Λειτουργικά Συστήματα

1η Εργαστηριακή Άσκηση

Μηνάς Δημοσθένης Α.Μ. 1059602 email st1059602@ceid.upatras.gr

Γεωργιάδης Χρήστος ΑΜ 1059579 email st1059579@ceid.upatras.gr

Μακρής Γιώργος ΑΜ 1059679 email st1059679@ceid.upatras.gr

Μέρος Α Project 1

Ερώτημα Α

pid = fork(); //δημιουργείται ένα αντίγραφο της εκτελούμενης διεργασίας. Το αντίγραφο είναι η διεργασία 'παιδί' που εκτελείται ταυτόχρονα με τη διεργασία 'πατέρα' (δηλ. την αρχική διεργασία)

```
for (i=1; i<=200; i++) //η επανάληψη αυτή εκτελείται και από την 'πατέρα' 200 φορές αλλά και από το 'παιδί' 200 φορές
    if (pid > 0) //Το if αληθεύει μόνο στον 'πατέρα' και εκτελείται 200 φορές από αυτόν
      printf("%3i (parent)\n", i); //τυπώνονται 200 μηνύματα μόνο από τον 'πατέρα' που παραθέτουμε στη συνέχεια
    else ////Το else αληθεύει μόνο στο 'παιδί και εκτελείται 200 φορές από αυτό
      printf("%3i (child)\n", i); //τυπώνονται 200 μηνύματα μόνο από το 'παιδί που παραθέτουμε επίσης στη συνέχεια

return (0);
}
```

Μηνύματα Εκτέλεσης

```
Από διεργασία 'πατέρα'
1 (Parent)
2 (Parent)
3 (Parent)
……..
200 (Parent)

Από διεργασία 'παιδί'
1 (child)
2 (child)
3 (child)
……..
200 (child)
```

Ερώτημα Β

```
	for (i=1; i<=10;    i++)
	{
		prid=fork(); //create a child process that runs concurrently with parent process

		   if(prid==0) //this condition is true only for child process
		{
			printf("%d Child with id %d from Parent %d\n",i,getpid(),getppid());//Child process prints its ID and its
parent id
			exit(0); //child process terminates in order to create all processes from parent
		   }

	  }

	  for(i=1; i<=10; i++)//Parent process waits first for the termination of its children and after that terminates itself
		wait(0);

	return 0;
}
```

Ερώτημα Γ

```
	int i,pid=fork(); //here we create first child process

	if(pid!=0)//this is true only for the initial parent process
	  {
		printf("Process id %d,\t\tIts parent id %d,\tIts Child id %d\n", getpid(),getppid(),pid);//we print process id, its par-
ent id and its child id in order to ensure that each child generates a new one

			wait(0);//each parent process waits the termination of its child before it terminates itself
	  }

	for(i=2; i<=10; i++) //this for i sbeing executed 9 times in order to create 9 children
	  {
		if(pid==0) //this if is true in each children
		{
			pid=fork(); //each children generates a new one

			  if(pid!=0)//this is true for each nes parent process
			{
				printf("Process id %d,\t\tIts parent id %d,\tIts Child id %d\n", getpid(),getppid(),pid);//we print
process id, its parent id and its child id in order to ensure that each child generates a new one

					wait(0); //each parent process waits the termination of its child before it terminates
itself
			}
		   }

	}

	return 0;
}
```

Ερώτημα Δ

```
#define P 10000

void foo();

int main()
{
	time_t start, end;

	  int i=0, start_value=0,pid;

	  pid_t idp;

	start=time(0);//we store current time before the creation of processes in variable start

	printf("Arxiki timi deuteroleptwn %ld \n",start);//we use %ld format in order to print and reperesent time

	while(i<P) //while loop for P processes
	{
		pid=fork(); //we create a child process

		if(pid==0) //the if condition is true only for the child process
		   {
				foo(); //each child process executes function foo()

			exit(0); //after that each chid terminates
		   }

		i++;
	}



	for(i=1;i<=P; i++) //parent process wait for its children to terminate
		waitpid(idp,&start_value,WNOWAIT);


	end= time(0);//we store current time after the creation of processes in end variable

	printf("Teliki timi deuteroleptwn = %ld\n", end);

	printf("Diarkeia ektelesis =%ld sec \n",end-start);//we compute and print the total execution time of the P processes

	float mo=(float)(end-start)/100; //we compute the average execution time of the P processes

	printf("Mesi diarkeia =%f\n",mo); //we print the average execution time of the P processes

	return 0;
}


void foo()
{
	int x=0;

	x=x+10;
}
```

Μέρος Β Project 1

Άσκηση Α

(i)

Δηλώνουμε αρχικά τους πέντε σημαφόρους ως εξής:

var

a, b, c, d, e:semaphore;

Στη συνέχεια αρχικοποιούμε τους πέντε σημαιοφόρους μας με τις εξής τιμές:

begin

a=1,b=c=d=e=0;

cobegin

Process 1	**Process 2**	**Process 3**
for k=1 to 10 do	for k=1 to 10 do	for k=1 to 10 do
begin	**begin**	**begin**
P(a);	P(b);	P(c);
E1.1;	E2.1;	E3.1;
V(b);	V(c);	V(d);
P(d);	P(e);	P(c);
E1.2;	E2.2;	E3.2;
V(e);	V(c);	V(a);
end	**end**	**end**

coend

(ii)Τώρα πάμε να κάνουμε την ίδια διαδικασία με τον ελάχιστο αριθμό σημαφόρων. Χρησιμοποιούμε μόνο 3 αντί για 5 σημαφόρους γιατί μπορούμε να πετύχουμε τον ίδιο συγχρονισμό χρησιμοποιώντας μόνο 3 σημαφόρους όπως δείχνουμε στη συνέχεια.

var

a, b, c:semaphore;

Στη συνέχεια αρχικοποιούμε τους πέντε σημαιοφόρους με τις ακόλουθες τιμές:

begin

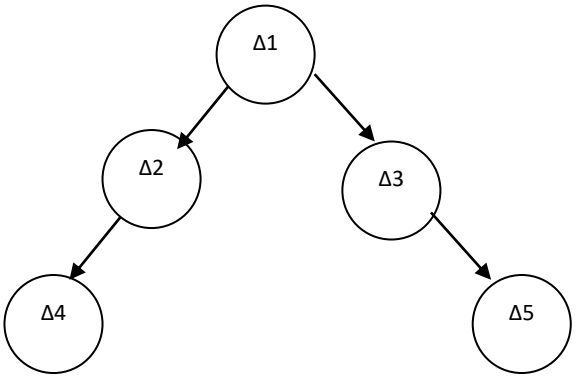
a=1, b=c=0.

Process 1	**Process 2**	**Process 3**
for k=1 to 10 do	for k=1 to 10 do	for k=1 to 10 do
begin	**begin**	**begin**
P(a);	P(b);	P(c);
E1.1;	E2.1;	E3.1;
V(b);	V(c);	V(a);
P(a);	P(b);	P(c);
E1.2;	E2.2;	E3.2;
V(b);	V(c);	V(a);
end	**end**	**end**

coend

Άσκηση Β
1)

Γράφος Προτεραιότητας για ένα κύκλο επανάληψης



Ο συγκεκριμένος γράφος αρχίζει από την Δ1 καθώς από αυτήν, σύμφωνα με την εκφώνηση, ξεκινάνε όλα από αυτή. Μετά η Δ2 και Δ3 διαβάζουν ταυτόχρονα από τον buffer1 που έχει γράψει τον τυχαίο αριθμό η Δ1, για αυτό και βάζουμε την Δ2 και Δ3 ταυτόχρονα (ίδιος κύκλος επανάληψης) να αρχίζουν μετά τη Δ1 επειδή ακριβώς διαβάζουν και γράφουν στους buffer2 και buffer 3 αντίστοιχα τον αριθμό που επέλεξε η Δ1. Μετά βάζουμε την Δ4 και Δ5 να ξεκινούν μετά τις Δ2 και Δ3, διότι, σύμφωνα με την εκφώνηση, παίρνουν τους αριθμούς που έχουν αποθηκεύσει οι Δ2 και Δ3 στους buffer2 και buffer3 αντίστοιχα και εντοπίζουν μια τυχαία τιμή από το διάστημα 1..10 και την προσθέτουν σε αυτόν τον αριθμό. Κατόπιν η Δ4 αρχίζει μετά τη Δ2 γιατί η Δ4 διαβάζει από τον buffer2 την τιμή που έχει γράψει η Δ2 και ομοίως η Δ5 αρχίζει μετά τη Δ3 γιατί η Δ5 διαβάζει από τον buffer3 την τιμή που έχει γράψει η Δ3. Τέλος, η Δ4 και Δ5 επιστρέφουν στην Δ1, διότι οι 5 διεργασίες αυτές εκτελούνται συνεχώς.

Ο κώδικας του γράφου είναι ο ακόλουθος:

```
repeat
    Δ1;
  cobegin
      begin
          Δ2;
          Δ4;
      end;

      begin
          Δ3;
          Δ5;
      end;
  coend;
forever
```

2) Ένα λεπτό σημείο του συγκεκριμένου ερωτήματος είναι ότι για να ξαναρχίσει η Δ1 πρέπει η Δ4, Δ5 να έχουν ολοκληρωθεί, καθώς επίσης θα πρέπει να έχουμε εξασφαλίσει ότι η Δ4, Δ5 θα έχουν υπολογίσει πρώτα το αποτέλεσμα τους πριν τα συγκρίνουν οι δύο διεργασίες γιατί η Δ4 συγκρίνει το δικό της αποτέλεσμα με αυτό της Δ5 και η Δ5 με αυτό της Δ4.

Δηλώνουμε 5 σημαφόρους

```
var
      s12,s13,24,s35,s41,s51: semaphore;
```

Αρχικοποιούμε τους σημαιοφόρους ως εξής:

```
begin
    s12=s13=s24=s35=s41=s51=0;
```

cobegin
Δ1: begin repeat down(s41); down(s51); X=random(1..10); write (buf1,x); up(s12); up(s13); forever end;
Δ2: begin repeat down(s12); read(buf1,z); write(buf3,z); up(s35); forever end;
Δ3: begin repeat down(s13); read (buf1,z); write (buf3,z); up(s35); forever end
Δ4: begin repeat down(s24); read (buf2,a); b=random(1..10); c=a+b; up(sem2); down(sem1); if c>f write('Διεργασία 4'); up(s41); forever end;

Δ5: begin repeat down(s35); read(buf3,d); e=random(1..10); f=d+e; up(sem1); down(sem2); if f≥c write (‘Διεργασία 5’); up(s51); forever end; coend;

Ερώτημα Γ

Τα βήματα που παραθέτουμε είναι μετά ακριβώς από αυτά που παρατίθενται στην εκφώνηση και συγκεκριμένα μετά τη Δ1 που εκτελεί το εσωτερικό while

Διεργασία Δ0	**Διεργασία Δ1**	**Flag0**	**Flag1**	**turn**
Εξέρχεται απο το Κ.Τ.		TRUE	TRUE	0
Flag0= FALSE		FALLSE	TRUE	0
	Εξέρχεται απο το εσωτερικό while	FALSE	TRUE	0
Flag0= TRUE		TRUE	TRUE	0
Εκτελεί το εξωτερικό while		TRUE	TRUE	0
Εισέρχεται στο Κ.Τ.		TRUE	TRUE	0
	turn=1	TRUE	TRUE	1
	Εκτελεί το εξωτερικό while	TRUE	TRUE	
	Εισέρχεται στο Κ.Τ.	TRUE	TRUE	1

Παρατηρούμε ότι με βάση αυτό το σενάριο και οι 2 διεργασίες εισέρχονται ταυτόχρονα στο Κ.Τ. άρα δεν επιτυγχάνεται αμοιβαίος αποκλεισμός

Ερώτημα Δ

Αν παραληφθεί ο διάδικος σημαφόρος mutex τότε υπάρχει περίπτωση το σύστημα να πέσει σε αδιέξοδο. Ένα τέτοιο σενάριο παρουσιαζεται στη συνέχεια όταν στο σύστημα φτάσουν τρεις worker και δύο supervisor.

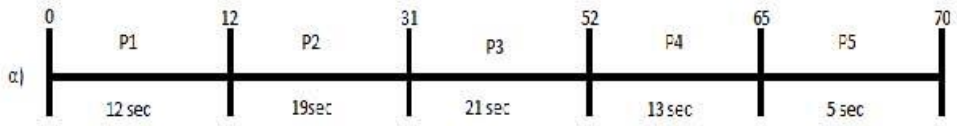
Worker 1	**Worker 2**	**Worker 3**	**Supervisor 1**	**Supervisor 2**
		wait(S) και μπλοκάρει διότι ο S=0		
	wait(S) και μπλοκάρει διότι ο S=0			
wait(S) και μπλοκάρει διότι ο S=0				
			signal(S)	
		Αφυπνίζεται και αρχίζει εργασία		
			signal(s)	
	Αφυπνίζεται και αρχίζει εργασία			
			signal(s)	
Αφυπνίζεται και αρχίζει εργασία				
			wait(W) και μπλοκάρει διότι ο W=0	
				signal(S)
				signal(S)
				signal(S)

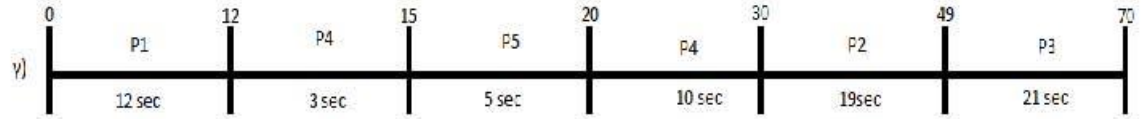
		signal(W)		
			αφυπνίζεται και συνεχίζει	
			wait(W) μπλοκάρει διότι ο W=0	
	signal(w)			
			αφυπνίζεται και συνεχίζει	
			wait(W) μπλοκάρει διότι ο W=0	
				wait(W) μπλοκάρει διότι ο W=0
signal(W)				
				αφυπνίζεται και συνεχί-ζει
				wait(W) μπλοκάρει διότι ο W=0

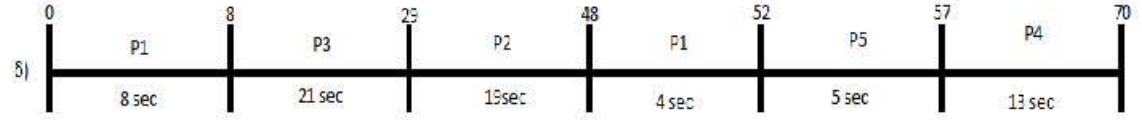
Το συμπέρασμα που προκύπτει είναι και μπλοκάρουν ταυτόχρονα και οι 2 supervisor όπως δείξαμε με το σενάριο αυτό.

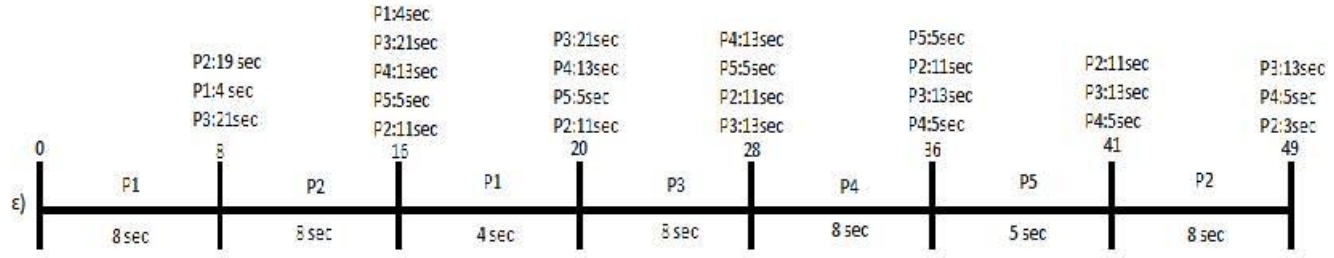
Άσκηση 5

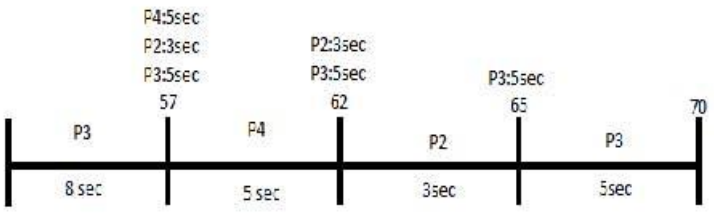
Τα αντίστοιχα διαγράμματα Gantt για τους αλγορίθμους χρονοπρογραμματισμού είναι τα ακόλουθα:











Οι μέσοι χρόνοι ολοκλήρωσης και αναμονής υπολογίζονται από τους ακόλουθους τύπους:

α)

$$\text{ΜΧΔ} = \frac{(12-0)+(31-5)+(52-8)+(65-11)+(70-15)}{5} = 38,2 m\sec$$

$$\text{Μ.Χ.Α} = \frac{(0-0)+(12-5)+(31-8)+(52-11)+(65-15)}{5} = 24,2 m\sec$$

β)

$$\text{ΜΧΔ} = \frac{(12-0)+(49-5)+(70-8)+(25-11)+(30-15)}{5} = 29,4 m\sec$$

$$\text{Μ.Χ.Α} = \frac{(0-0)+(30-5)+(49-8)+(12-11)+(25-15)}{5} = 15,4 m\sec$$

γ)

$$\text{ΜΧΔ} = \frac{(12-0)+(49-5)+(70-8)+(30-11)+(20-15)}{5} = 28,4 m\sec$$

$$\text{Μ.Χ.Α} = \frac{(0-0)+(30-5)+(49-8)+\big((12-11)+(20-15)\big)+(15-15)}{5} = 14,4 m\sec$$

δ)

$$\text{ΜΧΔ} = \frac{(52-0)+(48-5)+(29-8)+(70-11)+(57-15)}{5} = 43,4 m\sec$$

$$\text{Μ.Χ.Α} = \frac{\big((0-0)+(48-8)\big)+(29-5)+(8-8)+(57-11)+(52-15)}{5} = 29,4 m\sec$$

ε)

$$\text{ΜΧΔ} = \frac{(20-0)+(65-5)+(70-8)+(62-11)+(41-15)}{5} = 43,8$$

$$\text{Μ.Χ.Α} = \frac{\big((0-0)+(16-8)\big)+\big((8-5)+(41-16)+(62-49)\big)+\big((20-8)+(49-28)+(65-57)\big)+\big((28-11)+(57-36)\big)+(36-15)}{5} = 29,8$$