήματα.
Δεδομένου ότι η `Fn()` δεν επιστρέφει ποτέ, οι διεργασίες που δημιουργεί το πρόγραμμα έρχονται σε *μόνιμη* κατάσταση: το δέντρο διεργασιών μένει σταθερό για πάντα και κάθε διεργασία εκτελεί συγκεκριμένη συνάρτηση ή κλήση συστήματος.

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

i. Τι κάνει η κλήση συστήματος `pipe()` στο UNIX; Πόσες και τι τύπου τιμές δίνει στο πρόγραμμα που την καλεί; Με ποιον τρόπο επιστρέφει αυτές τις τιμές και πού τις γράφει;

ii. Τι παθαίνει μια διεργασία που καλεί την `read()` σε άδειο pipe όταν είναι ανοιχτό το άκρο εγγραφής;

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

iv. Τι επιστρέφουν οι κλήσεις `getpid()`, `getppid()` στο UNIX; Υπάρχει περίπτωση οι τιμές που επιστρέφουν να αλλάξουν για την ίδια διεργασία κατά τη διάρκεια της ζωής της;

β. (12%) Για την *τελική* κατάσταση των διεργασιών του προγράμματος: σχεδιάστε το δέντρο διεργασιών τότε. Εξηγήστε *συνοπτικά* πώς προκύπτει.

γ. (4%) Για κάθε κόμβο του δέντρου, γράψτε: (i) την κλήση συστήματος ή συνάρτηση μέσα στην οποία βρίσκεται ο PC της αντίστοιχης διεργασίας, (ii) τα ορίσματα με τα οποία αυτή έχει κληθεί, (iii) τη γραμμή του προγράμματος απ' όπου έγινε η κλήση της.
Για τα ορίσματα, κάντε οποιαδήποτε υπόθεση χρειάζεστε για νούμερα που δεν γνωρίζετε, π.χ. PIDs που ανατίθενται από το σύστημα στις νέες διεργασίες.

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

```
1  void handler(int signum)
2  { Fn(0, -1); }
3
4  int main(void)
5  {
6      int i, p[2], writefd[4];
7      pid_t pid[4];
8
9      signal(SIGUSR1, handler);
10
11     for (i = 0; i < 4; i++) {
12         pipe(p);
13         pid[i] = fork();
14         if (pid[i] == 0) {
15             close(p[1]);
16             read(p[0], &pid[i], sizeof(pid_t));
17             if (pid[i] > 0) kill(pid[i], SIGUSR1);
18             Fn(i, pid[i]);
19         }
20         close(p[0]);
21         writefd[i] = p[1];
22     }
23
24     write(writefd[1], &pid[0], sizeof(pid_t));
25     close(writefd[3]);
26     wait(NULL);
27     write(writefd[2], &pid[1], sizeof(pid_t));
28     Fn(-1, -2);
29     return 0;
30 }
```

Θέμα 2 (25%)

α. (10%) Σε ένα γραφείο μπορούν να βρίσκονται είτε οι εργαζόμενοι που δουλεύουν εκεί, είτε η καθαρίστρια που επιμελείται το χώρο. Οι εργαζόμενοι εργάζονται μέσα στο χώρο (`work_for_a_while()`) και κάνουν περιοδικά διάλειμμα στην αυλή (`take_a_break()`). Η καθαρίστρια δεν μπαίνει ποτέ στο γραφείο όσο υπάρχουν υπάλληλοι εκεί· αντίστοιχα, οι εργαζόμενοι δεν μπαίνουν στο γραφείο όσο η καθαρίστρια καθαρίζει.
Οι εργαζόμενοι αναπαρίστανται από διεργασίες που εκτελούν τη διαδικασία `worker()`, η καθαρίστρια αναπαρίσταται από διεργασία που εκτελεί τη διαδικασία `cleaner()`.

```
void worker()
{
    for (;;) {
        . . .
        work_for_a_while();
        . . .
        take_a_break();
    }
}
```

```
void cleaner()
{
    for (;;) {
        . . .
        clean_the_room();
        . . .
        take_a_break();
    }
}
```

Στα σημεία που υποδηλώνονται με ". . ." ζητείται να υλοποιήσετε σχήμα συγχρονισμού ώστε να εξασφαλίζεται ο τρόπος εργασίας που περιγράφηκε. Μπορείτε να χρησιμοποιήσετε κλήσεις `signal()` και `wait()` σε κατάλληλα αρχικοποιημένους σημαφόρους κι όποιες μεταβλητές μοιραζόμενες ανάμεσα στις διεργασίες χρειάζεστε.
Στη λύση που δώσατε, υπάρχει ενδεχόμενο λιμοκτονίας; Σχολιάστε.

β. (5%) Το γραφείο του προηγούμενου ερωτήματος διαθέτει ακριβώς N θέσεις εργασίας. Τροποποιήστε το σχήμα συγχρονισμού του προηγούμενου ερωτήματος, ώστε το πολύ N εργαζόμενοι να μπορούν να βρίσκονται μέσα στο χώρο εργασίας, να εκτελούν δηλαδή την `work_for_a_while()` ταυτόχρονα.

γ. (10%) Επειδή οι ανάγκες καθαρισμού αυξήθηκαν, προσλαμβάνονται περισσότερες καθαρίστριες, οι οποίες μπορούν να εργάζονται ταυτόχρονα στον ίδιο χώρο. Και πάλι, μόνο όταν δεν υπάρχουν εργαζόμενοι εκεί.
Τροποποιήστε κατάλληλα το σχήμα συγχρονισμού του προηγούμενου ερωτήματος, ώστε στο χώρο να μπορούν να βρίσκονται ταυτόχρονα είτε το πολύ N εργαζόμενοι είτε το πολύ M καθαρίστριες.

Θέμα 3 (25%)

α. (5%) Σχεδιάστε ενδεικτικό διάγραμμα καταστάσεων διεργασίας, με τουλάχιστον τις καταστάσεις RUNNING, WAITING, READY, TERMINATED.

β. (5%) Αναφέρετε μια αιτία για κάθε μετάβαση.

γ. (5%) Σε ποιες μεταβάσεις λαμβάνεται απόφαση χρονοδρομολόγησης από το ΛΣ; Πού διαφέρει ένας διακοπτός (preemptive) από ένα μη-διακοπτό/συνεργατικό (non-preemptive/ cooperative) αλγόριθμο χρονοδρομολόγησης σε σχέση με το πότε παίρνονται αποφάσεις χρονοδρομολόγησης; Σχολιάστε τη συμπεριφορά τους όταν συμβαίνουν διακοπές λογισμικού και υλικού (software και hardware interrupts).

δ. (5%) Αναφέρετε ένα πλεονέκτημα κι ένα μειονέκτημα για κάθε μία από τις δύο στρατηγικές (διακοπτή χρονοδρομολόγηση ή μη-διακοπτή). Ποια από τις δύο θα διαλέγατε για ένα ΛΣ με γραφικό περιβάλλον που εκτελεί εφαρμογές που έχουν συχνή αλληλεπίδραση με το χρήστη;

ε. (5%) Δουλεύετε στον editor `vi` σε μονοεπεξεργαστικό σύστημα UNIX, ενώ μεταγλωττίζετε μεγάλο πρόγραμμα. Πιέζετε ένα πλήκτρο. Η ενέργειά σας αυτή μπορεί να προκαλέσει context switch όταν το ΛΣ χρησιμοποιεί (i) διακοπτό (ii) μη-διακοπτό αλγόριθμο χρονοδρομολόγησης; Αν ναι, περιγράψτε με βήματα τι συμβαίνει στο σύστημα και καταλήγει σε context switch.

Θέμα 4 (25%)

α. (15%) Μια διεργασία εκτελεί το ακόλουθο τμήμα κώδικα σε σύστημα UNIX εικονικής μνήμης με σελιδοποίηση. Θεωρήστε ότι οι σελίδες που περιέχουν τον κώδικά της και τις μεταβλητές `i`, `n`, `sum`, `len`, `total`, είναι πάντοτε στη φυσική μνήμη. Καμία κλήση δεν αποτυγχάνει. Υπάρχουν κι άλλες διεργασίες στο σύστημα κι ο αλγόριθμος χρονοδρομολόγησης είναι RR. Το σύστημα έχει 16GB φυσικής μνήμης.

```
1   long i, n, sum, len, total = 1024 * 1048576;
2   char *p, *buf = malloc(total);
3
4   fd = open("a_file_larger_than_1GB", O_RDONLY);
5   p = buf; len = total;
6   while (len > 0) { /* 1GB total */
7       n = read(fd, p, 1048576); /* 1MB */
8       p += n; len -= n;
9   }
10
11  for (sum = 0, i = 0; i < total; i++) sum += i;
12  sleep(60);
13  for (i = 0; i < total; i++) buf[i]++;
```

Απαντήστε *συνοπτικά*, όχι πάνω από δύο-τρεις γραμμές, στις παρακάτω ερωτήσεις για τη συμπεριφορά της διεργασίας. Κάντε όποιες ρεαλιστικές υποθέσεις χρειάζεστε, αρκεί να τις καταγράψετε. Σε *κάθε* περίπτωση δικαιολογήστε θετική απάντησή σας με αντίστοιχο σενάριο/παράδειγμα.

i. Ποιες κλήσεις συστήματος εκτελεί η διεργασία στις γραμμές 4–9; Γίνεται να υποστεί μετάβαση RUNNING → WAITING κατά την εκτέλεσή τους;

ii. Γίνεται να μην πάει σε WAITING καθ' όλη την εκτέλεση των 4–9;

iii. Γίνεται να *μην* παραμείνει RUNNING καθ' όλη την εκτέλεση της 11;

iv. Γίνεται να παραμείνει RUNNING κατά την εκτέλεση της 12;

v. Ποιες κλήσεις συστήματος εκτελεί στη γραμμή 13; Ποιες είναι όλες οι δυνατές μεταβάσεις κατά την εκτέλεση της 13; Είναι δυνατή μετάβαση RUNNING → WAITING; Γιατί; Θεωρήστε καταστάσεις RUNNING, WAITING, READY, TERMINATED.

β. (10%) Σε ΛΣ με σελιδοποίηση κατ'απαίτηση (demand paging), επιχειρούμε να τρέξουμε με λάθος παραμέτρους ένα πολύ μεγάλο εκτελέσιμο, της τάξης των 400MB. Όταν το εκτελέσιμο τρέξει, εκτυπώνει μήνυμα λάθους για τις παραμέτρους κι η διεργασία τερματίζει. Τα μπλοκ του δίσκου που το περιέχουν δεν έχουν προσπελαστεί ποτέ από την εκκίνηση του ΛΣ. Το ΛΣ δεν γνωρίζει τίποτε για την εσωτερική οργάνωση του κώδικα του εκτελέσιμου. Ο δίσκος υποστηρίζει ρυθμό μεταφοράς 20MB/s προς την κύρια μνήμη.

i. Πόσο χρόνο τουλάχιστον θα χρειαζόταν η ανάγνωση του εκτελέσιμου από το δίσκο στην κύρια μνήμη;

ii. Περιγράψτε μηχανισμό του ΛΣ με τον οποίο μια διεργασία μπορεί να προσπελάσει δεδομένα αρχείου στο δίσκο, χωρίς την εκτέλεση κλήσεων συστήματος `read()`/`write()`.

iii. Περιγράψτε σενάριο όπου το εκτελέσιμο τρέχει, παράγει το μήνυμα λάθους και τερματίζει, σε πολύ λιγότερο από μερικά δευτερόλεπτα.