Λειτουργικά Συστήματα

2019 - 2020

1η Εργαστηριακή Άσκηση

Μέρος 1 [30 μονάδες]

Ερώτημα Α [5]: Εξηγήστε προσεκτικά τι κάνει το παρακάτω πρόγραμμα. Πόσα μυνήματα εκτυπώνονται συνολικά?

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
      int i;
      int pid;

      pid = fork();

      for (i=1; i<=200; i++)
          if (pid > 0)
              printf("%3i (parent)\n", i);
            else
              printf("%3i (child)\n", i);


      return (0);
}
```

Ερώτημα Β [5]:: Δημιουργείστε ένα πρόγραμμα στο οποίο μία διεργασία στο Linux/Unix παράγει άλλες 10 θυγατρικές της.

Ερώτημα Γ [5]: Να δημιουργήσετε ένα πρόγραμμα στο οποίο να δημιουργούνται 10 διεργασίες (αλυσίδα – κάθε μία να είναι παιδί της άλλης). Κάθε διεργασία να τυπώνει, το id του πατέρα της, το id της και το id του μοναδικού παιδιού που δημιουργεί.

Ερώτημα Δ [15]: Να δημιουργήσετε πρόγραμμα στο οποίο:

1. Θα ορίσετε μια συνάρτηση foo() η οποία θα ορίζει μια μεταβλητή int x=0 και θα κάνει την πράξη x=x+10. Αυτή η συνάρτηση δεν κάνει τίποτα χρήσιμο, αλλά απλώς υπάρχει για να μας βοηθήσει στη μέτρηση.
2. Μέσα στη main() θα εκτελεί μια φορά τη συνάρτηση time() με τις κατάλληλες παραμέτρους, θα αποθηκεύεται ο αριθμός των δευτερολέπτων στη μεταβλητή start και αμέσως μετά θα εκτυπώνεται το μήνυμα “Αρχική τιμή δευτερολέπτων” ακολουθούμενο από τη start.
3. Στη συνέχεια θα υπάρχει ένας βρόχος while() ο οποίος θα δημιουργεί 100 διεργασίες (δοκιμάστε και για 5000, 10000) με τη fork() οι οποίες όλες θα εκτελούν τη foo(). **Προσοχή, ο πατέρας θα δημιουργήσει όλες τις διεργασίες και όχι η κάθε διεργασία θα δημιουργεί άλλη διεργασία.**
4. Μόλις δημιουργηθούν οι 100 διεργασίες και μόνο τότε θα εκτελεί ο πατέρας 100 φορές τη waitpid() ώστε να περιμένει την επιτυχή ολοκλήρωση όλων των θυγατρικών.

5. Στη συνέχεια θα καλείται η time(), θα αποθηκεύεται ο αριθμός των δευτερολέπτων στη μεταβλητή end και αμέσως μετά θα εκτυπώνεται το μήνυμα "Τελική τιμή δευτερολέπτων" ακολουθούμενη από την end. Θα γίνεται η πράξη end-start, θα εκτυπώνεται το αποτέλεσμα ενώ θα εκτυπώνεται και το αποτέλεσμα «(end-start)/100» για να εμφανιστεί ο μέσος χρόνος δημιουργίας, εκτέλεσης και τερματισμού των 100 διεργασιών.

Πόσος είναι ο συνολικός χρόνος και ο μέσος χρόνος δημιουργίας, εκτέλεσης και τερματισμού των 100 διεργασιών στο σύστημα σας (μη ξεχάσετε τη μονάδα μέτρησης);

*****Μπορείτε εναλλακτικά να χρησιμοποιήσετε τους κύκλους της CPU μετρώντας τους με την συνάρτηση clock() και στην συνέχεια να τους διαιρέσετε με την built-in σταθερά CLOCKS_PER_SEC ώστε να πάρετε τον αριθμό των δευτερολέπτων.**

Μέρος 2 [70 μονάδες]

Ερώτημα Α [20]

Θεωρήστε τις διεργασίες Process1, Process2 και Process3 που εκτελούνται «παράλληλα» (δηλ. εκτελούνται σύμφωνα με το σχήμα εκτέλεσης "cobegin Process1; Process2; Process3; coend"). Κάθε διεργασία εκτελεί επαναληπτικά (για 10 φορές) δύο εντολές. Συγκεκριμένα:

- Η διεργασία Process1 εκτελεί επαναληπτικά για 10 φορές την εντολή E1.1 και την εντολή E1.2.
- Η διεργασία Process2 εκτελεί επαναληπτικά για 10 φορές την εντολή E2.1 και την εντολή E2.2.
- Η διεργασία Process3 εκτελεί επαναληπτικά για 10 φορές την εντολή E3.1 και την εντολή E3.2.

Ο κώδικας των διεργασιών δίνεται ακολούθως:

```
     Process1              Process2              Process3
for k = 1 to 10 do    for j = 1 to 10 do    for l = 1 to 10 do
  begin                 begin                 begin
     E1.1;                 E2.1;                 E3.1;
     E1.2;                 E2.2;                 E3.2;
  end                   end                   end
```

Καλείστε να συμπληρώσετε τον παραπάνω κώδικα των τριών διεργασιών με εντολές P (ή wait ή down) και V (ή signal ή up) σε σημαφόρους (που πρέπει να αρχικοποιήσετε κατάλληλα), προκειμένου να εξασφαλίζονται οι ακόλουθες συνθήκες συγχρονισμού για τη σειρά εκτέλεσης των εντολών των διεργασιών:

1. Στην i επανάληψη (1<=i<=10) της Process2, η εκτέλεση της E2.1 να έπεται της εκτέλεσης της E1.1 στην ίδια επανάληψη i της Process1. Π.χ. στην $5^η$ επανάληψη της Process2, η E2.1 να εκτελείται μετά από την E1.1 στην $5^η$ επανάληψη της Process1.
2. Στην i επανάληψη (1<=i<=10) της Process3, η εκτέλεση της E3.1 να έπεται της εκτέλεσης της E2.1 στην ίδια επανάληψη i της Process2. Π.χ. στην $5^η$ επανάληψη της Process3, η E3.1 να εκτελείται μετά από την E2.1 στην $5^η$ επανάληψη της Process2.
3. Στην i επανάληψη (1<=i<=10) της Process2, η εκτέλεση της E2.2 να έπεται της εκτέλεσης της E1.2 στην ίδια επανάληψη i της Process1. Π.χ. στην $5^η$ επανάληψη της Process2, η E2.2 να εκτελείται μετά από την E1.2 στην $5^η$ επανάληψη της Process1.

4. Στην i επανάληψη (1<=i<=10) της Process3, η εκτέλεση της E3.2 να έπεται της εκτέλεσης της E2.2 στην ίδια επανάληψη i της Process2. Π.χ. στην 5η επανάληψη της Process3, η E3.2 να εκτελείται μετά από την E2.2 στην 5η επανάληψη της Process2.

5. Στην i επανάληψη (2<=i<=10) της Process1, η εκτέλεση της E1.1 να έπεται της εκτέλεσης της E3.2 στην προηγούμενη επανάληψη (i-1) της Process3. Π.χ. στην 5η επανάληψη της Process1, η E1.1 να εκτελείται μετά από την E3.2 στην 4η επανάληψη της Process3.

Σημείωση: Στην πρώτη επανάληψη (i=1) εκτελείται αρχικά μόνο η εντολή E1.1 της Process1 (στην 1η επανάληψη η E1.1 είναι η μόνη που δεν υπόκειται σε κάποιο περιορισμό εκτέλεσης).

(i) Δώστε αρχικά μία λύση χρησιμοποιώντας πέντε (5) σημαφόρους (έναν για κάθε συνθήκη συγχρονισμού που αναφέρεται παραπάνω). **[12]**

(ii) Στη συνέχεια δώστε μια λύση προσπαθώντας να χρησιμοποιήσετε τον ελάχιστο (κατά την κρίση σας) δυνατό αριθμό σημαφόρων που αρκεί για να επιτευχθούν οι προαναφερόμενες συνθήκες συγχρονισμού. **[8]**

Ερώτημα Β [10]

Ένα λειτουργικό σύστημα εξυπηρετεί 5 παράλληλες διεργασίες Δi (i = 1..5). Οι 5 διεργασίες εκτελούνται συνεχώς (δηλ. ο κώδικας της κάθε μιας βρίσκεται εντός ενός αδιάκοπου βρόγχου). Το λειτουργικό σύστημα προσφέρει τη χρήση σημαφόρων. Στη συνέχεια δίνεται η περιγραφή των 5 διεργασιών.

Διεργασία Δ1: Η διεργασία Δ1 βρίσκει **έναν** τυχαίο αριθμό από το διάστημα 1..10 και τον γράφει στον απομονωτή (buffer) buf1.
Διεργασία Δ2: Η διεργασία Δ2 διαβάζει από τον απομονωτή buf1 την τιμή που έχει γράψει **στον ίδιο κύκλο επανάληψης**, η διεργασία Δ1, και γράφει την τιμή που διάβασε στον απομονωτή buf2.
Διεργασία Δ3: Η διεργασία Δ3 διαβάζει από τον απομονωτή buf1 την τιμή που έχει γράψει **στον ίδιο κύκλο επανάληψης**, η διεργασία Δ1, και γράφει την τιμή που διάβασε στον απομονωτή buf3.
Διεργασία Δ4: Η διεργασία Δ4 διαβάζει από τον απομονωτή buf2 την τιμή που έχει γράψει **στον ίδιο κύκλο επανάληψης**, η διεργασία Δ2. Στη συνέχεια βρίσκει μια τυχαία τιμή από το διάστημα 1..10 και την προσθέτει στην τιμή που διάβασε από τον απομονωτή buf2. Αν το αποτέλεσμα είναι μεγαλύτερο του αντίστοιχου αποτελέσματος (στον ίδιο κύκλο επανάληψης) της διεργασίας Δ5, τότε εκτυπώνει στη οθόνη το όνομά της.
Διεργασία Δ5: Η διεργασία Δ5 διαβάζει από τον απομονωτή buf3 την τιμή που έχει γράψει **στον ίδιο κύκλο επανάληψης**, η διεργασία Δ3. Στη συνέχεια βρίσκει μια τυχαία τιμή από το διάστημα 1..10 και την προσθέτει στην τιμή που διάβασε από τον απομονωτή buf3. Αν το αποτέλεσμα είναι μεγαλύτερο ή ίσο του αντίστοιχου αποτελέσματος (στον ίδιο κύκλο επανάληψης) της διεργασίας Δ4, τότε εκτυπώνει στη οθόνη το όνομά της.

Ερώτημα Β.1 [4]

Τεκμηριώστε τον απαιτούμενο συγχρονισμό και προσδιορίστε τη σειρά εκτέλεσης των διεργασιών. Δώστε το γράφο προτεραιότητας (για έναν κύκλο επανάληψης) για τις διεργασίες Δ1, Δ2, Δ3, Δ4, Δ5.

Ερώτημα Β.2 [6]

Δώστε τον κώδικα των 5 παράλληλων διεργασιών. Ο κώδικας θα πρέπει να υλοποιεί τις διεργασίες όπως αυτές περιγράφονται ανωτέρω και να συμπεριλαμβάνει τις απαραίτητες εντολές συγχρονισμού με **σημαφόρους**.

Σημειώσεις - Σχόλια:

1. Μια διεργασία μπορεί να βρει έναν τυχαίο αριθμό στο διάστημα 1..κ καλώντας τη συνάρτηση random(1..k).
2. Η εγγραφή μιας μεταβλητής/τιμής y στον απομονωτή buf πραγματοποιείται χρησιμοποιώντας την εντολή write(buf, y). Με την εγγραφή μιας νέας τιμής, η παλιά τιμή του απομονωτή χάνεται.
3. Η ανάγνωση του περιεχομένου του απομονωτή buf στη μεταβλητή y πραγματοποιείται με την εντολή read(buf, y). Η ανάγνωση δεν τροποποιεί τα περιεχόμενα και μπορεί να εκτελεστεί παράλληλα από δυο ή περισσότερες διεργασίες.
4. Η εκτύπωση στην οθόνη της μεταβλητής y, γίνεται με την εντολή write(y).
5. Η σύγκριση των τελικών τιμών των Δ4, Δ5 προτείνεται να γίνει μέσω της χρήσης/σύγκρισης διαμοιραζόμενων μεταβλητών.
6. Η λύση σας πρέπει να προσφέρει το μέγιστο παραλληλισμό, δηλ. πρέπει να αποφεύγει την τετριμμένη λύση της “σειριοποίησης” (δηλ. της αναγκαστικής εκτέλεσης της μιας μετά την άλλη) των διεργασιών, όταν αυτή μπορεί να αποφευχθεί.
7. Στο ερώτημα Β.1 δεν σας ζητείται μόνο ο γράφος προτεραιότητας αλλά θα πρέπει και να αναλύσετε/τεκμηριώσετε τον συγχρονισμό των διεργασιών. Ουσιαστικά σας ζητείται να δώσετε τις λεπτομερείς προδιαγραφές συγχρονισμού που θα υλοποιήσετε στο ερώτημα Β.2.

Ερώτημα Γ [8]

Θεωρείστε τον ακόλουθο κώδικα για δύο διεργασίες Δ0 και Δ1 που εκτελούνται παράλληλα. Στον κώδικα των διεργασιών χρησιμοποιούνται οι εξής κοινά διαμοιραζόμενες μεταβλητές:οι λογικές μεταβλητές flag0 και flag1, που η καθεμία αρχικοποιείται στην τιμή FALSE (ψευδής), και η ακέραια μεταβλητή turn, με αρχική τιμή 0.

```
shared boolean flag0 = flag1 = FALSE;
shared integer turn = 0;
```

Διεργασία Δ0

```
repeat
   flag0 = TRUE;
   while (turn != 0) {
       while (flag1 == TRUE) do noop;
       turn = 0;
   }
   <ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
   flag0 = FALSE;
forever
```

Διεργασία Δ1

```
repeat
   flag1 = TRUE
   while (turn != 1) {
       while (flag0 == TRUE) do noop;
       turn = 1;
   }
   <ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
   flag1 = FALSE;
forever
```

Ο παραπάνω κώδικας δεν εξασφαλίζει τον αμοιβαίο αποκλεισμό των διεργασιών Δ0 και Δ1, αφού υπάρχει περίπτωση οι δύο διεργασίες να εκτελούν ταυτόχρονα το τμήμα εντολών <ΚΡΙΣΙΜΟ ΤΜΗΜΑ>. Ζητείται να προσδιορίσετε ένα σενάριο εκτέλεσης των εντολών των δύο διεργασιών που οδηγεί σε αυτή την περίπτωση.

Για την περιγραφή του σεναρίου, καλείστε να συμπληρώσετε πιο συγκεκριμένα τον ακόλουθο ημιτελή πίνακα, παρουσιάζοντας την εκτέλεση των εντολών των δύο διεργασιών που οδηγεί σε παραβίαση του αμοιβαίου αποκλεισμού. *Σημείωση:* το σενάριο που θα παρουσιάσετε θεωρείστε ότι αρχίζει όπως υποδεικνύουν οι πρώτες (ήδη συμπληρωμένες για διευκόλυνσή σας) γραμμές του πίνακα. Ο αριθμός των κενών γραμμών που έχουν εισαχθεί από εκεί και κάτω έχει επιλεγεί τυχαία (συμπληρώστε δηλαδή με όσες γραμμές απαιτούνται - λιγότερες ή περισσότερες - για την ακριβή παρουσίαση της δικιάς σας προτεινόμενης συνέχειας του σεναρίου ώστε να προκύπτει παραβίαση του αμοιβαίου αποκλεισμού).

Διεργασία Δ0	Διεργασία Δ1	flag0	flag1	turn
		FALSE	FALSE	0
flag0 = TRUE		TRUE	FALSE	0
Εκτελεί το εξωτερικό while		TRUE	FALSE	0
Εισέρχεται στο ΚΡΙΣΙΜΟ ΤΜΗΜΑ		TRUE	FALSE	0
	flag1 = TRUE	TRUE	TRUE	0
	Εκτελεί το εξωτερικό while	TRUE	TRUE	0
	Εκτελεί το εσωτερικό while	TRUE	TRUE	0
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….
….	….	….	….	….

Σημείωση: Με noop στον κώδικα των διεργασιών συμβολίζουμε ότι δεν εκτελείται κάποια εντολή (no operation).

Ερώτημα Δ[12]

Στο χώρο ενός εργοταξίου ισχύει ένας συγκεκριμένος κανονισμός λειτουργίας που επιβάλλει να είναι παρών ένας επιβλέπων μηχανικός για κάθε τρεις εργάτες. Κάθε επιβλέπων μηχανικός που προσέρχεται στο εργοτάξιο προσομοιώνεται από τη διεργασία Supervisor, ενώ κάθε εργάτης που προσέρχεται στο εργοτάξιο προσομοιώνεται από τη διεργασία Worker.

Ο κώδικας των διεργασιών δίνεται ακολούθως και για να επιτευχθεί ο απαραίτητος συγχρονισμός (δηλ. για να ισχύει πάντοτε ο κανονισμός λειτουργίας που προαναφέρθηκε) χρησιμοποιούνται σημαφόροι (που αρχικοποιούνται σε συγκεκριμένες τιμές) και αντίστοιχες εντολές signal / wait επί αυτών των σημαφόρων.

```
semaphore S = W = 0;
binary semaphore mutex = 1;
```

Διεργασία Supervisor

```
<Προσέλευση Επιβλέποντα στο Εργοτάξιο>
signal(S);
signal(S);
signal(S);
<Επίβλεψη Εργατών>
wait(mutex);
wait(W);
wait(W);
wait(W);
signal(mutex);
<Αποχώρηση Επιβλέποντα από το Εργοτάξιο>
```

Διεργασία Worker

```
<Προσέλευση Εργάτη στο Εργοτάξιο>
wait(S);
<Εργασία στο Εργοτάξιο>
signal(W);
<Αποχώρηση Εργάτη από το Εργοτάξιο>
```

Αφού μελετήσετε τον κώδικα και κατανοήσετε το ρόλο των σημαφόρων και των εντολών signal/wait στην επίτευξη του απαραίτητου συγχρονισμού, καλείστε να εντοπίσετε/προσδιορίσετε ποιο πρόβλημα μπορεί να συμβεί εάν δεν χρησιμοποιηθεί ο δυαδικός σημαφόρος mutex και διαγραφούν οι αντίστοιχες εντολές (wait(mutex) και signal(mutex)) από τον κώδικα της διεργασίας Supervisor. Για περαιτέρω τεκμηρίωση, να δώσετε ένα σενάριο εκτέλεσης των εντολών των διεργασιών που να καταλήγει στο πρόβλημα που εντοπίσατε.

Ερώτημα Ε [20]

Δίνεται το παρακάτω σύνολο διεργασιών οι οποίες καταφθάνουν για να εκτελεστούν σε ένα υπολογιστικό σύστημα:

Διεργασία	Χρόνος Άφιξης	Χρόνος Εκτέλεσης	Προτεραιότητα
P1	0	12	3
P2	5	19	3
P3	8	21	5
P4	11	13	2
P5	15	5	3

Σχεδιάζοντας τα κατάλληλα διαγράμματα Gantt δείξτε πώς θα εκτελεστούν οι διεργασίες αυτές στην Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ), και υπολογίστε τους μέσους χρόνους διεκπεραίωσης (ΜΧΔ) και αναμονής (ΜΧΑ), για κάθε έναν από τους παρακάτω αλγόριθμους χρονοδρομολόγησης. Θεωρήστε ότι ο χρόνος εναλλαγής (context switch) είναι αμελητέος και ότι μεγάλες τιμές προτεραιότητας σηματοδοτούν μεγαλύτερες προτεραιότητες.

α) FCFS (First Come First Served) **[2.5]**

β) SJF (Shortest Job First) **[2.5]**

γ) SRTF (Shortest Remaining Time First) **[5]**

δ) Προεκχωρητικός Προτεραιότητας (Preemptive Priority Scheduling) [με χρήση αλγορίθμου FCFS (First Come First Served) στις περιπτώσεις που έχουμε ίσες προτεραιότητες] **[5]**

ε) RR (Round Robin) με κβάντο χρόνου 8 msec **[5]**

Σημείωση: Στην περίπτωση της χρονοδρομολόγησης RR, θεωρήστε πως αν τη χρονική στιγμή που μια νέα διεργασία έρχεται στο σύστημα διακόπτεται η εκτέλεση μιας διεργασίας (γιατί τελειώνει το κβάντο χρόνου της), τότε η νέα διεργασία εισέρχεται μετά από αυτή που διακόπτεται στην ουρά των έτοιμων διεργασιών.

Καλή Επιτυχία!!!

Αριθμός μελών ανά ομάδα=3
Ημερομηνία Παράδοσης: 13/01/2020

Αποστολή εργασιών με e-mail στους διδάσκοντες:

makri@ceid.upatras.gr, aristeid@ceid.upatras.gr, sioutas@ceid.upatras.gr