Λειτουργικά Συστήματα

Απαντήσεις 1ης εργασίας

Μάριος Στεφανίδης (1067458) & Μαύρα Πολυδώρου (1064885)

Μέρος Α

Ερώτημα Α: Παρακάτω περιγράφεται η λειτουργία των δοθέντων προγραμμάτων.

- Αρχικά, η πατρική διεργασία καλεί την fork(), δημιουργώντας μια διεργασία-παιδί και στην συνέχεια η πρώτη περιμένει για 2 δευτερόλεπτα (καλώντας την sleep(2)) και έπειτα τερματίζει. Η διεργασία παιδί συνεχίζει εμφανίζοντας το process ID της γονικής διεργασίας για 3 δευτερόλεπτα (getppid()). Παρατηρούμε ότι το ID της γονικής διεργασίας της διεργασίας-παιδί αλλάζει στο τελευταίο printf μετά τον τερματισμό της πατρικής διεργασίας. Αυτό συμβαίνει, γιατί αφού ολοκληρωθεί η πατρική διεργασία, ο έλεγχος επιστρέφει στο κέλυφος του λειτουργικού συστήματος, αφού το παιδί εκτελείται στο παρασκήνιο.

  Compile: My parent is 8850
  My parent is 8850
  My parent is 1611

- Καταρχάς, η πατρική διεργασία καλεί την fork(), δημιουργώντας μια διεργασία-παιδί, οι οποίες στην συνέχεια τυπώνουν παράλληλα αυξανόμενους αριθμούς. Παρατηρούμε το αποτέλεσμα του χρονοπρογραμματισμού στις εναλλαγές που γίνονται μεταξύ των printf (όταν pid>0 αναφέρομαι στον γονέα, διαφορετικά στο παιδί).

Ερώτημα Β: Ο απαιτούμενος κώδικας περιλαμβάνεται στο ίδιο archive με αυτό το pdf στον υποφάκελο Meros1_QuestionB. Παραθέτουμε επεξηγήσεις για το αρχείο που θα δείτε.

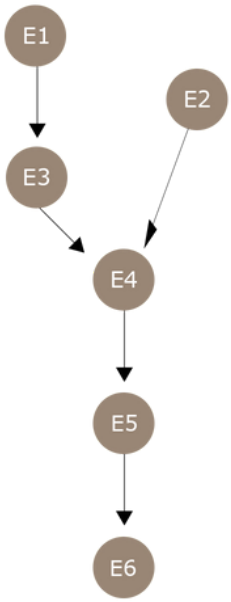
Το κομμάτι κώδικα περιλαμβάνει:

- Υλοποίηση του Peterson αλγορίθμου χρησιμοποιώντας pthreads και τις αντίστοιχες συναρτήσεις create και join.
- Έχουν δημιουργηθεί οι συναρτήσεις lock και unlock (σύμφωνα με το πρότυπο του αλγορίθμου), ώστε κάθε διεργασία να εισέρχεται και εξέρχεται από την κρίσιμη περιοχή.
- Η συνάρτηση child_process_i υλοποιεί το ζητούμενο του προβλήματος.
- Ακόμα, γίνεται χρήση της atomic (στην μεταβλητή X που είναι global και κοινή και για τις δύο διεργασίες), προκειμένου να εξασφαλιστεί race-free access.

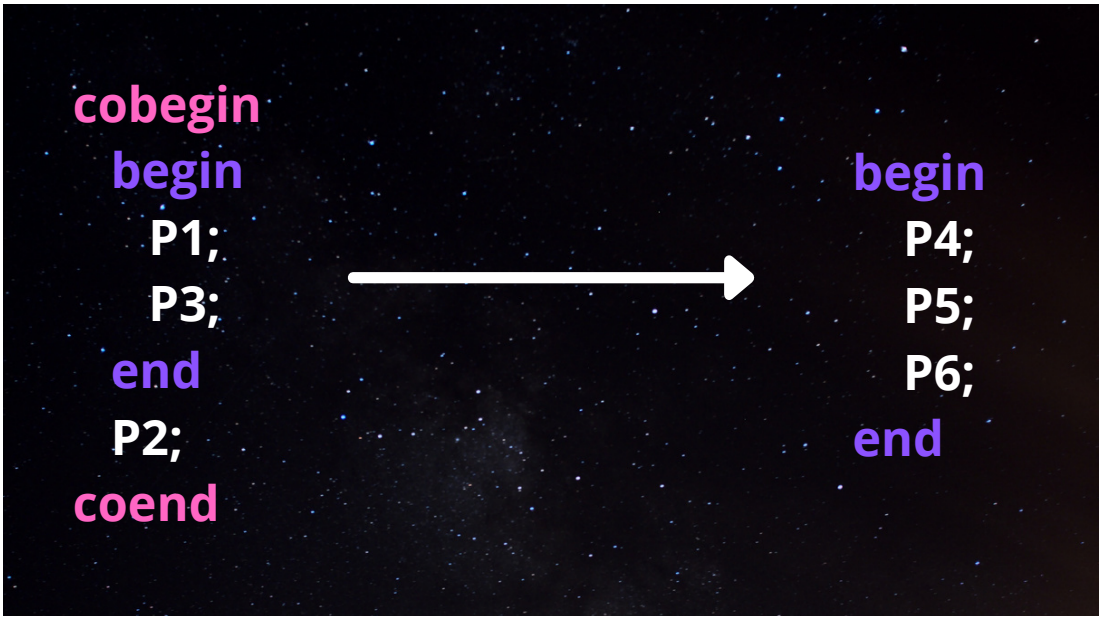
Μέρος Β

Ερώτημα Α

- Στον παρακάτω γράφο προτεραιοτήτων αναπαρίσταται ο δοθέν ακολουθιακός κώδικας με τη μέγιστη δυνατή παραλληλία.



- Παρακάτω παρατίθεται ένα πρόγραμμα που είναι ισοδύναμο με το δοθέν ακολουθιακό πρόγραμμα, με τη μέγιστη δυνατή παραλληλία και με τη χρήση εντολών cobegin ... coend και εντολών begin ... end.



- Παρακάτω παρατίθεται ένα παράλληλο πρόγραμμα χρησιμοποιώντας την εντολή cobegin ... coend και δυαδικούς σημαφόρους, το οποίο βασίζεται στον παραπάνω γράφο προτεραιοτήτων.

```
var S1, S23, S4, S5: semaphores;
S1=0; S23=-1; S4=0; S5=0;

cobegin
  begin E1; up(S1); end
  begin down(S1); E2; up(S23); end
  begin E3; up(S23); end
  begin down(S23); E4; up(S4); end
  begin down(S4); E5; up(S5); end
  begin down(S5); E6; end
coend
```

Ερώτημα Β

Η σειρά που πρέπει να ακολουθήσουν τα τμήματα εντολών προκειμένου να εξασφαλιστούν όλες οι απαιτήσεις συγχρονισμού είναι: **E1.1 -> E3.1 -> E2.1 -> E1.2, -> E2.2 -> E3.2**.

```
var s11, s31, s21, s12,
s22, s32 : semaphores;
s11:=1; s31:=0; s21:=0;
s12:=0; s22:=0; s32:=0;

cobegin

    //Process 1
    while (TRUE) {
        wait(s11);
        //Τμήμα εντολών E1.1
        signal(s31);
        wait(s12);
        //Τμήμα εντολών E1.2
        //Κρίσιμο Τμήμα
        signal(s22);
        }

        //Process 2
    while (TRUE) {
        wait(s21);
```

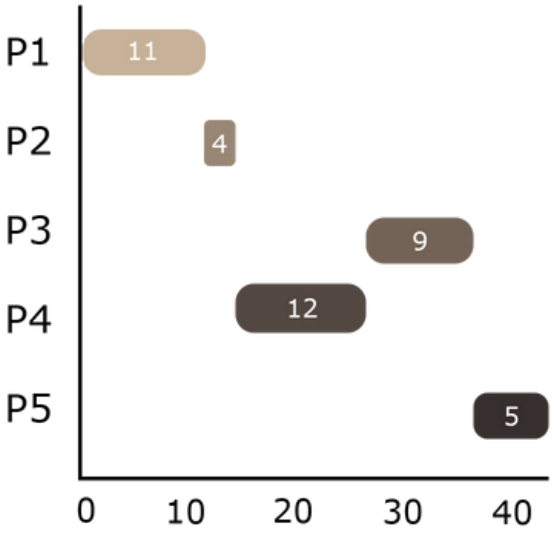
```
    while (TRUE) {
        wait(s21);
        //Τμήμα εντολών E2.1
        signal(s12);
        wait(s22);
        //Τμήμα εντολών E2.2
        //Κρίσιμο Τμήμα
        singal(3.2);
    }

        //Process 3
    while (TRUE) {
        wait(s31);
        //Τμήμα εντολών E3.1
        signal(s21);
        wait(s32);
        //Τμήμα εντολών E3.2
        //Κρίσιμο τμήμα
    }

coend;
```

Ερώτημα Γ: Παρακάτω παρατίθεται ο ζητούμενος πίνακας:

Producer	Consumer	empty	full
		2	0
	wait(full)	2	-1
wait(empty)		1	-1
signal(full)		1	0
wait(empty)		0	0
	signal(empty)	1	0
signal(full)		1	1
wait(empty)		0	1
signal(full)		0	2
wait(empty)		-1	2
	wait(full)	-1	1
	signal(empty)	0	1

<u>Ερώτημα Δ:</u> Παρακάτω παρατίθεται οι απαραίτητοι υπολογισμοί και τα Gantt. Εσωτερικά των χρωματιστών πλαισίων αναγράφεται ο χρόνος που απασχόλησε η κάθε διεργασία την ΚΜΕ.

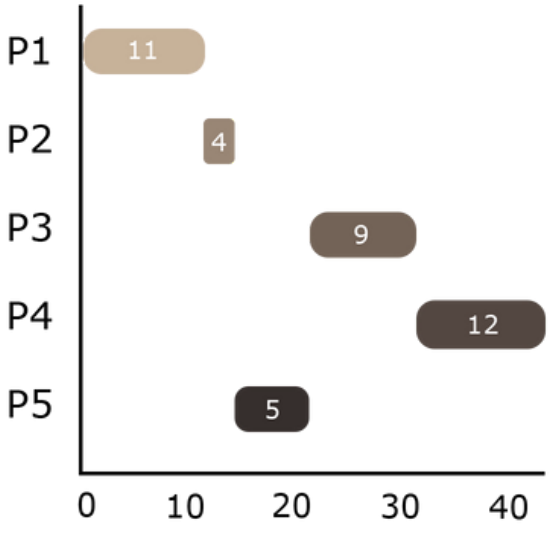
First Come - First Served



P	t1	t2	d	ΧΔ	ΧΑ
P1	0	11	11	11	0
P2	2	15	4	13	9
P3	3	36	9	33	24
P4	3	27	12	24	12
P5	5	41	5	36	31
ΜΧΑ					15.2
ΜΧΔ				23.4	

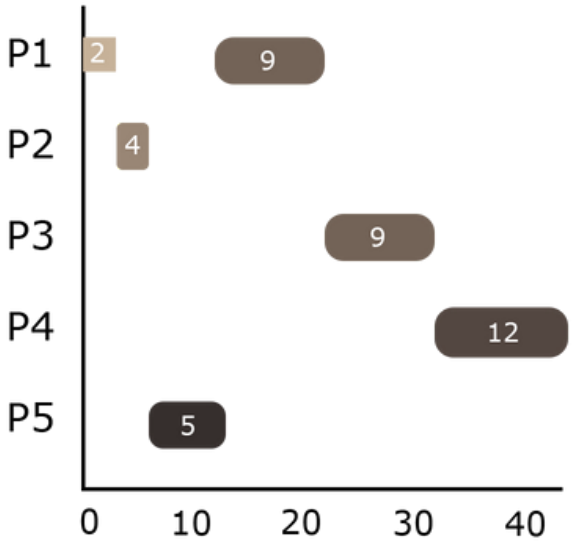
*Οι διεργασίες P3 και P4 φθάνουν την ίδια χρονική στιγμή, οπότε επιλέχθηκε τυχαία η διεργασία P4 να εκτελεστεί πρώτη.

Shortest Job First



P	t1	t2	d	ΧΔ	ΧΑ
P1	0	11	11	11	0
P2	2	15	4	13	9
P3	3	29	9	26	17
P4	3	41	12	38	26
P5	5	20	5	15	10
ΜΧΑ					12.4
ΜΧΔ				20.6	

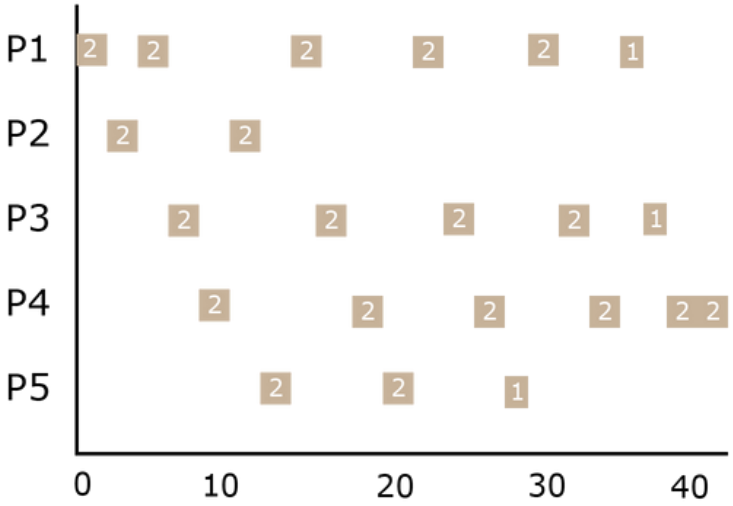
Shortest remaining time first



P	t1	t2	d	ΧΔ	ΧΑ
P1	0	20	11	20	9
P2	2	6	4	4	0
P3	3	29	9	26	17
P4	3	41	12	38	26
P5	5	11	5	6	1
ΜΧΑ					10.6
ΜΧΔ				20.6	

*Στο σημείο που οι διεργασίες P1 & P3 χρειάζοντουσαν τον ίδιο χρόνο στη CPU (->9 ms), επιλέχθηκε η διεργασία που ήδη είχε εξυπηρετηθεί για 2ms (δηλαδή η P1).

Round Robin



P	t1	t2	d	ΧΔ	ΧΑ
P1	0	36	11	36	25
P2	2	12	4	10	6
P3	3	37	9	34	25
P4	3	41	12	38	26
P5	5	29	5	24	19
ΜΧΑ					20.2
ΜΧΔ				20.6	