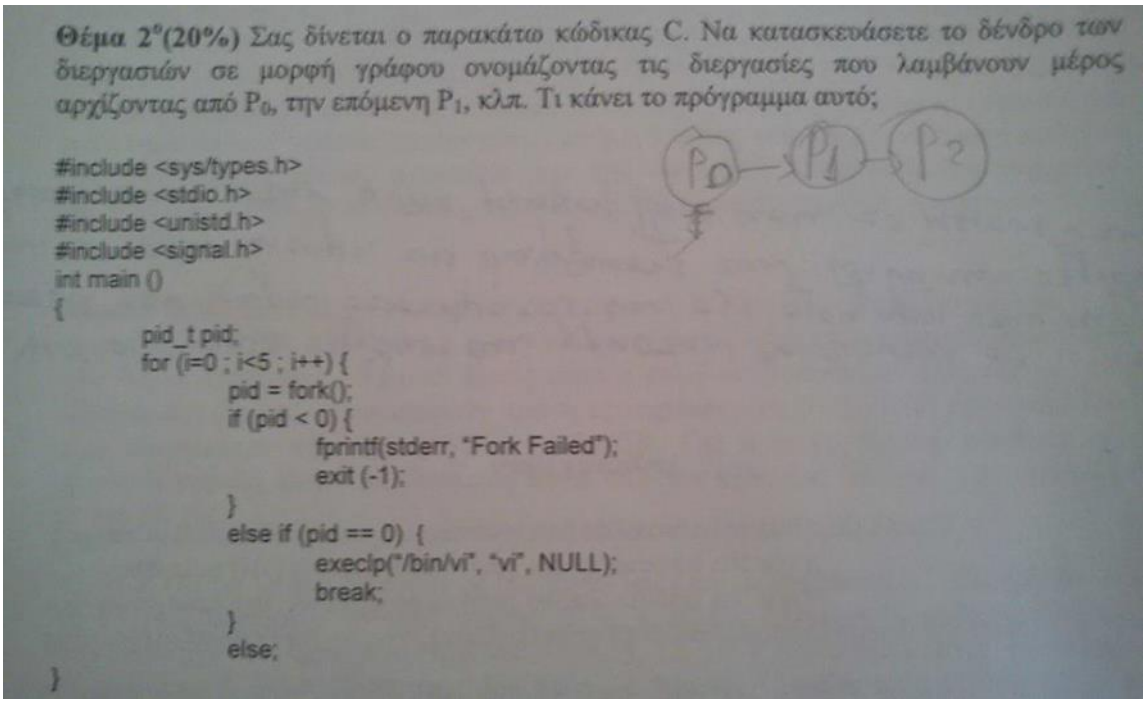


Θέμα 2°(20%) Σας δίνεται ο παρακάτω κώδικας C. Να κατασκευάσετε το δένδρο των διεργασιών σε μορφή γράφου ονομάζοντας τις διεργασίες που λαμβάνουν μέρος αρχίζοντας από P_0, την επόμενη P_1, κλπ. Τι κάνει το πρόγραμμα αυτό;

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
int main ()
{
        pid_t pid;
        for (i=0 ; i<5 ; i++) {
                pid = fork();
                if (pid < 0) {
                        fprintf(stderr, "Fork Failed");
                        exit (-1);
                }
                else if (pid == 0) {
                        execlp("/bin/vi", "vi", NULL);
                        break;
                }
                else;
}
```

Εδώ το μυστικό είναι να προσέξετε που βρίσκεται η break. Τι κάνει η break όταν είναι μέσα σε ένα for;
Μα φυσικά βγαίνει το πρόγραμμα από την for και δεν συμβαίνει άλλη επανάληψη.

Στο παράδειγμά μας, η break βρίσκεται μέσα στον κώδικα που εκτελεί το παιδί. Αυτό σημαίνει ότι το παιδί δεν Θα ξαναδιαβάσει την for που έχει κληρονομήσει από τον πατέρα..

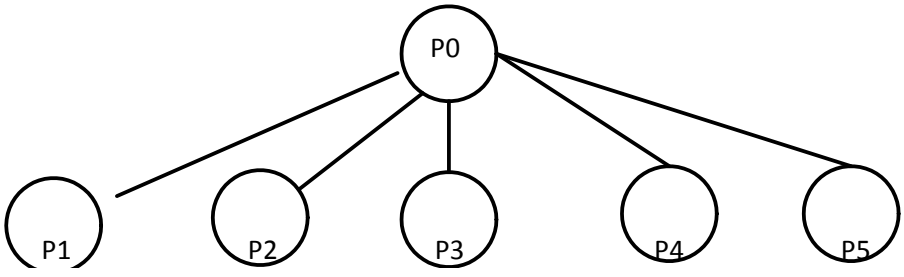
Θυμίζουμε εδώ ,ότι το παιδί κληρονομεί τον ίδιο κώδικα που έχει απο πριν ο πατέρας του... Για αυτόν τον λόγο Κληρονομεί και την for.

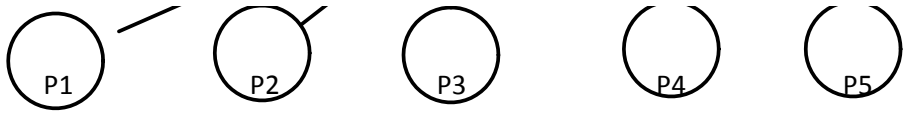
Αν είχαμε break και στον κώδικα που εκτελεί ο πατέρας, τότε θα έβγαινε και ο πατέρας από τις επαναλήψεις της for και δεν θα τις έκανε ούτε αυτός.

Τώρα... Τι κάνει αυτό το πρόγραμμα..
Από ότι βλέπουμε με μία for μια διεργασία δημιουργεί 5 παιδιά.
Το κάθε παιδί απλά ανοίγει το πρόγραμμα επεξεργαστή κειμένου vi.
Ο πατέρας κάθε φορά το μόνο που κάνει είναι να επιστρέφει στην fork για να δημιουργήσει το επόμενο παιδί μέχρι να γίνουν 5.

Η δεντρική μορφή είναι πολύ απλή.
Ο πατέρας P0 δημιουργεί 5 παιδιά, τα P1,P2,P3,P4,P5

Αρα

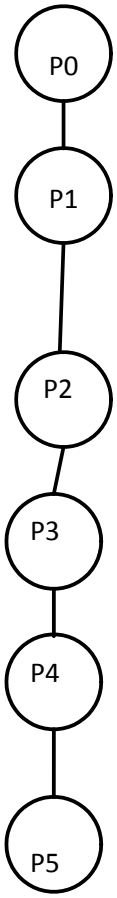




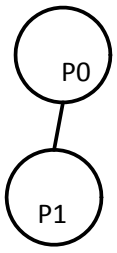
Στην περίπτωση που το break ήταν στον κώδικα του πατέρα και όχι στου παιδιού,

ο κάθε φορά πατέρας θα σταματούσε να κάνει παιδιά, ενώ το κάθε φορά παιδί θα συνέχιζε αυτήν την λειτουργία.

Έτσι ο πατέρας P0 θα δημιουργούσε το παιδί P1 και θα σταματούσε μιας και θα έβγαινε από την for.
Μετά το παιδί P1 θα δημιουργούσε (σαν πατέρας τώρα πια) το παιδί P2 και θα σταματούσε.
Το παιδί P2 θα έκανε το παιδί P3 και πάει λέγοντας. ¨Ετσι το σχήμα θα ήτανε



Βλεέπετε ότι προκύπτει τελείως διαφορετικό δέντρο...
Για αυτό θέλει προσοχή... Πάντα **μα πάντα** κοιτάμε να εντοπίσουμε την break..
Αν πάλι δεν έχει ούτε break ούτε for το δέντρο μας είναι πολύ πιο εύκολο μιας και απλά ο πατέρας P0 δημιουργεί ένα παιδί P1 και σταματάει.

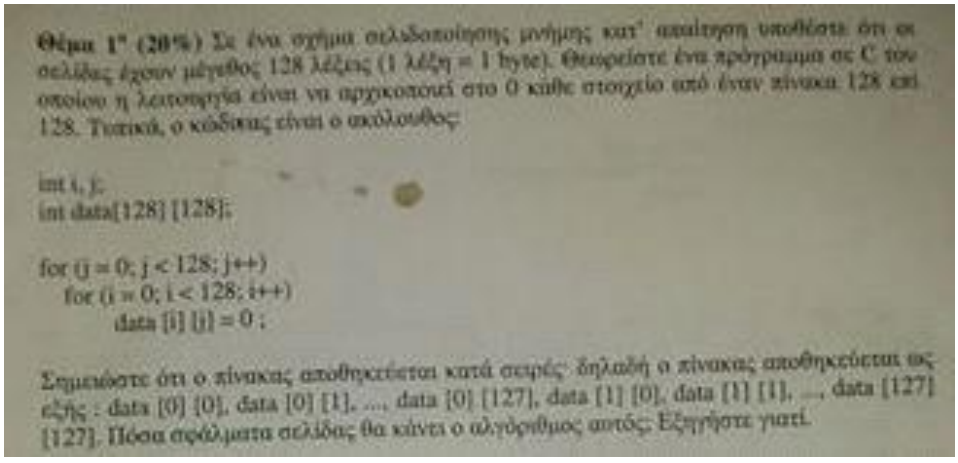


Η περίπτωση να υπάρχει for και να μην υπάρχει break είναι απίθανο να ζητηθεί γιατί αντί για δέντρο Θα έπρεπε να ζωγραφίσουμε περικοκλάδα.. Αφού το κάθε παιδί θα γινόταν πατέρας και ο κάθε πατέρας θα συνέχιζε να κάνει παιδιά!!!!!

Επίσης άλλο ένα πράγμα που πρέπει να προσέξετε...

Όταν σας ρωτάει τι κάνει το πρόγραμμα, κοιτάτε τι περιέχει μέσα η exec... Και ανάλογα απαντάτε.. Αν περιέχει πχ ps-A εμφανίζει διεργασίες... Και τα γνωστά.... Είναι οι εντολές που μαθαίνουμε στο εργαστήριο στην αρχή...

Πάμε τώρα να δούμε το δεύτερο ανάλαφρο θεματάκι..



Θέμα 1ο (20%) Σε ένα σχήμα σελιδοποίησης μνήμης κατ' απαίτηση υποθέστε ότι οι σελίδες έχουν μέγεθος 128 λέξεις (1 λέξη = 1 byte). Θεωρείστε ένα πρόγραμμα σε C του οποίου η λειτουργία είναι να αρχικοποιεί στο 0 κάθε στοιχείο από έναν πίνακα 128 επί 128. Τυπικά, ο κώδικας είναι ο ακόλουθος:

```
int i, j;
int data[128] [128];

for (j = 0; j < 128; j++)
   for (i = 0; i < 128; i++)
        data [i] [j] = 0 ;
```

Σημειώστε ότι ο πίνακας αποθηκεύεται κατά σειρές· δηλαδή ο πίνακας αποθηκεύεται ως εξής : data [0] [0], data [0] [1], ..., data [0] [127], data [1] [0], data [1] [1], ..., data [127] [127]. Πόσα σφάλματα σελίδας θα κάνει ο αλγόριθμος αυτός; Εξηγήστε γιατί.

Εδώ πρέπει απλά να προσέξουμε την σειρά που έχουν τα i και j στις for..

Να εξηγήσουμε τι εννοούμε..

Καταρχήν μας λένε ότι ο πίνακας αποθηκεύεται σε σειρές και επίσης ότι ο πίνακας είναι 128 σειρές Και 128 στήλες.
Μιας και ο πίνακας αποθηκεύεται σε σειρές, αυτό σημαίνει ότι κάθε σειρά αφού αποτελείται από 128 στοιχεία, αποτελεί και μία σελίδα . Σφάλμα σελίδας εννοούμε όταν αλλάζουμε σελίδα..
Αφού λοιπόν κάθε γραμμή του πινακα είναι μια σελίδα, τότε σφάλμα σελίδας θα έχουμε όταν αλλάζουμε γραμμη.

Και τώρα ερχόμαστε στομ κώδικα..

Πρώτα μας ενδιαφέρει το κομμάτι data[i][j]=0;

Αυτό που ξέρουμε από την c είναι ότι τα στοιχεία ενός πίνακα αναφέρονται πρώτα στην γραμμή και μετά στην στήλη τους . Έτσι όταν λέμε data[i][j] εννοούμε το στοιχείο που βρίσκεται στην γραμμή i και στην στήλη j.
Στο δικό μας παράδειγμα, αφού έχουμε data[i][j] αυτό σημαίνει ότι οι γραμμές συμβολίζονται με i στον κώδικά μας.
Τώρα το μόνο που έχουμε να κάνουμε είναι να δούμε πόσες φορές αλλάζει το i μέσα στην διπλή for του κώδικα.

Βλέπουμε ότι υπάρχει μια for που περιέχει το i και μέσα σε αυτήν αλλάζει 128 φορές...
Παρατηρούμε όμως ότι αυτή η for βρίσκεται μέσα σε μια άλλη for που αφορά το j.
Αυτό σημαίνει ότι κάθε φορά που αλλάζει το j, σύμφωνα με την εσωτερική for , το i αλλάζει 128 φορές. Άρα αφού και το j αλλάζει 128 φορές, τα συνολικά σφάλματα σελίδας είναι 128*128.

Αν οι 2 for ήταν ανάποδα <u>ή</u> αν είχαμε data[j][i] αντί για data[i][j], τότε τα συνολικά σφάλματα θα ήταν 128 μόνο.

Αυτά τα δύο ανάλαφρα θεματάκια για σήμερα, για να χαλαρώσουμε λιγάκι...
Από αύριο συνεχίζουμε με κάποια πιο μεγάλα θεματάκια και την θεωρία που τα διέπει..

Καλή μελέτη....