Αναφορά Άσκησης 2: Διαχείριση Διεργασιών και Διαδιεργασιακή Επικοινωνία

Ομάδα 55

Αραβανής Τηλέμαχος
03119024
el19024@mail.ntua.gr

Καραφύλλης Νικόλαος
03119890
el19890@mail.ntua.gr

Άσκηση 1.1:

Ο πηγαίος κώδικας επισυνάπτεται (fork-given-tree.c).

Έξοδος εκτέλεσης του προγράμματος fork-given-tree:

```
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/oslab2test/Oslab_2h_erg$ ./fork-given-tree
A: Is being created...
A: Creating B...
A: Creating C...
B: Creating D...
A: waiting for C to terminate...
C: Sleeping...
B: waiting for D to terminate...
D: Sleeping...


A(13803)─┬─B(13804)──D(13806)
         └─C(13805)


C: Exiting...
My PID = 13803: Child PID = 13805 terminated normally, exit status = 17
A: waiting for B to terminate...
D: Exiting...
My PID = 13804: Child PID = 13806 terminated normally, exit status = 13
B: Exiting...
My PID = 13803: Child PID = 13804 terminated normally, exit status = 19
A: Exiting...
My PID = 13802: Child PID = 13803 terminated normally, exit status = 16
```

Απαντήσεις στις ερωτήσεις:

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

Το πρόγραμμα θα συνεχίσει να τρέχει ανενόχλητα αφού η αρχική διεργασία του προγράμματος θα «μπλοκάρει» στην εντολή *pid = wait(&status)* εφόσον δεν θα κάνει exit() κανένα παιδί της (το μοναδικό παιδί της Α «σκοτώθηκε» πρόωρα). Οι διεργασίες παιδιά της Α (Β και C) θα κληρονομηθούν από την init (PID=1), που κάνει συνεχώς wait().

```
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/Oslab_2h_erg$ ./fork-given-tree
A: Is being created...
A: Creating B...
A: Creating C...
B: Creating D...
A: waiting for C to terminate...
C: Sleeping...
B: waiting for D to terminate...
D: Sleeping...


A(10390)─┬─B(10391)──D(10393)
         └─C(10392)


My PID = 10389: Child PID = 10390 was terminated by a signal, signo = 15
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/Oslab_2h_erg$ C: Exiting...
D: Exiting...
My PID = 10391: Child PID = 10393 terminated normally, exit status = 13
B: Exiting...
^C
```

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

Καλώντας την show_pstree() με όρισμα το pid της αρχικής διεργασίας του προγράμματος στο δέντρο εμφανίζεται πέρα από την αρχική διεργασία του προγράμματος (fork-given-tree), όπως αναμένεται ως ρίζα του δέντρου, και οι διεργασίες παιδιά sh (shell) και pstree που που έχουν δημιουργηθεί ως βοηθητικές της show_pstree().

```
fork-given-tree(40876)─┬─A(40877)─┬─B(40878)──D(40880)
                       │          └─C(40879)
                       └─sh(40916)──pstree(40917)
```

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

Σε ένα σύστημα πολλαπλών χρηστών αν ένας χρήστης μπορούσε να δημιουργήσει δυσανάλογα μεγαλύτερο αριθμό διεργασιών σε σχέση με έναν άλλο ο υπολογιστικός χρόνος που αναλογεί σε κάθε διεργασία τα μειωνόταν σημαντικά και έτσι η αποκρισιμότητα των διεργασιών του χρήστη με τις λιγότερες διεργασίες θα ήταν πολύ μικρή σε σχέση με τις δυνατότητες που θα έπρεπε να του παρέχει το σύστημα. Έτσι το ανώτατο όριο στον αριθμό διεργασιών ανά χρήστη είναι κάποιου είδους ισοκατανομή των υπολογιστικών πόρων.

Άσκηση 1.2:

Ο πηγαίος κώδικας επισυνάπτεται (fork-arbitrary-tree.c).

Έξοδος εκτέλεσης του προγράμματος fork-arbitrary-tree με ενδεικτικό αρχείο εισόδου το testing.tree(επισυνάπτεται):

```
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/oslab2test/Oslab_2h_erg$ ./fork-arbitrary-tree testing.tree
A
        B
                E
                F
        C
        D
A: Is being created...
B: Is being created...
C: Is being created...
E: Is being created...
D: Is being created...
C: Sleeping...
F: Is being created...
E: Sleeping...
D: Sleeping...
F: Sleeping...


A(16266)─┬─B(16267)─┬─E(16269)
         │          └─F(16271)
         ├─C(16268)
         └─D(16270)


C: Exiting...
D: Exiting...
E: Exiting...
F: Exiting...
My PID = 16266: Child PID = 16268 terminated normally, exit status = 1
My PID = 16266: Child PID = 16270 terminated normally, exit status = 1
My PID = 16267: Child PID = 16269 terminated normally, exit status = 1
My PID = 16267: Child PID = 16271 terminated normally, exit status = 1
B: Exiting...
My PID = 16266: Child PID = 16267 terminated normally, exit status = 1
A: Exiting...
My PID = 16265: Child PID = 16266 terminated normally, exit status = 1
```

Απαντήσεις στις ερωτήσεις:

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

Η σειρά εμφάνισης των μηνυμάτων εξαρτάται από τον χρονοδρομολογητή. Από εκτέλεση σε εκτέλεση και από μηχάνημα σε μηχάνημα παραηρούμε ότι διαφέρει.

Άσκηση 1.3:

Ο πηγαίος κώδικας επισυνάπτεται (signals.c).

Έξοδος εκτέλεσης του προγράμματος signals με ενδεικτικό αρχείο εισόδου το testing.tree(επισυνάπτεται):

```
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/oslab2test/Oslab_2h_erg$ ./signals testing.tree
A
        B
                E
                F
        C
        D
PID = 16717, name A, starting...
PID = 16718, name B, starting...
PID = 16719, name C, starting...
My PID = 16717: Child PID = 16719 has been stopped by a signal, signo = 19
PID = 16720, name D, starting...
My PID = 16717: Child PID = 16720 has been stopped by a signal, signo = 19
PID = 16721, name E, starting...
My PID = 16718: Child PID = 16721 has been stopped by a signal, signo = 19
PID = 16722, name F, starting...
My PID = 16718: Child PID = 16722 has been stopped by a signal, signo = 19
My PID = 16717: Child PID = 16718 has been stopped by a signal, signo = 19
My PID = 16716: Child PID = 16717 has been stopped by a signal, signo = 19


A(16717)─┬─B(16718)─┬─E(16721)
         │          └─F(16722)
         ├─C(16719)
         └─D(16720)


PID = 16717, name = A is awake
PID = 16718, name = B is awake
PID = 16721, name = E is awake
E: Exiting...
My PID = 16718: Child PID = 16721 terminated normally, exit status = 0
PID = 16722, name = F is awake
F: Exiting...
My PID = 16718: Child PID = 16722 terminated normally, exit status = 0
B: Exiting...
My PID = 16717: Child PID = 16718 terminated normally, exit status = 0
PID = 16719, name = C is awake
C: Exiting...
My PID = 16717: Child PID = 16719 terminated normally, exit status = 0
PID = 16720, name = D is awake
D: Exiting...
My PID = 16717: Child PID = 16720 terminated normally, exit status = 0
A: Exiting...
My PID = 16716: Child PID = 16717 terminated normally, exit status = 0
```

Απαντήσεις στις ερωτήσεις:

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

Με τη χρήση sleep() για κάποια διεργασία στην ουσία κάνω μια εκτίμηση για το χρόνο που απαιτείται έως ότου οι υπόλοιπες διεργασίες εκτελέσουν κάποιο κομμάτι κώδικα (π.χ. να γίνει fork() του δέντρου διεργασιών) . Με τα σήματα πετυχαίνω με ακρίβεια τον συγχρονισμό των διεργασιών κατά την εκτέλεση του κώδικα. Μια διεργασία έχει σταματήσει και οι υπόλοιπες εκτελούν κάποιο κομμάτι κώδικα αλλά αυτή τη φορά επιλέγω εγώ πότε θα ξυπνήσω τη διεργασία που αναμένει..

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

Η χρήση της wait_for_ready_children() εξασφαλίζει ότι (αναδρομικά) έχουν «σταματήσει» (raise(SIGSTOP)) όλες οι διεργασίες του δέντρου διεργασιών και επιπλέον κατ΄ επέκταση ότι έχει δημιουργηθεί το δέντρο διεργασιών. Η παράβλεψή θα απέτρεπε τον συγχρονισμό αφού η διεργασία πατέρας θα έστελνε σήμα στα παιδιά της για να συνεχίσουν την εκτέλεσή τους(SIGCONT) χωρίς να έχει λάβει αναγνωριστικό για την παύση τους (SIGSTOP). Αυτό θα επέφερε πρώτον λανθασμένη εκτύπωση του δέντρου (showpstree()), και πιθανον παύση μιας διεργασίας για πάντα

(!) (πρακτικά παρατηρώ ότι αυτό δεν συμβαίνει αφού κάποια στιγμή το λειτουργικό σύστημα “ξυπνά” τις διεργασίες που αναμένουν).

Άσκηση 1.4:

Ο πηγαίος κώδικας επισυνάπτεται (pipes.c) .

Έξοδος εκτέλεσης του προγράμματος pipes με ενδεικτικό αρχείο εισόδου το expr.tree(επισυνάπτεται):

```
(base) nickarafyllis@nickarafyllis-Inspiron-5570:~/oslab/oslab2test/Oslab_2h_erg$ ./pipes expr.tree
+
        10
        *
                +
                        5
                        7
                4
Root: Creating pipe...
+: Is being created...
PID = 17067, name +, starting...
+: Creating pipe...
PID = 17068, name 10, starting...
10: write value to pipe
10: Wrote to pipe and exiting...
PID = 17069, name *, starting...
*: Creating pipe...
Parent: received value 10 from the pipe.
My PID = 17067: Child PID = 17068 terminated normally, exit status = 7
PID = 17070, name +, starting...
+: Creating pipe...
PID = 17071, name 4, starting...
4: write value to pipe
4: Wrote to pipe and exiting...
Parent: received value 4 from the pipe.
PID = 17072, name 5, starting...
5: write value to pipe
5: Wrote to pipe and exiting...
Parent: received value 5 from the pipe.
My PID = 17069: Child PID = 17071 terminated normally, exit status = 7
PID = 17073, name 7, starting...
7: write value to pipe
7: Wrote to pipe and exiting...
My PID = 17070: Child PID = 17072 terminated normally, exit status = 7
Parent: received value 7 from the pipe.
My PID = 17070: Child PID = 17073 terminated normally, exit status = 7
+: Wrote to pipe and exiting...
Parent: received value 12 from the pipe.
My PID = 17069: Child PID = 17070 terminated normally, exit status = 7
*: Wrote to pipe and exiting...
Parent: received value 48 from the pipe.
My PID = 17067: Child PID = 17069 terminated normally, exit status = 7
+: Wrote to pipe and exiting...
Root: received value 58 from the pipe.
My PID = 17066: Child PID = 17067 terminated normally, exit status = 7
Result 58
```

Απαντήσεις στις ερωτήσεις:

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;

Χρειάζεται μία μόνο σωλήνωση ανά δύο διεργασίες παιδιά (κάθε διεργασία για τα παιδιά της εκτός της αρχικής διεργασίας που έχει μόνο ένα παιδί), αφού οι αριθμητικοί τελεστές “+” και “*” είναι αντιμεταθετικοί και άρα δεν έχει σημασία η σειρά με την οποία θα γράψουν οι διεργασίες παιδιά και κατ’ επέκταση θα διαβάσουν οι διεργασίες γονείς στα άκρα της μίας σωλήνωσης. Σε αριθμητικούς τελεστές που δεν ισχύει η αντιμεταθετική ιδιότητα (όπως το “-” και το “/”) θα χρειαζόμασταν παραπάνω σωληνώσεις (μία ανά παιδί – τελεστέο).

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;

Με την χρήση δέντρου διεργασιών σε ένα σύστημα πολλαπλών επεξεργαστών, μπορούμε να αποτιμήσουμε την έκφραση σε λιγότερο χρόνο, έναντι της αποτίμησης από μία μόνο διεργασία αφού οι διεργασίες που εκτελούν τις επιμέρους πράξεις εκτελούνται παράλληλα στους ν πυρήνες (πολυπλοκότητα O(N/ν)).