Αναφορά 1ης Εργαστηριακής Άσκησης Στα Λειτουργικά Συστήματα

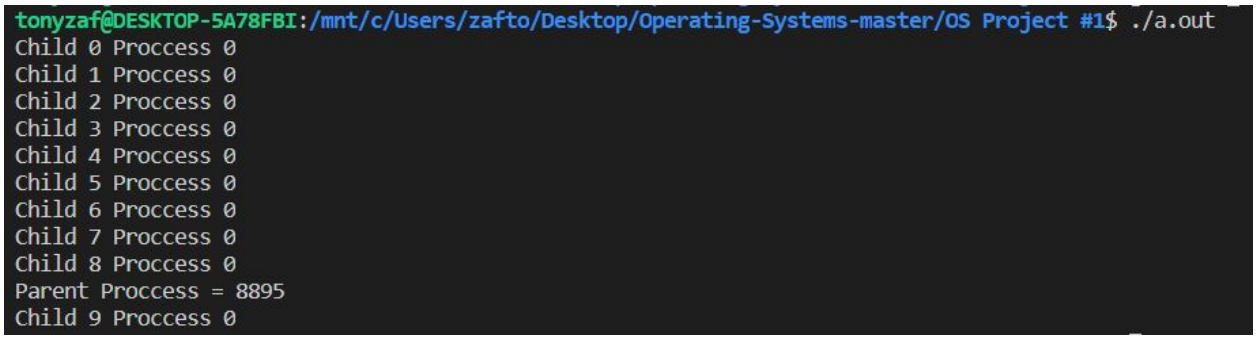
Ζαφειρέλης Αντώνιος Ζαφείριος ΑΜ:1059605

Μέρος 1ο:

Ερώτημα 1:

Εκτελώντας το πρόγραμμα παρατηρούμε ότι εκτυπώνονται 400 συνολικά μηνύματα. Γνωρίζουμε ότι η κλήση του system call fork() φτιάχνει μια ανεξάρτητη διεργασία-κλώνο (παιδί) της τρέχουσας διεργασίας (γονέας). Από τη δημιουργία της διεργασίας-παιδί και μετά η κάθε διεργασία εκτελεί το ίδιο κώδικα. Έτσι η διεργασία-γονέας (pid!=0) εκτελώντας τη δική της for θα εκτυπώσει 200 σχετικά μηνύματα και η διεργασία-παιδί εκτελώντας τη δική της for θα εκτυπώσει άλλα 200 σχετικά μηνύματα. Σε κάθε διεργασία, η σειρά των μηνυμάτων είναι αυτή που η μεταβλητή i καθορίζει αλλά είναι τυχαίος ο τρόπος που τα μηνύματα των 2 διεργασιών εμφανίζονται στην οθόνη.

Ερώτημα 2:



Το pid = 0 μας δείχνει ότι πρόκειται για Child Proccess

Ερώτημα 3:

```
tonyzaf@DESKTOP-5A78FBI:/mnt/c/Users/zafto/Desktop/Operating-Systems-master/OS Project #1$ ./a.out
PID           Parent PID    Child PID
1418          0             1419
1419          1418          1420
1420          1419          1421
1421          1420          1422
1422          1421          1423
1423          1422          1424
1424          1423          1425
1425          1424          1426
1426          1425          1427
1427          1426          1428
```

Μέρος 2ο:

Ερώτημα Α:

1: **Var:**s5=1, s1=0, s2=0, s3=0, s4=0: **Semaphores**

Process 1	***Process 2***	***Process 3***

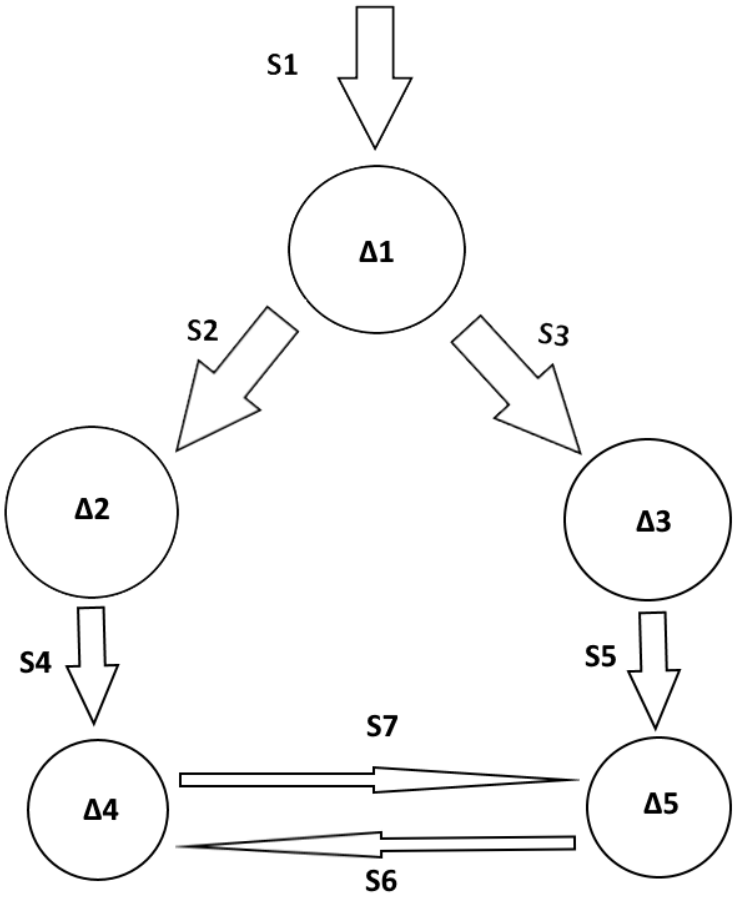
for k=1 to 10 do begin
wait(s5);
E1.1;
signal(s1);
E1.2;
signal(s3)
end
	for j=1 to 10 do begin
	wait(s1);
	E2.1;
	signal(s2);
	wait(s3);
	E2.2;
	signal (s4);
	end
		for l=1 to 10 do begin
		wait(s2);
		E.3.1;
		wait(s4);
		E.3.2;
		signal (s5);
		end

2. **Var:** Sem1=1, Sem2=0, Sem3=0: **Semaphores;**

Process 1	***Process 2***	***Process 3***

for k=1 to 10 do begin
wait(Sem1);
E1.1;
signal(Sem2);
E1.2;
signal(Sem2);
end
	for j=1 to 10 do begin
	wait(Sem2);
	E2.1;
	signal(Sem3);
	wait(Sem2);
	E2.2;
	signal(Sem3);
	end
		for l=1 to 10 do begin
		wait(Sem3);
		E.3.1;
		wait(Sem3);
		E.3.2;
		signal(Sem1);
		end

Ερώτημα Β.1:



Ερώτημα Β.2:

```
var s1=1,s2=0,s3=0,s4=0,s5=0,s6=0,s7=0: semaphores;

shared var d4=0,d5=0: integer;

cobegin

        repeat

                Δ1: begin down(s1); y=random(1..10); write(buf1,y); up(s2); up(s3); end;

                Δ2: begin down(s2); read(buf1,y); write(buf2,y); up(s4); end;

                Δ3: begin down(s3); read(buf1,z); write(buf3,z);up(s5);end;

                Δ4: begin down(s4); read(buf2,x); k=random(1..10); d4=x+k;
                        up(s7); down(s6);

                                if d4>d5 then begin write("d4"); up(s1); end;

                  end

                    Δ5: begin down(s5); read(buf3,w); a=random(1…10); d5=w+a;
                        down(s7);up(s6);
                        if d4≤ d5 then begin write("d5"); up(s1); end;
                  end

        forever

coend
```

Ερώτημα Γ:

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	*flag1 = TRUE*	*TRUE*	*TRUE*	*0*
	Εκτελεί το εξωτερικό while	*TRUE*	*TRUE*	*0*
	Εκτελεί το εσωτερικό while	*TRUE*	*TRUE*	*0*
	Σταματά στο While (flag0==TRUE) do noop;	TRUE	TRUE	0
Ολοκληρώνει το ΚΡΙΣΙΜΟ ΤΜΗΜΑ		TRUE	TRUE	0
flag0=FALSE		FALSE	TRUE	0
Μπαίνει στην εξωτερική repeat		FALSE	TRUE	0
While (turn!=0) {		FALSE	TRUE	0
	Turn=1	FALSE	TRUE	1
Εισέρχεται στο ΚΡΙΣΙΜΟ ΤΜΗΜΑ		FALSE	TRUE	1
	While (turn!=1) {	FALSE	TRUE	1

	ξρχεται στο ΚΡΙΣΙΜΟ ΤΜΗΜΑ	FALSE	TRUE	1

Ερώτημα Δ:

Semaphores S = W =0;

binary semaphore mutex = 1;

Διεργασία Supervisor	**Διεργασία Worker**
<Προσέλευση Επιβλέποντα στο Εργοτάξιο>	<Προσέλευση Εργάτη στο Εργοτάξιο>
1: signal(S);	wait(S);
2:signal(S);	<Εργασία στο Εργοτάξιο>
3:signal(S);	signal(W);
<Επίβλεψη Εργατών>	<Αποχώρηση Εργάτη από το Εργοτάξιο>
wait(mutex);	
4:wait(W);	
5:wait(W);	
6:wait(W);	
signal(mutex);	
<Αποχώρηση Επιβλέποντα από το Εργοτάξιο>	

Έστω ότι έχουν έρθει στο εργοτάξιο αρχικά 3 supervisors (s1,s2,s3):

Έστω ότι στη συνέχεια προσέρχονται για δουλειά 9 workers (w1,w2,…,w9)

Τότε η wait(S) χωρίς αναμονή, κάνει S=S-1.

Έτσι η S θα μηδενιστεί.

Έστω ότι αποχωρούν 3 workers (π.χ. w1, w2, w3) αφού ολοκλήρωσαν την εργασία τους και έστω ότι η εκτέλεση των διεργασιών Worker τους (signal(W)) προηγείται αυτής της εκτέλεσης οποιασδήποτε wait(W) των supervisor διεργασιών.

Τότε η W θα έχει την τιμή 3.

Έστω ότι αμέσως μετά εκτελούνται με τη σειρά:

Supervisor1: 4:wait(W)

Supervisor2: 4:wait(W)
Supervisor3: 4:wait(W)

Τώρα η W έχει τιμή 0 (από τις 3 wait που εκτελέστηκαν)

Έτσι όλες οι μετέπειτα 5:wait(W), 6:wait(W) όλων supervisor διεργασιών δεν προχωρούν αφού όλες εκτελούν το while loop τους.

<u>Έτσι τώρα ενώ ένας supervisor θα μπορούσε (σύμφωνα με τον κανονισμό) να αποχωρήσει, η εκτέλεση της διεργασίας του δεν του το επιτρέπει!!!</u>

Έστω τώρα ότι αποχωρούν 3 άλλοι workers (π.χ. w4, w5, w6) αφού ολοκλήρωσαν την εργασία τους και έστω ότι η εκτέλεση των διεργασιών Worker τους (signal(W)) προηγείται αυτής της εκτέλεσης οποιασδήποτε wait(W) των supervisor διεργασιών.

Τότε η W θα έχει πάλι την τιμή 3.

Έστω ότι αμέσως μετά εκτελούνται με τη σειρά:

Supervisor1: 5:wait(W)

Supervisor2: 5:wait(W)
Supervisor3: 5:wait(W)

Έτσι όλες οι μετέπειτα 6:wait(W) όλων supervisor διεργασιών δεν προχωρούν αφού όλες εκτελούν το while loop τους.

Έτσι τώρα ενώ δύο supervisors θα μπορούσαν (σύμφωνα με τον κανονισμό) να αποχωρήσουν, η εκτέλεση της διεργασίας τους δεν του το επιτρέπει!!!

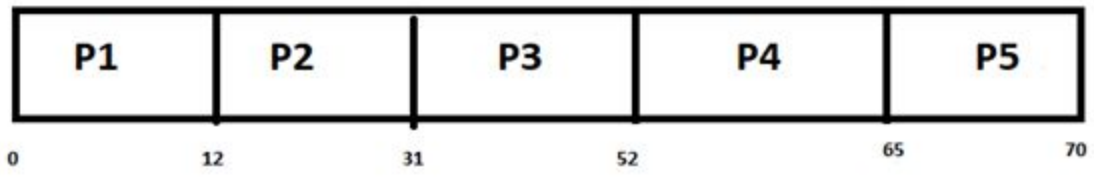
Παρατηρούμε ότι για να μπορέσει κάποιος από τους 3 επιβλέποντες να αποχωρήσει μόλις αποχωρήσουν οι πρώτοι τρεις εργάτες (σύμφωνα με τον κανονισμό), πρέπει το συγκεκριμένο τμήμα

4:wait(W);
5:wait(W);
6:wait(W);

να οριστεί σαν critical section με τη χρήση του δυαδικού σημαφόρου mutex και η πρόσβαση σε αυτό να επιτραπεί μόνο σε μία supervisor διεργασία κάθε φορά.

Ερώτημα Ε:

FCFS Διάγραμμα Gantt



Χρόνοι Διεκπεραίωσης

- ΧΠ1=12-0=12
- ΧΠ2=31-5=26
- ΧΠ3=52-8=44
- ΧΠ4=65-11=54

· ΧΠ5=70-15=55

Ο Μέσος Χρόνος Διεκπεραίωσης είναι: (12+26+44+54+55)/5 =38,2

· ΧΑ1=0-0=0

· ΧΑ2=12-5=7

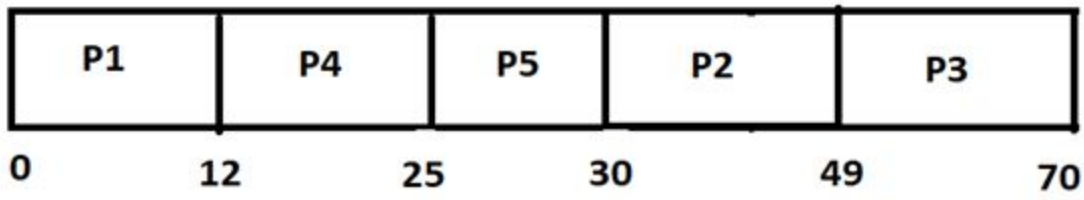
· ΧΑ3=31-8=23

· ΧΑ4=52-11=41

· ΧΑ5=65-15=50

Ο Μέσος Χρόνος Αναμονής είναι: (0+7+23+41+50)/5 = 24,2

SJF Διάγραμμα Gantt



Χρόνοι Διεκπεραίωσης

· ΧΠ1=12-0=12

· ΧΠ4=25-11=14

· ΧΠ5=30-15=15

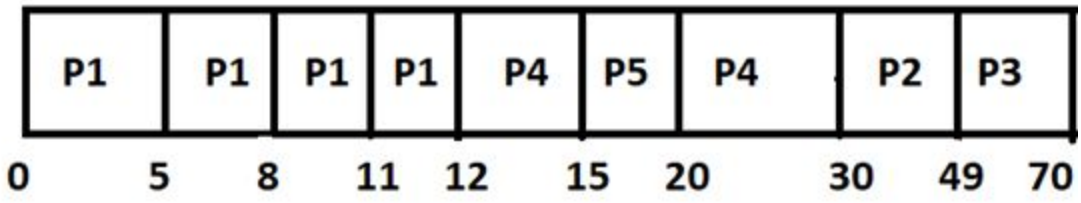
- ΧΠ2=49-5=44
- ΧΠ3=70-8=62

Ο Μέσος Χρόνος Διεκπεραίωσης είναι: (12+14+15+44+62)/5 =29,4

- ΧΑ1=0-0=0
- ΧΑ2=12-11=1
- ΧΑ3=25-15=10
- ΧΑ4=30-5=25
- ΧΑ5=49-8=41

Ο Μέσος Χρόνος Αναμονής είναι: (0+1+10+25+41)/5 = 15,4

SRTF Διάγραμμα Gantt



Χρόνοι Διεκπεραίωσης

- ΧΠ1=12-0=12
- ΧΠ2=49-5=44
- ΧΠ3=70-8=62
- ΧΠ4=30-11=19

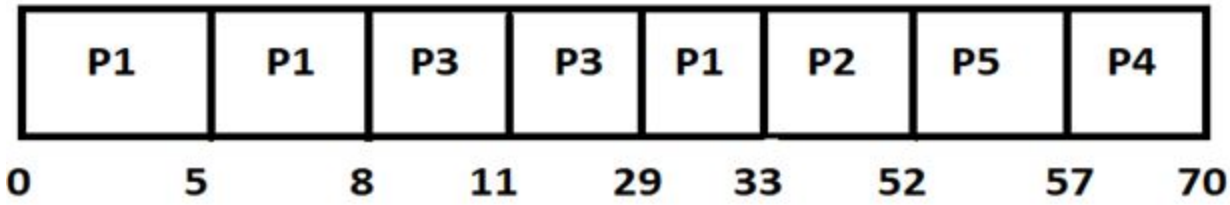
- ΧΠ5=20-15=5

Ο Μέσος Χρόνος Διεκπεραίωσης είναι: (12+44+62+19+5)/5 =28,4

- ΧΑ1=0-0=0
- ΧΑ2=30-5=25
- ΧΑ3=49-8=41
- ΧΑ4=(12-11)+(20-15)=1+5=6
- ΧΑ5=15-15=0

Ο Μέσος Χρόνος Αναμονής είναι: (0+25+41+6+0)/5 = 14,4

Preemptive Priority Διάγραμμα Gantt



Χρόνοι Διεκπεραίωσης

- ΧΠ1=33-0=33
- ΧΠ2=52-5=47
- ΧΠ3=29-8=21
- ΧΠ4=70-11=59

· ΧΠ5=57-15=42

Ο Μέσος Χρόνος Διεκπεραίωσης είναι: (33+47+21+59+42)/5 =40,4

· ΧΑ1=(0-0)+(29-8)=21

· ΧΑ2=33-5=28

· ΧΑ3=8-8=0

· ΧΑ4=57-11=46

· ΧΑ5=52-15=37

Ο Μέσος Χρόνος Αναμονής είναι: (21+28+0+46+37)/5 = 26,4

<u>Round Robbin Διάγραμμα Gantt</u>