

Λειτουργικά Συστήματα

Ονοματεπώνυμο: Ιωάννα Γέμου
ΑΜ: 1070525
Ακαδημαϊκό έτος: 2022-2023

Περιεχόμενα

1	Σκοπός της εργασίας	2
2	Άσκηση 1	2
3	Άσκηση 2	2
4	Άσκηση 3	3
5	Άσκηση 4	4

1. Σκοπός της εργασίας

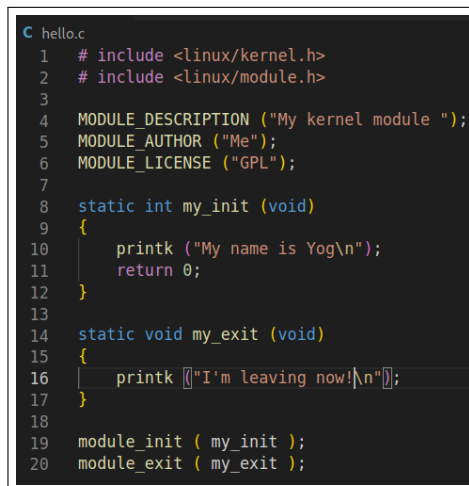
Σκοπός της συγκεκριμένης εργασίας είναι να κατασκευάσουμε απλά αρθρώματα πυρήνα (**kernel modules**).

Ο πυρήνας του **kernel** είναι μονολιθικός, οπότε αν θέλαμε να προσθέσουμε κάποια επιπλέον λειτουργικότητα θα έπρεπε να επεξεργαστούμε τον πηγαίο κώδικα, να τον επαναμεταγλώττισουμε και στην συνέχεια να επανεκκινήσουμε το σύστημα μας για να δούμε τις αλλαγές. Τα **modules** απλοποιούν αυτήν την διαδικασία, αφού φορτώνονται στον ήδη υπάρχον πυρήνα κατά την διάρκεια της εκτέλεσής του, διευκολύνοντάς μας έτσι από τα κουραστικά παραπάνω βήματα.

2. Άσκηση 1

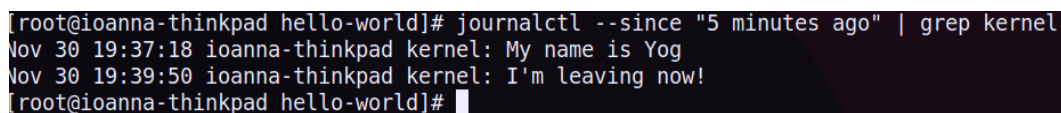
Στην πρώτη άσκηση μας ζητείται να φορτώσουμε στον πυρήνα ένα **module**. Έπειτα έχουμε την δυνατότητα να το εκφορτώσουμε. Για να το κάνουμε αυτό ακολουθήσαμε τα βήματα που περιγράφονται στην εκφώνηση.

Το συγκεκριμένο **module** που μας δίνεται τυπώνει στα **logs** του συστήματος "Hi" και "Bye".



```
C hello.c
1  #include <linux/kernel.h>
2  #include <linux/module.h>
3
4  MODULE_DESCRIPTION("My kernel module ");
5  MODULE_AUTHOR("Me");
6  MODULE_LICENSE("GPL");
7
8  static int my_init (void)
9  {
10     printk ("My name is Yog\n");
11     return 0;
12 }
13
14 static void my_exit (void)
15 {
16     printk ("I'm leaving now!\n");
17 }
18
19 module_init ( my_init );
20 module_exit ( my_exit );
```

Εμείς αλλάζοντας τα μηνύματα μέσα στην **printk**, μπορούμε να δούμε διαφορετικά αποτελέσματα στα **logs** του συστήματος!



```
[root@ioanna-thinkpad hello-world]# journalctl --since "5 minutes ago" | grep kernel
Nov 30 19:37:18 ioanna-thinkpad kernel: My name is Yog
Nov 30 19:39:50 ioanna-thinkpad kernel: I'm leaving now!
[root@ioanna-thinkpad hello-world]#
```

Σχήμα 1: Το νέο μήνυμα που τυπώνεται

3. Άσκηση 2

Σε αυτήν την άσκηση μας ζητείται να μελετήσουμε την δομή **taskstruct**.

Πρόκειται για μία δομή δεδομένων, της οποίας σκοπός είναι να αποθηκεύει σημαντικές πληροφορίες που το **kernel** χρειάζεται να γνωρίζει για τις διεργασίες που είναι φορτωμένες σε αυτό.

Η συγκεκριμένη δομή είναι υπεύθυνη για τα εξής:

- κατάσταση κάθε διεργασίας που είναι φορτωμένη στον πυρήνα

- καθορίζει ποια διεργασία θα τρέξει
- αποθήκευση περιεχομένων του επεξεργαστή για κάθε διεργασία
- αποθήκευση πληροφοριών για τις διεργασίες-παιδιά

4. Άσκηση 3

Σε αυτήν την άσκηση μας ζητείται να γράψουμε ένα **module** το οποίο θα τυπώνει πληροφορίες για όλες τις διεργασίες που είναι ενεργές την στιγμή που αυτό φορτώνεται στον πυρήνα.

Για να βρούμε το **ID** κάθε διεργασίας, χρησιμοποιούμε το **macro current**.

Για να τυπώσουμε όλες τις διεργασίες που είναι φορτωμένες στον πυρήνα χρησιμοποιούμε το **macro for_each_process**:

```
#define for_each_process(p) \
    for (p = &init_task ; (p = next_task(p)) != &init_task ; )
```

Σχήμα 2: Το macro for_each_process

Κάνοντας την εξής αλλαγή στον κώδικα παίρνουμε την λίστα με τα **processes** που τρέχουν όταν εμείς φορτώνουμε το **module** μας:

```
static int my_proc_init(void)
{
    struct task_struct *p; /* Needed for later */
    printk("Current process: pid = %d; name = %s\n",
           current->pid, current->comm);
    printk("\nProcess list :\n\n");

    for_each_process(p){
        printk("Current process: pid = %d; name = %s\n",
               p->pid, p->comm);
    }

    return 0;
}
```

Σχήμα 3: Η συνάρτηση my_proc_init

```

Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 1632; name = skypeforlinux
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 1646; name = firefox
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 1730; name = Socket Process
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 1754; name = Privileged Cont
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 1797; name = WebExtensions
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2028; name = nautilus
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2031; name = gvfsd-trash
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2049; name = dconf-service
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2062; name = gvfsd-recent
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2077; name = gvfsd-metadata
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2099; name = alacrity
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2112; name = bash
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2146; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2149; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2150; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2152; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2165; name = chrome_crashpad
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2180; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2194; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2214; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2236; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2257; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2457; name = RDD Process
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2458; name = Utility Process
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2671; name = Isolated Web Co
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2721; name = Isolated Web Co
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 2803; name = Isolated Web Co
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 3337; name = kworker/3:2
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 3756; name = kworker/1:3
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 4039; name = kworker/3:0
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 4661; name = kworker/2:0
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 4922; name = kworker/u9:1
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5000; name = kworker/0:1
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5138; name = Web Content
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5195; name = gvfsd-network
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5213; name = gvfsd-dnssd
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5233; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5247; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5289; name = cpptools
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5320; name = code
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5383; name = cpptools-srv
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5579; name = kworker/u8:1
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5612; name = kworker/1:0
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 5692; name = Web Content
Dec 02 12:53:40 ioanna-thinkpad kernel: Current process: pid = 6545; name = kworker/u8:3

```

Σχήμα 4: Η λίστα με τις διεργασίες

5. Άσκηση 4

Μέσα στην δομή `task_struct` είναι ορισμένα τα πεδία `children` και `sibling`. Το `children` είναι μία λίστα με όλα τα παιδιά της διεργασίας γονέα. Το

Ψάχνοντας μέσα στο αρχείο `linux/tools/include/linux/list.h` βλέπουμε πως ορίζεται το `macro list_for_each_entry`. Με αυτό το `macro` μπορούμε να κάνουμε εύκολα `iterate` μέσα σε μια λίστα. Στην δική μας περίπτωση η λίστα που στην οποία κάνουμε `iterate` είναι η `children`.

```

#define list_for_each_entry(pos, head, member) \
    for (pos = list_first_entry(head, typeof(*pos), member); \
         !list_entry_is_head(pos, head, member); \
         pos = list_next_entry(pos, member))

```

Σχήμα 5: Το `macro list_for_each_entry`

Στην συγκεκριμένη περίπτωση μας ζητείται να φτιάξουμε ένα `kernel module`, το οποίο θα τυπώνει πληροφορίες για μία διεργασία και τις θυγατρικές τις διεργασίες.

Για να το υλοποιήσουμε αυτό αρχικά κάνουμε `compile` το αρχείο `forking.c` και έπειτα το εκτελούμε. Το πρόγραμμα αυτό φτιάχνει μία διεργασία και στην συνέχεια με την εντολή

`fork()` δημιουργεί τα παιδιά της. Επαναληπτικά, οι διεργασίες- παιδιά που δημιουργήθηκαν, γεννάνε κι αυτά τα δικά τους παιδιά.

Πρέπει ωστόσο να φροντίσουμε να ορίσουμε ένα αρκετά μεγάλο `SLEEP_TIME` ενδιάμεσα της δημιουργίας των διεργασιών-παιδιά, έτσι ώστε να έχουμε χρόνο να πάρουμε το `ID` της διεργασίας, να το ορίσουμε στο αρχείο `list-children.c`, έπειτα να κάνουμε `compile` το αρχείο αυτό και να το φορτώσουμε στο `kernel`.

```
[ioanna@ioanna-thinkpad list-children-module]$ gcc forking.c -o forking
[ioanna@ioanna-thinkpad list-children-module]$ ./forking
PID: 15810
Forked! PID: 16251
Forked! PID: 0
Forked! PID: 16370
Forked! PID: 16369
Forked! PID: 0
Forked! PID: 0
Forked! PID: 16415
Forked! PID: 16416
Forked! PID: 16417
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 16418
Forked! PID: 0
Forked! PID: 16486
Forked! PID: 16488
Forked! PID: 16489
Forked! PID: 16490
Forked! PID: 16487
Forked! PID: 16491
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 16492
Forked! PID: 0
Forked! PID: 0
Forked! PID: 0
Forked! PID: 16493
Forked! PID: 0
Forked! PID: 0
```

Σχήμα 6: Το πρόγραμμα `forking.c` δημιουργεί τις διεργασίες παιδιά

Το αρχείο `list-children.c` με την σειρά του, ψάχνει στην λίστα με τις διεργασίες που τρέχουν και ελέγχει αν υπάρχει κάπου διεργασία με το `ID` που ορίσαμε. Αφού την βρεί, κάνει `iterate` πάνω στις διεργασίες-παιδιά της αρχικής μας διεργασίας, τυπώνοντάς μας το `ID` τους.

Ο τροποποιημένος κώδικας που υλοποιεί τα παραπάνω είναι ο εξής:

```
#define PID 15810

static void print_process_info(struct timer_list *unused)
{
    struct task_struct* task;
    struct task_struct* child;

    /* Synchronization mechanism needed before searching for the process */
    rcu_read_lock();

    /* Search through the global namespace for the process with the given PID */
    task = pid_task(find_pid_ns(PID, &init_pid_ns), PIDTYPE_PID);

    if (task)
    {
        printk("pid: %d, name: %s\n", task->pid, task->comm);

        /* TODO: iterate over the process' children and print their PID */
        list_for_each_entry(child, &task->children, sibling ){
            printk("pid: %d", child->pid);
        }
    }

    rcu_read_unlock(); /* Task pointer is now invalid! */

    /* Restart the timer. */
    check_timer.expires = jiffies + DELAY;
    add_timer(&check_timer);
}
```

Σχήμα 7: Χρήση του macro list_for_each_entry

Στην συνέχεια, μπορούμε να δούμε το PID κάθε διεργασίας :

```
Dec 02 15:35:42 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:43 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:43 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:44 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:44 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:45 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:45 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:46 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:46 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:47 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:47 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:48 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:48 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:49 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:49 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:50 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:50 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:51 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:51 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:52 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:52 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:53 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:53 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:54 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:54 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:55 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:55 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:55 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:56 ioanna-thinkpad kernel: pid: 16369
Dec 02 15:35:56 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:56 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:57 ioanna-thinkpad kernel: pid: 16369
Dec 02 15:35:57 ioanna-thinkpad kernel: pid: 15810, name: forking
Dec 02 15:35:57 ioanna-thinkpad kernel: pid: 16251
Dec 02 15:35:58 ioanna-thinkpad kernel: pid: 16369
Dec 02 15:35:58 ioanna-thinkpad kernel: pid: 15810, name: forking
```

Σχήμα 8: