Λειτουργικά Project A

Μέλη ομάδας: Μπουλαφέντης Χρήστος Ντάκος Γιώργος Παπαβασιλείου Μάριος

ΑΜ μελών: 1059612 1059569 1059649

Μέρος Α

---------------------------------------------------------------------------------------------------------------------------------------

Ερώτημα Α:

Το πρόγραμμα γεννάει μια διεργασια παιδί και έπειτα η κάθε μια ξεχωριστά γράφει “Ειμαι γονέας” ή “Ειμαι παιδι” αντίστοιχα από 200 φορές η κάθε μία (400 μηνύματα συνολικά).

Ερώτημα Β: (Αρχείο κώδικα επίσης συννημένο)

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i=0;

for(i=0;i<10;i++)

{

if(fork()==0) //Δημιουργεί το παιδί και έπειτα η printf και exit εκτελείται μόνο απο το παιδι.

{

printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());

exit(0); //Τερματίζει το παιδί για να μην τρέξει περα απο αυτό το σημειο.

}

}

return(0);

}

Ερώτημα Γ: (Αρχείο κώδικα επίσης συννημένο)

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i=0,pid;

for(i=0;i<10;i++)

{

pid=fork();

if(pid==0)

{

printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());

}

else goto end; //Ο πατέρας παει στο wait(NULL) και περιμενει

//τα παιδια να τελειωσουν και δεν γενναει αλλο.

}

end:

wait(NULL);

return(0);

}

Ερώτημα Δ: (Αρχείο κώδικα επίσης συννημένο)

#include<stdio.h>

#include<stdlib.h>

void foo();

int main()

{

time\_t start,end;

int i=1,pid=1;

long double temp;

start=time(NULL);

printf("Initial value of seconds: %ld \n",start);

while (i<=10000 && pid>0) ///Εχουμε βαλει 10000 γιατι με μικρότερα i

{ //το προγραμμα τελειωνε σε λιγοτερο απο 1

i=i+1; //δευτερολεπτο, ενω για μεγαλυτερα

pid=fork(); //ετρεχε χωρις να κανει κανενα printf.

if (pid==0) foo();

}

i=1;

while (i<=10000 && pid>0)

{

i=i+1;

waitpid();

}

if (pid>0)

{

end=time(NULL);

printf("Finishing value of seconds: %ld \n",end);

temp=end-start;

printf("Completion time=%Lf seconds.\n",temp);

temp=temp/10000;

printf("Average execution time: %Lf seconds per proccess.\n",temp);

}

return (0);

}

void foo()

{

int x=0;

x=x+10;

}

Μέρος Β:

Ερώτημα Αι:

binary semaphore s32.11=1;

binary semaphore s11.21=0;

binary semaphore s12.22=0;

binary semaphore s22.32=0;

binary semaphore s21.31=0;

cobegin

Process1

{

for k = 1 to 10 do

begin

DOWN s32.11

Ε1.1;

UP s11.21

Ε1.2;

UP s12.22

end

}

Process2

{

for j = 1 to 10 do

begin

DOWN s11.21

Ε2.1;

UP s21.31

DOWN s12.22

Ε2.2;

UP s22.32

end

}

Process3

{

for l = 1 to 10 do

begin

DOWN s21.31

Ε3.1;

DOWN s22.32

Ε3.2;

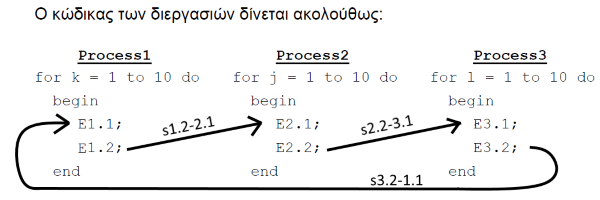
UP s32.11

end

}

coend

Ερώτημα Αιι)



binary semaphore s32.11=1;

binary semaphore s12.21=0;

binary semaphore s22.31=0;

cobegin

Process1

{

for k = 1 to 10 do

begin

DOWN s32.11

Ε1.1;

Ε1.2;

UP s12.21

end

}

Process2

{

for j = 1 to 10 do

begin

DOWN s12.21

Ε2.1;

Ε2.2;

UP s22.31

end

}

Process3

{

for l = 1 to 10 do

begin

DOWN s22.31

Ε3.1;

Ε3.2;

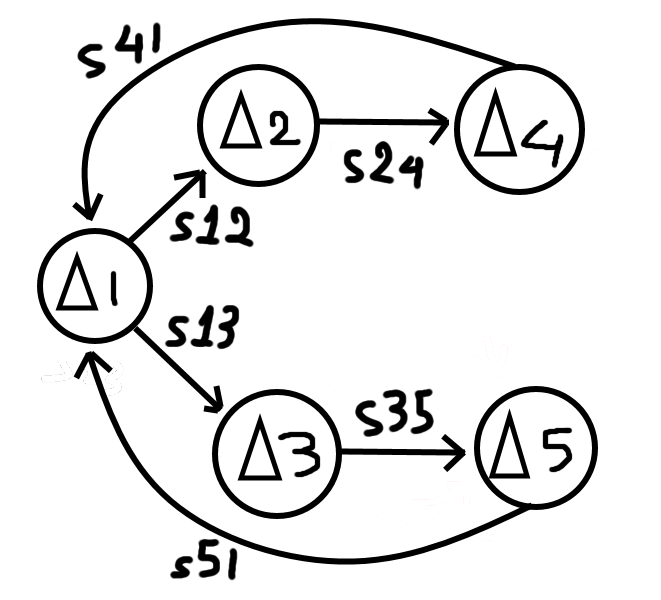
UP s32.11

end

}

coend

Ερώτημα Bι)



Οι διεργασιες Δ2 και Δ3 υποχρεωτικαπεριμενουν την Δ1 να τελειώσει καθως παιρνουν δεδομενα απο τον buf1 για την οποια τιμη του ειναι υπευθυνη η Δ1. Η Δ4 και Δ5 διαβαζουν απο buf2 και buf3 αντιστοιχα, οπου για τις τιμες τους ειναι υπευθυνες οι Δ2 και Δ3 αντιστοιχα. Ετσι η Δ4 περιμενει την ληξη της Δ2 και η Δ5 την ληξη της Δ3. Στο τελος η Δ1 περιμενει να τελειωσουν οι Δ4 και Δ5 για να ξεκινησει νεος κυκλος των διεργασιων.

Ερώτημα Bιι)

binary semaphore s12=0,s13=0,s24=0,s35=0,s41=1,s51=1;

Shared int y=0,z=0;

Δ1:

DOWN s41;

DOWN s51;

Int x1;

X1=random(1-10);

write(buf1,x1);

UP s12;

UP s13;

Δ2: Δ3:

DOWN s12; DOWN s13;

Int x2; nt x3;

Read(buf1,x2); Read(buf1,x3);

Write(buf2,x2); Write(buf3,x3);

Up s24; Up s35;

Δ4: Δ5:

DOWN s24; DOWN s35;

Int x4; Int x5;

x4=random(1-10); x5=random(1-10);

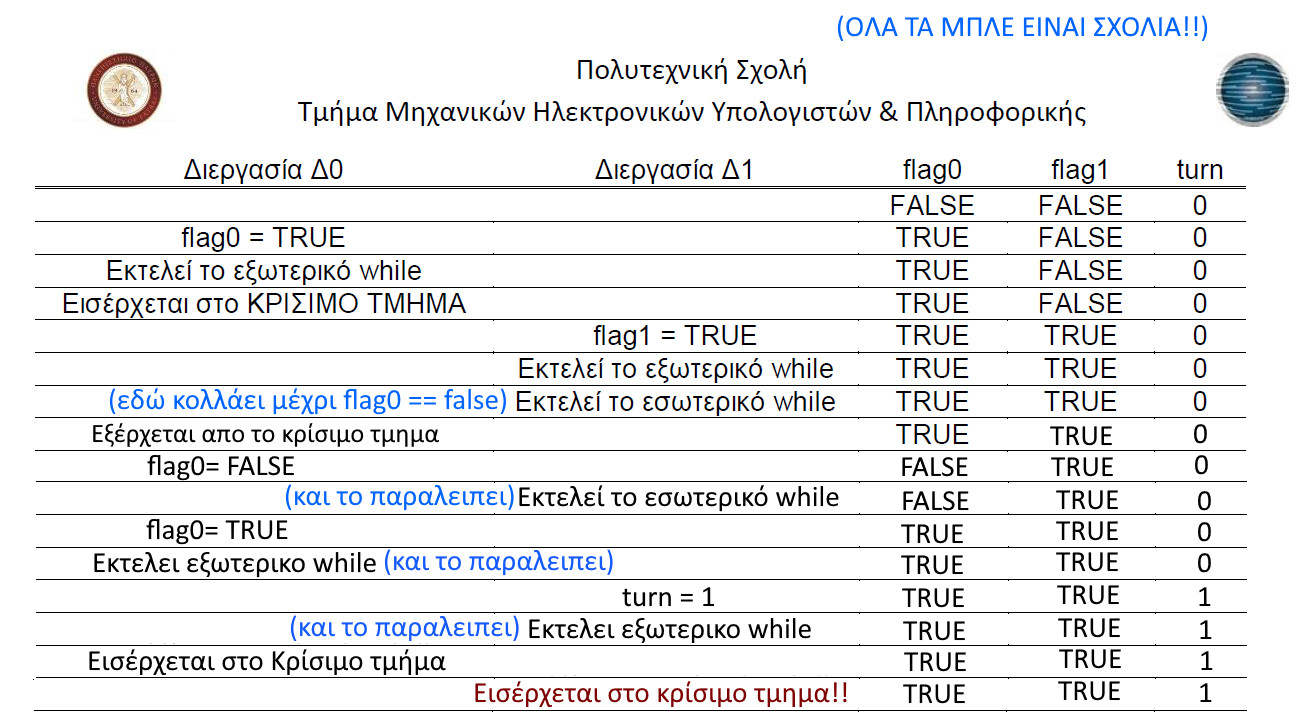
Read(buf2,y); Read(buf3,z);

y=y+x4; z=z+x5;

If (y>z) printf(“Δ4”); If (z>=y) printf(“Δ5”);

UP s41; UP s51;

Ερώτημα Γ)



Ερώτημα Δ)

Ο mutex σημαφόρος επιτυγχάνει τον συγχρονισμό της σχέσης ενος supervisor με 3 workers. Με την απουσια του υπαρχει κινδυνος παραπανω supervisors να επιτηρουν λιγοτερους απο 3 εργατες ο καθενας. Για παραδειγμα, αν φτασουν στο εργοταξιο 3 supervisors και 6 εργατες συμφωνα με το κανονισμο θα πρεπει 2 supervisors να επιβλεπουν 3 εργατες ο καθένας, ενω ο τριτος supervisor

Θα πρεπει να περιμενει αλλους 3 εργατες να φτασουν στο εργοταξιο. Χωρις το σημαφορο mutex

υπαρχει περιπτωση και οι 3 supervisors να κανουν wait(w) απο 2 φορές ο καθένας, με αποτελεσμα οι 6 εργατες να επιτηρουντε απο 3 supervisors, κατι το οποιο απαγορευεται απο τον κανονισμο.

Με την υπαρξη του mutex τελειωνει την επιβλεψη ο πρωτος εργατης χωρις διακοπη και φευγει απο το εργοταξιο, το ιδιο και ο δευτερος ενω ο τριτος περιμενει να φτασουν αλλοι workers, τηρωντας ετσι τον κανονισμο και τον σωστο συγχρονισμο των διεργασιων.

Ερώτημα Ε)

