

Λειτουργικά Συστήματα

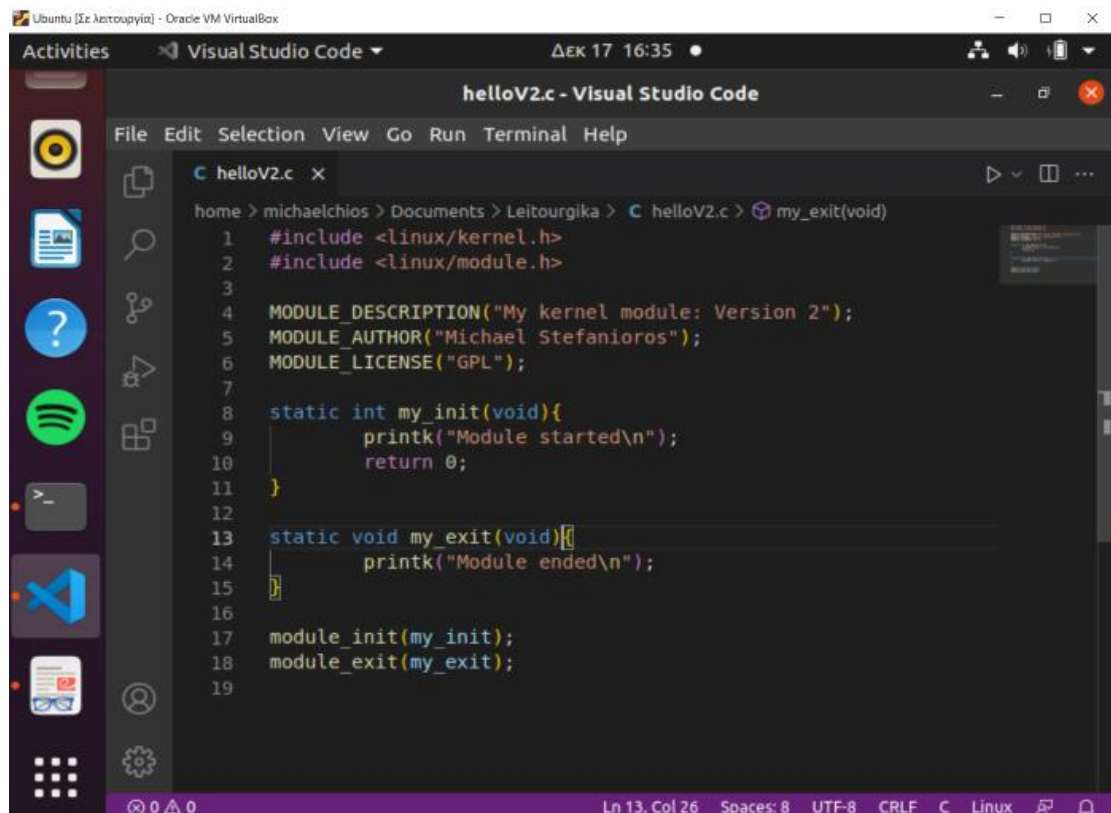
Δραστηριότητα 1^η : Εργασία Ανάπτυξης Αρθρώματος Πυρήνα (Kernel Module)

Ονοματεπώνυμο: Στεφανιώρος Μιχαήλ

Έτος: 4^ο

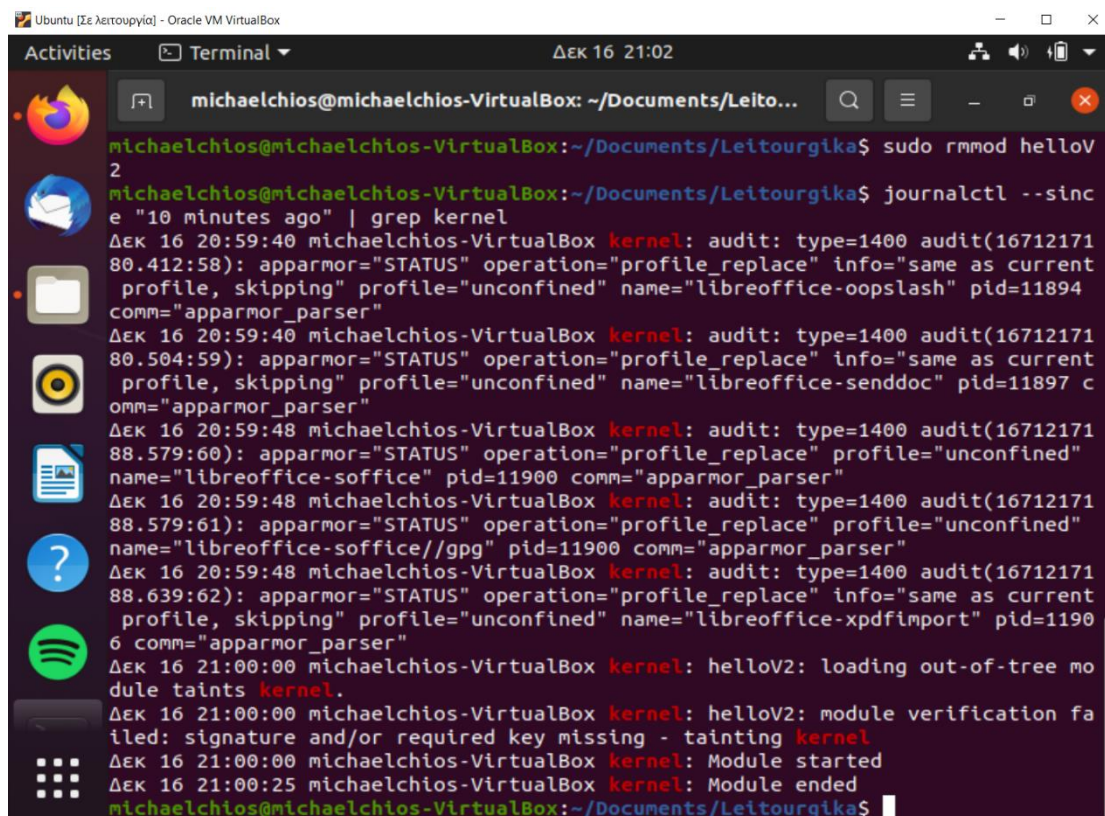
Άσκηση 1^η

Κώδικας:



```
helloV2.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
C helloV2.c x
home > michaelchios > Documents > Leitourgika > C helloV2.c > my_exit(void)
1  #include <linux/kernel.h>
2  #include <linux/module.h>
3
4  MODULE_DESCRIPTION("My kernel module: Version 2");
5  MODULE_AUTHOR("Michael Stefanioros");
6  MODULE_LICENSE("GPL");
7
8  static int my_init(void){
9      printk("Module started\n");
10     return 0;
11 }
12
13 static void my_exit(void){
14     printk("Module ended\n");
15 }
16
17 module_init(my_init);
18 module_exit(my_exit);
19
```

Αρχικά, κάνω include τα header files του linux: kernel.h, module.h. Έπειτα, με τις μακροεντολές MODULE_DESCRIPTION, MODULE_AUTHOR, MODULE_LICENSE ορίζω κάποια metadata. Στην ουσία, προσθέτω κάποιες έξτρα πληροφορίες στο module. Στην συνέχεια, υλοποιώ τις συναρτήσεις my_init και my_exit. Όταν φορτώνω το module με την εντολή “sudo insmod (...)ko” εκτελείται η my_init, ενώ όταν το αφαιρέσω με την εντολή “sudo rmmod (...)” θα εκτελεστεί η my_exit. Τέλος, οι μακροεντολές module_init(), module_exit() ορίζουν τις συναρτήσεις που έχουν σαν όρισμα, ως συναρτήσεις εισόδου και εξόδου αντίστοιχα.



```

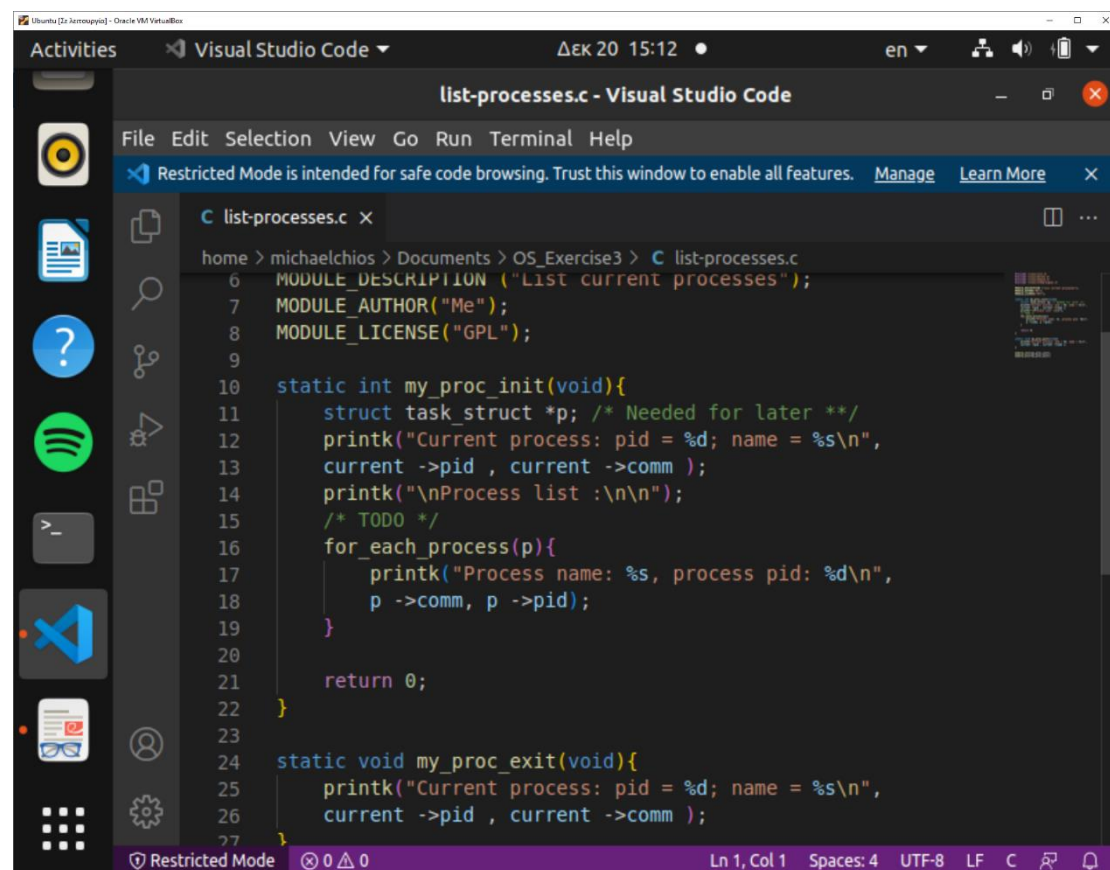
Ubuntu [Σε λειτουργία] - Oracle VM VirtualBox
Activities Terminal Δεκ 16 21:02
michaelchios@michaelchios-VirtualBox: ~/Documents/Leitourgika$ sudo rmmod helloV2
michaelchios@michaelchios-VirtualBox: ~/Documents/Leitourgika$ journalctl --since "10 minutes ago" | grep kernel
Δεκ 16 20:59:40 michaelchios-VirtualBox kernel: audit: type=1400 audit(1671217180.412:58): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="libreoffice-oopslash" pid=11894 comm="apparmor_parser"
Δεκ 16 20:59:40 michaelchios-VirtualBox kernel: audit: type=1400 audit(1671217180.504:59): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="libreoffice-senddoc" pid=11897 comm="apparmor_parser"
Δεκ 16 20:59:48 michaelchios-VirtualBox kernel: audit: type=1400 audit(1671217188.579:60): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="libreoffice-soffice" pid=11900 comm="apparmor_parser"
Δεκ 16 20:59:48 michaelchios-VirtualBox kernel: audit: type=1400 audit(1671217188.579:61): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="libreoffice-soffice//gpg" pid=11900 comm="apparmor_parser"
Δεκ 16 20:59:48 michaelchios-VirtualBox kernel: audit: type=1400 audit(1671217188.639:62): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="libreoffice-xpdfimport" pid=11906 comm="apparmor_parser"
Δεκ 16 21:00:00 michaelchios-VirtualBox kernel: helloV2: loading out-of-tree module taints kernel.
Δεκ 16 21:00:00 michaelchios-VirtualBox kernel: helloV2: module verification failed: signature and/or required key missing - tainting kernel
Δεκ 16 21:00:00 michaelchios-VirtualBox kernel: Module started
Δεκ 16 21:00:25 michaelchios-VirtualBox kernel: Module ended
michaelchios@michaelchios-VirtualBox: ~/Documents/Leitourgika$
```

Άσκηση 2^η

Η task_struct είναι μια, σχετικά, μεγάλη δομή δεδομένων στην οποία αποθηκεύονται όλες οι πληροφορίες μιας διεργασίας, τις οποίες χρειάζεται το λειτουργικό για να μπορεί να την διαχειριστεί. Μερικές πληροφορίες είναι τα δεδομένα ενός εκτελέσιμου αρχείου, ο χώρος στη μνήμη που καταλαμβάνει η διεργασία, η κατάστασή της, δηλαδή αν

είναι σε αναμονή ή αν εκτελείται κλπ και πολλά ακόμα. Δεδομένου του όγκου πληροφοριών που αποθηκεύονται για κάθε διεργασία, η `task_struct` θεωρείται μικρή σε μέγεθος. Η δομή αυτή είναι ουσιαστικά μια λίστα (task list) η οποία περιέχει σε κάθε θέση έναν process descriptor, στον οποίο υπάρχουν οι πληροφορίες που αφορούν μια διεργασία. Τέλος, γίνεται φανερό ότι η `task_struct` περιλαμβάνει το context κάθε διεργασίας και χρησιμοποιείται ένα pid για να ξεχωρίζει κάθε διεργασία.

Άσκηση 3^η



```
list-processes.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More X
C list-processes.c X
home > michaelchios > Documents > OS_Exercise3 > C list-processes.c
6 MODULE_DESCRIPTION ("List current processes");
7 MODULE_AUTHOR("Me");
8 MODULE_LICENSE("GPL");
9
10 static int my_proc_init(void){
11     struct task_struct *p; /* Needed for later */
12     printk("Current process: pid = %d; name = %s\n",
13         current ->pid , current ->comm );
14     printk("\nProcess list :\n\n");
15     /* TODO */
16     for_each_process(p){
17         printk("Process name: %s, process pid: %d\n",
18             p ->comm, p ->pid);
19     }
20
21     return 0;
22 }
23
24 static void my_proc_exit(void){
25     printk("Current process: pid = %d; name = %s\n",
26         current ->pid , current ->comm );
27 }
```

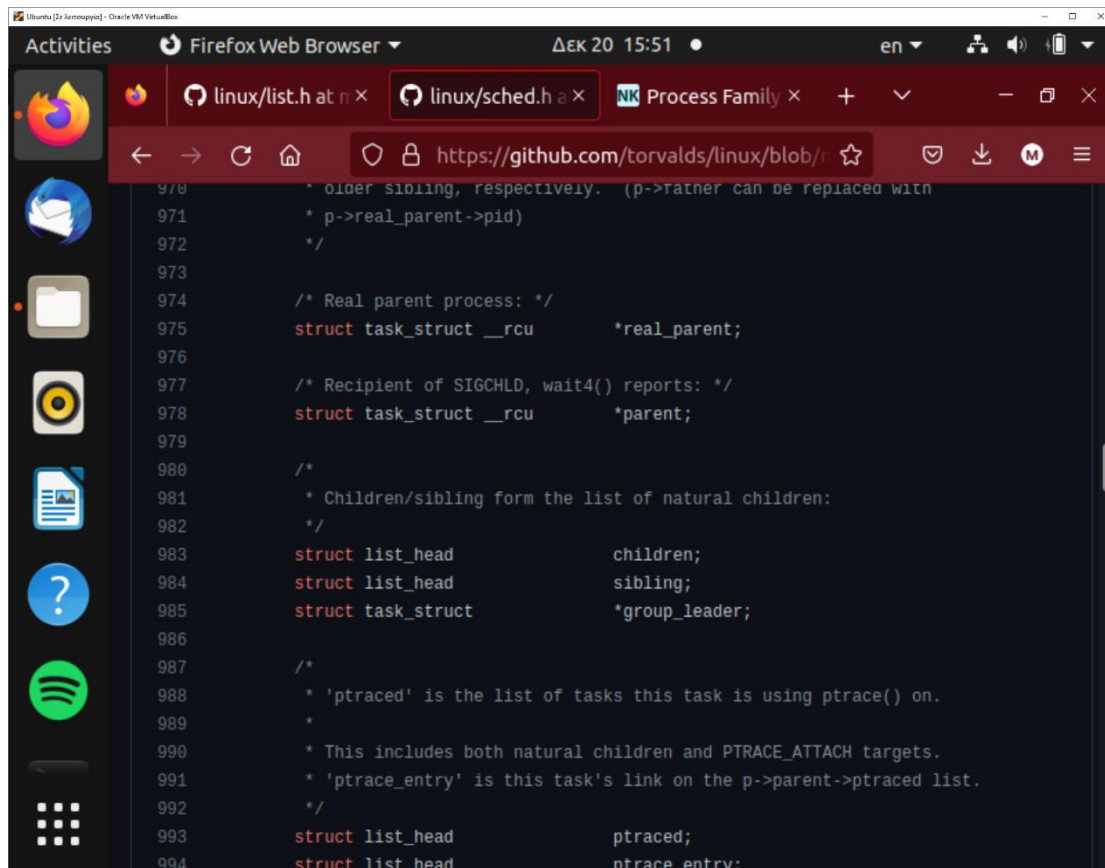
Στο σημείο TODO, για να τυπώσω τις πληροφορίες όλων των διεργασιών του συστήματος, χρησιμοποιώ την μακροεντολή

`for_each_process(p)` η οποία κάνει προσπέλαση σε όλο το `task_struct` και επιστρέφει τις πληροφορίες κάθε διεργασίας του συστήματος.

```
Ubuntu [2x kernel.org] - Oracle VM VirtualBox
Activities Terminal ΔΕΚ 20 15:14 en
michaelchios@michaelchios-VirtualBox: ~/Documents/OS_E...
michaelchios@michaelchios-VirtualBox:~/Documents/OS_Exercise3$ sudo insmod list-
processes.ko
[sudo] password for michaelchios:
michaelchios@michaelchios-VirtualBox:~/Documents/OS_Exercise3$ sudo rmmod list-
processes
michaelchios@michaelchios-VirtualBox:~/Documents/OS_Exercise3$ journalctl --sin
ce "10 minutes ago" | grep kernel
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: list_processes: loading out-of-
tree module taints kernel.
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: list_processes: module verifica
tion failed: signature and/or required key missing - tainting kernel
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Current process: pid = 5693; na
me = insmod
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel:
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: systemd, process
pid: 1
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: kthreadd, process
pid: 2
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: rcu_gp, process p
id: 3
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: rcu_par_gp, proce
ss pid: 4
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: netns, process pi
d: 5
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: kworker/0:0H, pro
cess pid: 7
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: kworker/0:1H, pro
cess pid: 9
ΔΕΚ 20 15:13:27 michaelchios-VirtualBox kernel: Process name: mm_percpu_wq, pro
```

Άσκηση 4^η

Ερώτηση 1:



```
970 * older sibling, respectively. (p->rather can be replaced with
971 * p->real_parent->pid)
972 */
973
974 /* Real parent process: */
975 struct task_struct __rcu    *real_parent;
976
977 /* Recipient of SIGCHLD, wait4() reports: */
978 struct task_struct __rcu    *parent;
979
980 /*
981 * Children/sibling form the list of natural children:
982 */
983 struct list_head            children;
984 struct list_head            sibling;
985 struct task_struct          *group_leader;
986
987 /*
988 * 'ptraced' is the list of tasks this task is using ptrace() on.
989 *
990 * This includes both natural children and PTRACE_ATTACH targets.
991 * 'ptrace_entry' is this task's link on the p->parent->ptraced list.
992 */
993 struct list_head            ptraced;
994 struct list_head            ptrace_entry;
```

Το πεδίο children είναι ένας δείκτης στην διεργασία παιδί της γονικής διεργασίας. Επίσης, όλες οι αδερφικές διεργασίες (sibling tasks) έχουν το ίδιο parent task και συνδέονται μεταξύ τους με ένα linked list. Σε αυτή τη λίστα, το task->children λειτουργεί ως list head και ανανεώνεται σε κάθε κύκλο της μακροεντολής list_for_each_entry.

Ερώτηση 2:

```
450
451 /**
452  * list_for_each_entry - iterate over list of given type
453  * @pos:      the type * to use as a loop cursor.
454  * @head:     the head for your list.
455  * @member:   the name of the list_head within the struct.
456  */
457 #define list_for_each_entry(pos, head, member) \
458     for (pos = list_first_entry(head, typeof(*pos), member); \
459          &pos->member != (head); \
460          pos = list_next_entry(pos, member))
461
```

Η μακροεντολή `list_for_each_entry` δέχεται ως ορίσματα έναν τύπο μεταβλητής ο οποίος μετράει τις επαναλήψεις, το head list και το όνομα της λίστας. Στην ουσία, κάνει προσπελάσεις στα παιδιά μιας διεργασίας.

Ερώτηση 3:

Έχοντας εξηγήσει στα προηγούμενα ερωτήματα την σημασία των πεδίων `children` και `sibling`, όπως και την λειτουργία της μακροεντολής `list_for_each_entry`, κάνω χρήση αυτής ώστε να τυπώσω τις πληροφορίες του parent task και των children. Έπειτα, ακολουθώντας τις οδηγίες της εκφώνησης, εκτελώ το αρχείο `forking.c` και μόλις τυπώσει το `pid` της parent, το δίνω ως τιμή στο `list-children.c` ώστε να μου επιστρέψει τις πληροφορίες των παιδιών της.

```

PID: 11526
Forked! PID: 11609
Forked! PID: 0
Forked! PID: 11932
Forked! PID: 11933
Forked! PID: 0
Forked! PID: 0
Forked! PID: 11938
Forked! PID: 0
Forked! PID: 11939
Forked! PID: 11940
Forked! PID: 11941
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 11945
Forked! PID: 11946
Forked! PID: 11947
Forked! PID: 11948
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 11949
Forked! PID: 0
Forked! PID: 11950
Forked! PID: 11951
Forked! PID: 0
Forked! PID: 0

```

```

michaelchios@michaelchios-VirtualBox: ~/Documents/list-c...
michaelchios@michaelchios-VirtualBox:~/Documents/list-children-module$ sudo ins
mod list-children.ko
[sudo] password for michaelchios:
michaelchios@michaelchios-VirtualBox:~/Documents/list-children-module$ sudo rmm
od list-children
michaelchios@michaelchios-VirtualBox:~/Documents/list-children-module$ journalc
tl --since "10 minutes ago" | grep kernel
ΔΕΚ 20 16:08:06 michaelchios-VirtualBox kernel: pid: 11526, name: forking
ΔΕΚ 20 16:08:06 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1609
ΔΕΚ 20 16:08:07 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1932
ΔΕΚ 20 16:08:07 michaelchios-VirtualBox kernel: pid: 11526, name: forking
ΔΕΚ 20 16:08:07 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1609
ΔΕΚ 20 16:08:07 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1932
ΔΕΚ 20 16:08:08 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1938
ΔΕΚ 20 16:08:08 michaelchios-VirtualBox kernel: pid: 11526, name: forking
ΔΕΚ 20 16:08:08 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1609
ΔΕΚ 20 16:08:08 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1932
ΔΕΚ 20 16:08:09 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1938
ΔΕΚ 20 16:08:09 michaelchios-VirtualBox kernel: pid: 11526, name: forking
ΔΕΚ 20 16:08:09 michaelchios-VirtualBox kernel: parent pid: 11526, child pid: 1
1609

```