ΔΕΛΗΓΙΑΝΝΗ ΜΥΡΤΩ 1067389

ΝΙΚΟΛΟΥΔΗΣ ΠΑΝΑΓΙΩΤΗΣ 1067076

ΠΑΝΑΪΚΑΣ ΣΩΤΗΡΙΟΣ 1067412

ΜΕΡΟΣ Α

Ερώτημα Α

Το πρόγραμμα περιλαμβάνει τις βιβλιοθήκες:

- unistd.h, που περιέχει τη fork(),
- sys/wait.h, που περιέχει τις WAITPID(), WIFEXITED(), WEXITSTATUS().

Στη main(), αρχικά δημιουργεί έναν πίνακα pid[N] μεταβλητών τύπου pid_t (unsigned int/μη-προσημασμένος ακέραιος) N=30 θέσεων. Έπειτα, μπαίνει σε βρόγχο επανάληψης και για 30 φορές κάνει τα εξής: Δημιουργεί ένα παιδί για κάθε διεργασία, χρησιμοποιώντας την εντολή fork() και το τοποθετεί στην i-οστή θέση του πίνακα. Έτσι κάθε κελί περιέχει το id του εκάστοτε παιδιού. Τέλος, ελέγχει αν όντως αναφερόμαστε σε παιδί της διεργασία. Αν ναι, το απενεργοποιεί για χρόνο που εξαρτάται από τον αριθμό της επανάληψης στην οποία βρισκόμαστε και το τερματίζει με έναν αριθμό κατάστασης εξόδου, ο οποίος πάλι εξαρτάται από τον αριθμό των επαναλήψεων.

Ύστερα για κάθε θέση του πίνακα, αναστέλλει την λειτουργία της κάθε γονικής διεργασίας μέχρι να τερματιστεί οποιοδήποτε παιδί του. Έπειτα, η WIFEXITED() ελέγχει αν έχει τερματιστεί ομαλά το παιδί αυτό και αν ναι, τότε εκτυπώνει την κατάσταση εξόδου του. Αλλιώς επιστρέφει το κατάλληλο μήνυμα για μη ομαλό τερματισμό της εκάστοτε διεργασίας.

ΜΕΡΟΣ Β

Ερώτημα Α

X:=X+1;	X :=Y+1;
1) TX:=X;	4) TY:=Y;
2) TX:=TX+1;	5) TY:=TY+1;
3) X:=TX;	6) X:=TY;

A)1,4,2,3,5,6

TX:=X; //TX=0

TY:=Y; //TY=10

TX:=TX+1; //TX=1

X:=TX; //X=1

TY:=TY+1; //TY=11

X:=TY; //X=11

B)1,4,5,6,2,3

TX:=X; //TX=0

TY:=Y; //TY=10

TY:=TY+1; //TY=11

X:=TY; //X=11

TX:=TX+1; //TX=1

X:=TX; //X=1

C)4,1,5,6,2,3

TY:=Y; //TY=10

TX:=X; //TX=1

TY:=TY+1; //TY=11

X:=TY; //X=11

TX:=TX+1; //TX=1

X:=TX; //X=1

D)4,1,2,3,5,6

TY:=Y; //TY=10

TX:=X; //TX=0

TX:=TX+1; //TX=1

X:=TX; //X=1

TY:=TY+1; //TY=11

X:=TY; // X=11

E)4,5,6,1,2,3

TY:=Y; //TY=10

TY:=TY+1; //TY=11

X:=TY; //X=11

TX:=X; //TX=11

TX:=TX+1; //TX=12

X:=TX; //X=12

F)1,2,3,4,5,6

TX:=X; //TX=0

TX:=TX+1; //TX=1

X:=TX; //X=1

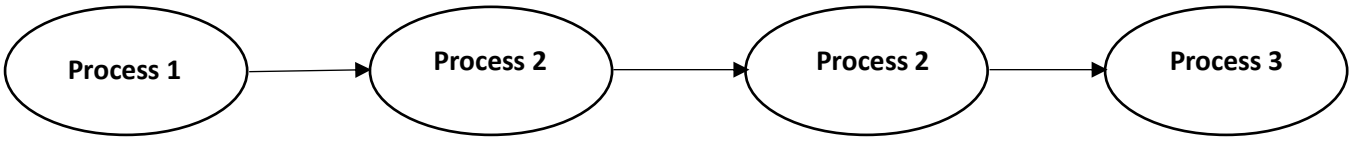
TY:=Y; //TY=10

TY:=TY+1; //TY=11

X:=TY; //X=11

ΕΡΩΤΗΜΑ Β:

a) Αρχικά,εφοσον θελούμε να τυπώσουμε την γραμματοσειρά 'PIZZA' πρέπει πρωτα να εκτελεστεί η **Process 1** οπου να εκτυπώσει 'PI',έπειτα δυο φορές η **Process 2** ώστε να εκτυπωθει ο διφθογγος 'ZZ' και τέλος η **Process 3** όπου εκτυπώνει 'A'. Όλες οι εκτύπωσεις συνολίκα φτιάχνουν την γραμματοσείρα 'PIZZA'. Ο γράφος προτεραιότητας λοιπόν για τις παραπάνω διεργασίες είναι ο εξής:



Η αρχικοποιήση των σημαφόρων γινεται ως εξης:

var s2,s3,s_{stop}: semaphores;

s2=0; s3=0; s_{stop}=0;

Process 1

```
while (TRUE)
{
print("P");
print("I");
signal(s2);
wait(s3);
signal(s2);
wait(s3);
signal(s3);
}
```

Process 2

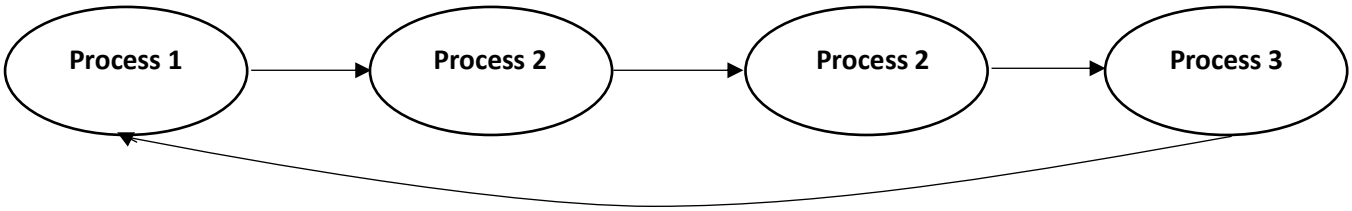
```
while (TRUE)
{
wait(s2);
print("Z");
signal(s3);
}
```

Process 3

```
while(TRUE)
```

{
wait(s3);
print("A");
wait(s_{stop});
}

b) Εφόσον λοιπόν τώρα επιθύμουμε το πρόγραμμα να εκτυπώνει 'PIZZAPIZZAPIZZA…' , δηλαδή το προηγούμενο πρόγραμμα σε ατέρμονο βρόγχο, πρέπει να βρούμε έναν τρόπο να δημιούργησουμε την δομή επανάληψης με σημαφόρους. Ο γράφος προτεραιότητας των διεργασιών μετατρεπεται στον εξής:



Η αρχικοποιήση των σημαφόρων γινεται ως εξης:

var s2,s3, s_{loop} : semaphores;

s2=0; s3=0; s_{loop}=1;

Process 1

while (TRUE)
{
wait(s_{loop});
print("P");
print("I");
signal(s2);
wait(s3);
signal(s2);
wait(s3);
signal(s3);
}

Process 2

while (TRUE)
{
wait(s2);
print("Z");
signal(s3);
}

Process 3

while(TRUE)
{
wait(s3);
print("A");
signal(s_{loop});

}

Ερώτημα Γ

ΠΡΩΤΗ ΠΕΡΙΠΤΩΣΗ:

A1,A2,B1,A3,B2

A1)s1=2,s2=0

Down(s1); //s1=1

Up(s2); //s2=1

A2)s1=1,2=1

Down(s1); //s1=0

Up(s2); //s2=2

B1)s1=0,s2=2

Down(s2); //s2=1

Down(s2); //s2=0

Up(s1); //s1=1

Up(s2); //s2=1

A3)s1=1,s2=1

Down(s1); //s1=0

Up(s2); //s2=2

B2)s1=0,s2=2

Down(s2); //s2=1

Down(s2); //s2=0

Up(s1); //s1=1

Up(s2); //s2=1

ΔΕΥΤΕΡΗ ΠΕΡΙΠΤΩΣΗ:

A1,A2,B1,B2,A3

A1)s1=2,s2=0

Down(s1); //s1=1

Up(s2); //s2=1

A2)s1=1,2=1

Down(s1); //s1=0

Up(s2); //s2=2

B1)s1=0,s2=2

Down(s2); //s2=1

Down(s2); //s2=0

Up(s1); //s1=1

Up(s2); //s2=1

B2)s1=1,s2=1

Down(s2); //s2=0

Down(s2); //s2=0

Up(s1);

Up(s2);

A3)s1=1,s2=1

Down(s1);

Up(s2);

Στην πρώτη περίπτωση δεν παρατηρείται κάποια παράβαση , οπότε ο κώδικας εκτελείται κανονικά χωρίς κάποιο πρόβλημα. Σε αντίθεση με την δεύτερη περίπτωση , όπου στην κλήση της B2 διεργασίας παρατηρείται σφάλμα για τον σημαφόρο s2 καθώς εισέρχεται σε ατέρμονο βρόγχο περιμένοντας να ενεργοποιηθεί χωρίς αυτό να είναι δυνατόν.

<u>Ερώτημα Δ</u>

(α) Κατά την παράλληλη εκτέλεση των διεργασιών, ο παραπάνω κώδικας μπορεί να οδηγήσει σε κάποιο μη επιθυμητό αποτέλεσμα. Αυτό συμβαίνει λόγω της έλλειψης ελέγχου και συγχρονισμού μέσω σημαφόρων. Έτσι, πολλές διεργασίες μπορούν να αποκτήσουν ταυτόχρονα πρόσβαση στην κρίσιμη περιοχή, με αποτέλεσμα οι διαμοιραζόμενες μεταβλητές να μην ανανεώνονται πάντα με τον κατάλληλο τρόπο. Για παράδειγμα, αν και οι N διεργασίες εκτελεστούν ταυτόχρονα, θα χρησιμοποιήσουν τις ίδιες τιμές των K και L (δηλ. K = L = 1) και επομένως

θα εκτελεστεί N φορές η ρουτίνα print_num(), άρα θα τυπώνονται οι ίδιοι 10 αριθμοί (δηλ. οι αριθμοί 1 – 11) για κάθε διεργασία.

(β) Όπως αναφέραμε και παραπάνω, το πρόβλημα αυτό οφείλεται στην έλλειψη συγχρονισμού των διεργασιών μέσω σημαφόρων, οπότε για να εξασφαλιστεί το απαιτούμενο αποτέλεσμα, θα πρέπει να γίνουν στον αρχικό κώδικα τη κάθε διεργασίας οι εξής τροποποιήσεις:

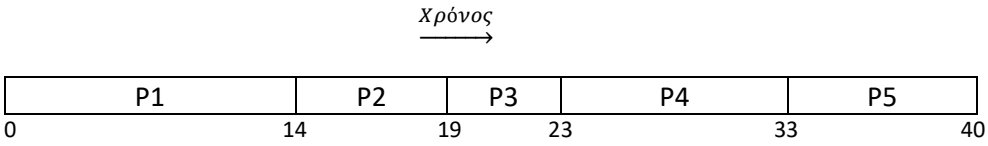
```
shared var K = L =1;
var s1, s2, s3, … , sn: semaphores;
s1 = 1, s2 = 0 ; s3 = 0; …, sn = 0;
```

Process_i

```
down(si);
while(TRUE){
  L := K;
  K := K + 11;
  print_num(L, L + 10);
  up(si+1);
}
```

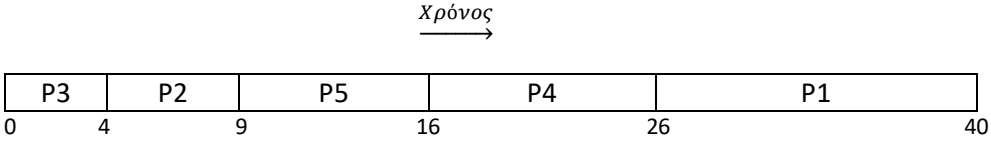
<u>Ερώτημα Ε</u>

- FCFS (First Come First Reserved): οι διεργασίες τοποθετούνται στην ουρά με τον τρόπο που φθάνουν στο σύστημα.



Από το διάγραμμα Gantt παρατηρούμε ότι:
ΧΔΡ1 = 14 ΧΔΡ2 = 19 ΧΔΡ3 = 23 ΧΔΡ4 = 33 ΧΔΡ5 = 40
Άρα ΜΧΔ = (ΧΔΡ1 + ΧΔΡ2 + ΧΔΡ3 + ΧΔΡ4 + ΧΔΡ5)/5 = (14 + 19 + 23 + 33 + 40)/5 = 129/5 = 25.8.
Επίσης,
ΧΑΡ1 = 0 ΧΑΡ2 = 14 ΧΑΡ3 = 19 ΧΑΡ4 = 22 ΧΑΡ5 = 33
Άρα ΜΧΑ = (ΧΑΡ1 + ΧΑΡ2 + ΧΑΡ3 + ΧΑΡ4 + ΧΑΡ5)/5 = (0 + 14 + 19 +23 + 33)/5 = 89/5 = 17.8.

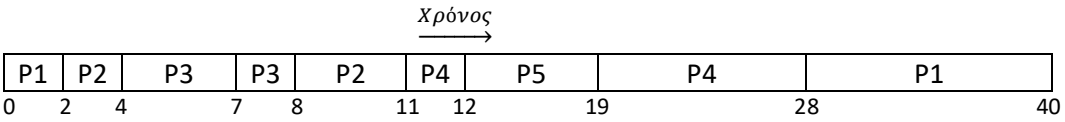
- SJF (Shortest Job First): οι διεργασίες δρομολογούνται με αύξουσα σειρά βάσει του χρόνου εκτέλεσης (πρώτα η πιο σύντομη, ακολουθεί η αμέσως επόμενη…).



Από το διάγραμμα Gantt παρατηρούμε ότι:
ΧΔΡ1 = 40 ΧΔΡ2 = 9 ΧΔΡ3 = 4 ΧΔΡ4 = 26 ΧΔΡ5 = 16

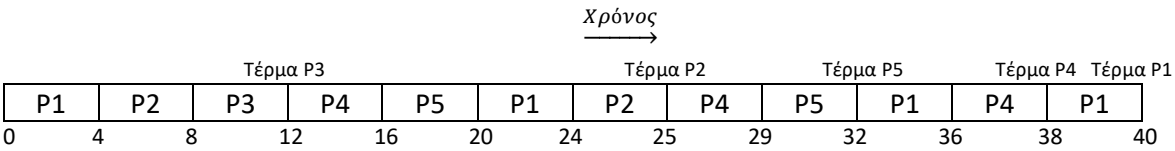
Άρα ΜΧΔ = (ΧΔΡ1 + ΧΔΡ2 + ΧΔΡ3 + ΧΔΡ4 + ΧΔΡ5)/5 = (40 + 9 + 4 + 26 + 16)/5 = 95/5 = 19.
Επίσης,
ΧΑΡ1 = 26 ΧΑΡ2 = 4 ΧΑΡ3 = 0 ΧΑΡ4 = 16 ΧΑΡ5 = 9
Άρα ΜΧΑ = (ΧΑΡ1 + ΧΑΡ2 + ΧΑΡ3 + ΧΑΡ4 + ΧΑΡ5)/5 = (26 + 4 + 0 + 16 + 9)/5 = 55/5 = 11.

- SRTF (Shortest Remaining Time First): λαμβάνοντας υπόψη το χρόνο άφιξης κάθε διεργασίας, δρομολογεί τις διεργασίες ανάλογα με τον χρόνο εκτέλεσης που απομένει.



Από το διάγραμμα Gantt παρατηρούμε ότι:
ΧΔΡ1 = 40 ΧΔΡ2 = 11 ΧΔΡ3 = 8 ΧΔΡ4 = 28 ΧΔΡ5 = 19
Άρα ΜΧΔ = (ΧΔΡ1 + ΧΔΡ2 + ΧΔΡ3 + ΧΔΡ4 + ΧΔΡ5)/5 = (40 + 11 + 8 + 28 + 19)/5 = 106/5 = 21.2.
Επίσης,
ΧΑΡ1 = (40 – 2) = 38 ΧΑΡ2 = (11 – 2) = 9 ΧΑΡ3 = 4 ΧΑΡ4 = (28 – 1) = 27 ΧΑΡ5 = 12
Άρα ΜΧΑ = (ΧΑΡ1 + ΧΑΡ2 + ΧΑΡ3 + ΧΑΡ4 + ΧΑΡ5)/5 = (38 + 9 + 4 + 27 + 12)/5 = 70/5 = 14.

- RR (Round Robin): αποδίδει περιοδικά 4 μονάδες του χρόνου της ΚΜΕ σε κάθε μία από τις διεργασίες της ουράς εκτέλεσης ξεκινώντας από εκείνη που είναι πρώτη στην ουρά και συνεχίζοντας προς το τέλος της.



Από το διάγραμμα Gantt παρατηρούμε ότι:
ΧΔΡ1 = 40 ΧΔΡ2 = 25 ΧΔΡ3 = 12 ΧΔΡ4 = 38 ΧΔΡ5 = 32
Άρα ΜΧΔ = (ΧΔΡ1 + ΧΔΡ2 + ΧΔΡ3 + ΧΔΡ4 + ΧΔΡ5)/5 = (40 + 25 + 12 + 38 + 32)/5 = 147/5 = 29.4.
Επίσης,
ΧΑΡ1 = (40 – 14) = 26 ΧΑΡ2 = (25 – 5) = 20 ΧΑΡ3 = 8 ΧΑΡ4 = (38 – 10) = 28
ΧΑΡ5 = (36 – 7) = 29
Άρα ΜΧΑ = (ΧΑΡ1 + ΧΑΡ2 + ΧΑΡ3 + ΧΑΡ4 + ΧΑΡ5)/5 = (26 + 20 + 8 + 28 + 29)/5 = 111/5 = 22.2.