2 η Εργαστηριακή Άσκηση: Διαχείριση Διεργασιών και Διαδιεργασιακή Επικοινωνία

Άσκηση 2.1 Δημιουργία δεδομένου δέντρου διεργασιών

Το αποτέλεσμα της άσκησης όταν την τρέχουμε είναι:

```
george@georgepag4028:~/Desktop/code/os/p2.1$ ./ask2-fork
Creating root A...
A creating B...
A creating C...
B creating D...
C Sleeping...
D Sleeping...


A(15858)─┬─B(15859)──D(15861)
         └─C(15860)


C Exiting...
D Exiting...
My PID = 15858: Child PID = 15860 terminated normally, exit status = 17
My PID = 15859: Child PID = 15861 terminated normally, exit status = 13
B: All done,exiting...
My PID = 15858: Child PID = 15859 terminated normally, exit status = 19
A: All done,exiting...
My PID = 15857: Child PID = 15858 terminated normally, exit status = 16
george@georgepag4028:~/Desktop/code/os/p2.1$
```

1. Τι θα γίνει αν τερματίσετε πρόωρα τη διεργασία A, δίνοντας kill -KILL , όπου το Process ID της;

Τερματίσαμε πρόωρα την εκτέλεση του προγράμματος, ενσωματώνοντας στον κώδικας μας την εντολή
kill(pid,SIGKILL)
μετά την εκτελέση της εντολής fork() για την ρίζα του δέντρου A
και εκτυπώθηκε στην κονσόλα:

```
george@georgepag4028:~/Desktop/code/os/p2.2$ ./fork-tree proc.tree


fork-tree(21551)


My PID = 21550: Child PID = 21551 was terminated by a signal, signo = 9
george@georgepag4028:~/Desktop/code/os/p2.2$
```

το οποίο δηλώνει ότι η διεργασία παιδί με pid = 3437 σκοτώθηκε από την εντολή μας και εκτυπώνει signo = 9 μέσω της συνάρτησης explain_wait_status(pid,status).

2. Τι θα γίνει αν κάνετε show_pstree(getpid()) αντί για show_pstree(pid) στη main(); Ποιες επιπλέον διεργασίες φαίνονται στο δέντρο και γιατί;

```
george@georgepag4028:~/Desktop/code/os/p2.1$ ./ask2-fork
Creating root A...
A creating B...
A creating C...
B creating D...
C Sleeping...
D Sleeping...


ask2-fork(15678)─┬─A(15679)─┬─B(15680)──D(15682)
                 │          └─C(15681)
                 └─sh(15686)──pstree(15687)


C Exiting...
D Exiting...
My PID = 15679: Child PID = 15681 terminated normally, exit status = 17
My PID = 15680: Child PID = 15682 terminated normally, exit status = 13
B: All done,exiting...
My PID = 15679: Child PID = 15680 terminated normally, exit status = 19
A: All done,exiting...
My PID = 15678: Child PID = 15679 terminated normally, exit status = 16
george@georgepag4028:~/Desktop/code/os/p2.1$
```

Εμφανίζεται σαν root του δέντρου το ask2-fork που είναι το όνομα του εκτελέσιμου αρχείου, και το A θεωρείται σαν ενδιάμεσο node. Μετά το τέλος του υποδέντρου με ρίζα A, εμφανίζει και ένα ακόμη υπόδεντρο με 2 διεργασίες, την sh και την pstree, οι οποίες καλούνται στην main απο το show_pstree().

3. Σε υπολογιστικά συστήματα πολλαπλών χρηστών, πολλές φορές ο διαχειριστής θέτει όρια στον αριθμό των διεργασιών που μπορεί να δημιουργήσει ένας χρήστης. Γιατί;

Σε αυτό το σύστημα μπορεί να έχουν πρόσβαση αρκετοί χρήστες ,όπου οι υπόλοιποι θα θέλουν και αυτοί να τρέχουν δικά τους προγράμματα,δηλαδή διεργασίες. Οπότε , ο διαχειριστής περιορίζει το όριο των διεργασιών άρα και την μνήμης που μπορεί να χρησιμοποιήσει ο κάθε χρήστης, έτσι να ώστε να μπορούν όλοι οι χρήστες να τρέχουν τα προγράμματα τους χωρίς καθυστέρηση και επιβάρυνση του συστήματος.

2.2 Δημιουργία αυθαίρετου δέντρου διεργασιών

Το αποτελεσμα της ασκησης οταν την τρεχουμε με το proc.tree:

```
george@georgepag4028:~/Desktop/code/os/p2.2$ ./fork-tree  proc.tree
I am A,and about to create some children.
I am B,and about to create some children.
My name is C and I am going to sleep
My name is D and I am going to sleep
My name is E and I am going to sleep
My name is F and I am going to sleep


A(17004)─┬─B(17005)─┬─E(17008)
         │          └─F(17009)
         ├─C(17006)
         └─D(17007)


Child C woke up and exiting now...
Child D woke up and exiting now...
Child E woke up and exiting now...
My PID = 17004: Child PID = 17007 terminated normally, exit status = 0
Child F woke up and exiting now...
My PID = 17005: Child PID = 17008 terminated normally, exit status = 0
My PID = 17004: Child PID = 17006 terminated normally, exit status = 0
My PID = 17005: Child PID = 17009 terminated normally, exit status = 0
My PID = 17004: Child PID = 17005 terminated normally, exit status = 0
My PID = 17003: Child PID = 17004 terminated normally, exit status = 0
george@georgepag4028:~/Desktop/code/os/p2.2$
```

1. Με ποια σειρά εμφανίζονται τα μηνύματα έναρξης και τερματισμού των διεργασιών; γιατί;

Αρχικά, ξεκινάμε με το A κάνοντας fork() στην main. Μετά κάνοντας fork μέσα στην βοηθητική συνάρτηση, κάνουμε generate και το πρώτο παιδί, το B. Τρέχει για δεύτερη φορά η βοηθητική, κάνουμε generate το C, μετά θεωρείται root = B, οποτέ γίνεται generate το πρώτο παιδί της B. Σειρά έχει η εκτέλεση της συνήρτησης για parent = C , το οποίο είναι leaf άρα το C κάνει sleep(). Μέτα γίνεται generate το 2ο παιδί της B, και με την σειρά που έγιναν fork() τα εναπομείναντα leaf nodes, κάνουν sleep(), δηλαδή τα E D F . Αντίστοιχα, με την ίδια φόρα που δημιουργήθηκαν, καλούνται και οι συναρτήσεις τερμιτισμού. Δηλαδή, C E D F.

2.3 Αποστολή και χειρισμός σημάτων

Το αποτέλεσμα της άσκησης όταν την τρέχουμε με το δεντρό proc.tree:

```
george@georgepag4028:~/Desktop/code/os/p2.3$ ./ask2-signals proc.tree


A(18561)─┬─B(18562)─┬─E(18565)
         │          └─F(18566)
         ├─C(18563)
         └─D(18564)


Haha,hey mama B its me,E,your favorite child.
My PID = 18562: Child PID = 18565 terminated normally, exit status = 0
Haha,hey mama B its me,F,your favorite child.
My PID = 18562: Child PID = 18566 terminated normally, exit status = 0
Hey,I am B,ooo,those children are trying.Can I have a nap too?
My PID = 18561: Child PID = 18562 terminated normally, exit status = 0
Haha,hey mama A its me,C,your favorite child.
My PID = 18561: Child PID = 18563 terminated normally, exit status = 0
Haha,hey mama A its me,D,your favorite child.
My PID = 18561: Child PID = 18564 terminated normally, exit status = 0
Hey,I am A,ooo,those children are trying.Can I have a nap too?
My PID = 18560: Child PID = 18561 terminated normally, exit status = 0
```

1. Στις προηγούμενες ασκήσεις χρησιμοποιήσαμε τη sleep() για τον συγχρονισμό των διεργασιών. Τι πλεονεκτήματα έχει η χρήση σημάτων;

Με τη χρήση σημάτων, έχουμε την δυνατότητα να τερματίζουν πρώτα οι λειτουργίες των nodes που έχουν τον μεγαλύτερο βαθμό στο δεντρο και μετά τερματίζουν οι λειτουργίες των nodes που έχουν μικροτέρο, καταλήγοντας στον A, που είναι το root.

2. Ποιος ο ρόλος της wait_for_ready_children(); Τι εξασφαλίζει η χρήση της και τι πρόβλημα θα δημιουργούσε η παράλειψή της;

Αυτή η συνάρτηση εξασφαλίζει ότι όλα τα παιδιά έχουν κάνει raise(SIGSTOP).
Η παράλειψή της θα έκανε κάποιες μητρικές διεργασίες να τελειώσουν προτού, τελειώσουν όλες οι διεργασίες-παιδιά.

2.4 Παράλληλος υπολογισμός αριθμητικής έκφρασης

Το αποτέλεσμα της άσκησης όταν τρέχουμε στην άσκηση το αρχείο expr.tree είναι:

1. Πόσες σωληνώσεις χρειάζονται στη συγκεκριμένη άσκηση ανά διεργασία; Θα μπορούσε κάθε γονική διεργασία να χρησιμοποιεί μόνο μία σωλήνωση για όλες τις διεργασίες παιδιά; Γενικά, μπορεί για κάθε αριθμητικό τελεστή να χρησιμοποιηθεί μόνο μια σωλήνωση;

Κάθε διεργασία χρειάζεται ένα μία σωλήνωση για να επικοινωνεί με το parent node, και άλλες 1 *των αριθμό των παιδιών της. Δηλαδή, άμα μία διεργασία που εκτελούταν σε node, που είχε 2 παιδιά, θα δημιουργούνταν άλλες 1 * 2 σωληνώσεις, άρα συνολικά 3.

Όχι, δε θα μπορούσε μια διεργασία να χρησιμοποιεί μόνο μία σωλήνωση για όλες τις διεργασίες παιδιά, γιατί κάθε παιδί χρειάζεται τη δικιά του σωλήνωση για να κάνει write προς τον γονέα, δηλαδή να του “στέλνει” δεδομένα.

Όχι, δε θα γινόταν κάθε αριθμητικός τελεστής να χρησιμοποιείμόνο μια σωλήνωση γιάτι παίρνει σαν ορίσματα 2 αριθμούς, δηλαδή τα παιδία του, οπότε χρείαζεται για κάθε παιδί μία σωλήνωση, έτσι ώστε κάθε παιδί να κάνει write().

2. Σε ένα σύστημα πολλαπλών επεξεργαστών, μπορούν να εκτελούνται παραπάνω από μια διεργασίες παράλληλα. Σε ένα τέτοιο σύστημα, τι πλεονέκτημα μπορεί να έχει η αποτίμηση της έκφρασης από δέντρο διεργασιών, έναντι της αποτίμησης από μία μόνο διεργασία;

Σε ένα σύστημα πολλαπλών επεξεργαστών, μας δίνεται η δυνατότητα να εκτελέσουμε πολλές διεργασίες παράλληλα(παράλληλος υπολογισμός). Με αυτή την δυνατότητα, αμά π.χ. στο παράδειγμά μας είχαμε να υπολογίσουμε μια μεγαλύτερη αριθμητική παράσταση με root node + και στην παράσταση είχαμε αριστερά και δεξία του + ένα περίπλοκο υπολογισμό, με το σύστημα πολλαπλών επεξεργαστών θα μπορούσαμε να υπολιγάζαμε ταυτόχρονα τον αριστερό αριθμό, αντίστοιχα και τον δεξίο και να γλυτώναμε χρόνο με αυτό τον τροπό.