ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

$2^{η}$ εργαστηριακή άσκηση

Δημήτριος Ξενίας

03115084

Μιχαήλ Φιλίππου

03115756

1.1

i. Αν βάλουμε στο command line την ώρα που εκτελείται η διεργασίας A την εντολή kill –KILL <pid> όπου pid το process id της διεργασίας τότε όλα τα παιδιά της A θα περάσουν κάτω από την init η οποία έχει process id 1 και θα τα κάνει διαρκώς wait.

ii. Με την χρήση του pid στην show_pstree εμφανίζεται το δέντρο διεργασιών που δημιουργείται μέσα στην ask2-fork.

```
oslabe01@orion:~/ask2/forktree$ ./ask2-fork
A: Sleeping...
B: Sleeping...
C: Sleeping...
D: Sleeping...


A(25358)─┬─B(25359)──D(25361)
         └─C(25360)


C: Exiting...
D: Exiting...
My PID = 25358: Child PID = 25360 terminated normally, exit status = 17
My PID = 25359: Child PID = 25361 terminated normally, exit status = 13
B: Exiting...
My PID = 25358: Child PID = 25359 terminated normally, exit status = 19
A: Exiting...
My PID = 25357: Child PID = 25358 terminated normally, exit status = 16
```

Αν όμως βάλουμε αντί αυτού την getpid() εμφανίζονται επιπλέον οι διεργασίες sh και pstree που χρησιμοποιούνται από την ask2-fork.Η sh είναι η διεργασία η οποία εκτελεί τις εντολές από το τερματικό ή από ένα αρχείο,ενώ η ps_tree δείχνει την τρέχουσα διαδικασία ως δέντρο.

```
oslabe01@orion:~/ask2/forktree$ ./ask2-fork
A: Sleeping...
B: Sleeping...
C: Sleeping...
D: Sleeping...


ask2-fork(25235)─┬─A(25236)─┬─B(25237)──D(25239)
                 │          └─C(25238)
                 └─sh(25240)──pstree(25241)


C: Exiting...
D: Exiting...
My PID = 25236: Child PID = 25238 terminated normally, exit status = 17
My PID = 25237: Child PID = 25239 terminated normally, exit status = 13
B: Exiting...
My PID = 25236: Child PID = 25237 terminated normally, exit status = 19
A: Exiting...
My PID = 25235: Child PID = 25236 terminated normally, exit status = 16
```

iii. Στα υπολογιστικά συστήματα ο διαχειριστής θέτει όριο στον αριθμό των διεργασιών που μπορούν να δημιουργηθούν διότι αν δημιουργούνται ανεξέλεγκτα (πχ fork bomb) το σύστημα θα καταρρεύσει ,αφού δεν θα υπάρχουν πόροι του συστήματος διαθέσιμοι για όλες τις διεργασίες που θα παραχθούν.

1.2

Τα μηνύματα έναρξης των διεργασιών εμφανίζονται σύμφωνα με την κατά βάθος αναζήτηση (BFS) ενώ τα μηνύματα τερματισμού των διεργασιών δεν ακολουθούν κάποιο συγγεκριμένο μοτίβο.

```
oslabe01@os-node1:~/ask2/forktree$ ./ask2-tree proc.tree
A
        B
                E
                F
        C
        D
child with 30022 created
child with 30023 created
child with 30024 created
child with 30025 created
child with 30026 created


A(30021)─┬─B(30022)─┬─E(30025)
         │          └─F(30026)
         ├─C(30023)
         └─D(30024)


My PID = 30021: Child PID = 30023 terminated normally, exit status = 1
My PID = 30021: Child PID = 30024 terminated normally, exit status = 1
My PID = 30022: Child PID = 30025 terminated normally, exit status = 1
My PID = 30022: Child PID = 30026 terminated normally, exit status = 1
My PID = 30021: Child PID = 30022 terminated normally, exit status = 1
My PID = 30020: Child PID = 30021 terminated normally, exit status = 1
```

1.3

i. Με την χρήση της εντολής sleep() καθορίζουμε εμείς τον χρόνο που θα χρειαστεί μέχρι να δημιουργηθεί το δένδρο διεργασιών .Αυτή η τεχνική

όμως έχει το μειονέκτημα ότι όταν έχουμε ένα δέντρο με μεγάλο αριθμό διεργασιών δεν είναι εύκολο να υπολογίσουμε του χρόνους και επίσης δεν θα έχουμε σωστό συγχρονισμό μεταξύ των διεργασιών. Απεναντίας με την χρήση των σημάτων έχουμε τον απόλυτο έλεγχο στον συγχρονισμό καθώς επιλέγουμε σε τι κατάσταση χρειάζεται να είναι η κάθε διεργασία ακριβώς όποτε την χρειαζόμαστε.

ii. Η wait_for_ready_children() σιγουρεύει ότι όλα τα παιδιά της εκάστοτε διεργασίας που την κάλεσε έχουν γίνει pause και δεν έχουν τερματίσει για κάποιον άλλο λόγο .Αυτό γίνεται με τη την χρήση των τιμών που επιστρέφει η waitpid().Στο πρόγραμμά μας καλείται αναδρομικά για κάθε παιδί και στην συνέχεα κάνουν pause τον εαυτό τους .Άρα η χρήση της είναι να εξασφαλίζει ότι γνωρίζουμε την κατάσταση του δέντρου πριν προσπαθήσουμε να ξυπνήσουμε παιδία τα οποία ενδεχόμενος να μην

έχουν γίνει pause ή να μην έχουν δημιουργηθεί.

```
oslabe01@orion:~/ask2/forktree$ ./ask2-signals proc.tree
PID = 13941, name A, starting...
PID = 13942, name B, starting...
PID = 13943, name C, starting...
PID = 13945, name E, starting...
My PID = 13941: Child PID = 13943 has been stopped by a signal, signo = 19
My PID = 13942: Child PID = 13945 has been stopped by a signal, signo = 19
PID = 13944, name D, starting...
My PID = 13941: Child PID = 13944 has been stopped by a signal, signo = 19
PID = 13946, name F, starting...
My PID = 13942: Child PID = 13946 has been stopped by a signal, signo = 19
My PID = 13941: Child PID = 13942 has been stopped by a signal, signo = 19
My PID = 13940: Child PID = 13941 has been stopped by a signal, signo = 19


A(13941)─┬─B(13942)─┬─E(13945)
         │          └─F(13946)
         ├─C(13943)
         └─D(13944)


PID = 13941, name = A is awake
PID = 13944, name = D is awake
I am D and I am done
My PID = 13941: Child PID = 13944 terminated normally, exit status = 0
PID = 13943, name = C is awake
I am C and I am done
My PID = 13941: Child PID = 13943 terminated normally, exit status = 0
PID = 13942, name = B is awake
PID = 13946, name = F is awake
I am F and I am done
My PID = 13942: Child PID = 13946 terminated normally, exit status = 0
PID = 13945, name = E is awake
I am E and I am done
My PID = 13942: Child PID = 13945 terminated normally, exit status = 0
I am B and I am done
My PID = 13941: Child PID = 13942 terminated normally, exit status = 0
I am A and I am done
My PID = 13940: Child PID = 13941 terminated normally, exit status = 0
oslabe01@orion:~/ask2/forktree$
```

1.4

Για την χρήση του expr.tree το αποτέλεσμα ήταν:

```
oslabe01@orion:~/ask2/forktree$ ./ask2-pipes expr.tree
Eimai o  10 kai exo grapsei : 10
My PID = 13956: Child PID = 13957 terminated normally, exit status = 0
Eimai o + kai exo diavasei : 10
Eimai o  4 kai exo grapsei : 4
My PID = 13958: Child PID = 13960 terminated normally, exit status = 0
Eimai o * kai exo diavasei : 4
Eimai o  7 kai exo grapsei : 7
Eimai o  5 kai exo grapsei : 5
My PID = 13959: Child PID = 13961 terminated normally, exit status = 0
Eimai o + kai exo diavasei : 7
My PID = 13959: Child PID = 13962 terminated normally, exit status = 0
Eimai o + kai exo diavasei : 5
Eimai o + kai egrapsa : 12
My PID = 13958: Child PID = 13959 terminated normally, exit status = 0
Eimai o * kai exo diavasei : 12
Eimai o * kai egrapsa : 48
My PID = 13956: Child PID = 13958 terminated normally, exit status = 0
Eimai o + kai exo diavasei : 48
Eimai o + kai egrapsa : 58
My PID = 13955: Child PID = 13956 terminated normally, exit status = 0
Diavasa :58
 58
```

I. Στο πρόγραμμα μας κάθε διεργασία αλληλεπιδρά με 2 σωληνώσεις ,μια που διαβάζει και μία που γράφει ,εκτός από τις ακραίες διεργασίες οι οποίες μόνο γράφουν και και η ρίζα η οποία διαβάζει το τελικό αποτέλεσμα και το τυπώνει .Για την επικοινωνία της γονικής διεργασίας χρησιμοποιήθηκε ένα pipe το οποίο μοιράζονται τα 2 παιδιά .Σε αυτή την περίπτωση το κοινό Pipe είναι υλοποιήσιμο διότι δεν απαιτείται να ξέρουμε ποιο αποτέλεσμα προήλθε από ποιο παιδί διότι η πράξη του πολλαπλασιασμού και της πρόσθεσης είναι αντιμεταθετικές.