Πανεπιστήμιο Πατρών
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Λειτουργικά Συστήματα
Δραστηριότητα 2

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

Περιεχόμενα

1. Σκοπός της εργασίας

Στην συγκεκριμένη δραστηριότητα μας ζητείται να ασχοληθούμε με βασικούς μηχανισμούς δέσμευσης μνήμης και συγχρονισμού στο Linux Kernel.
Σε περιβάλλοντα kernel έχουμε στη διάθεσή μας διαφορετικά είδη μνήμης, όπως την φυσική, την εικονική από τον χώρο διευθύνσεων του πυρήνα ή/και μιας διεργασίας και την resident μνήμη.
Τα δεδομένα ενός module βρίσκονται πάντα σε resident μνήμη.
Όταν χειριζόμαστε resident μνήμη, δηλαδή μνήμη που πράγματι υπάρχει στη φυσική μνήμη του συστήματός μας, μπορούμε απλά να έχουμε πρόσβαση σε οποιαδήποτε θέση της. Ωστόσο, non-resident μνήμη μπορεί να εκτελεστεί μόνο σε πλαίσιο διεργασίας.

2. Άσκηση 1

Στην συγκεκριμένη άσκηση μας ζήτείται να δώσουμε μία μικρή εξήγηση για την λειτουργία ορισμένων συμβόλων.

2.1. **kmalloc**: πρόκειται για μία συνάρτηση, η οποία χρησιμοποιείται για να δεσμεύσουμε resident μνήμη μέσα από τον πυρήνα. Παίρνει σαν όρισμα τον αριμό των bytes που θέλουμε να δεσμεύσουμε και τρόπο με τον οποίο θέλουμε να γίνει αυτή η δέσμευση, όπως επισημαίνεται και στην εκφώνηση. Επιστρέφει void pointer της διεύθυνσης της πρώτης δεσμευμένης θέσης. Αν δεν υπάρχει αρκετή μνήμη για να δεσμεύσουμε τότε επιστρέφει null pointer.

```
void * kmalloc ( size_t size,
                 gfp_t flags);
```

Σχήμα 1: kmalloc

2.2. **kfree**: ελευθερώνει την προηγουμένως δεσμευμένη μνήμη

2.3. **get_free_pages**: σε περιπτώσεις που το module μας χρειάζεται μεγάλα ποσά μνήμης, είναι προτιμότερο να χρησιμοποιηθεί page-oriented τεχνική για την δέσμευση της μνήμης. Για να δεσμεύσουμε μνήμη σελίδων, χρησιμοποιούμε την συνάρτηση get_free_pages, η οποία μας επιστρέφει έναν pointer για την νέα σελίδα.

2.4. **atomic_t**: για τον συγχρονισμό της πρόσβασης σε μία μεταβλητή, το Linux kernel προσφέρει ατομικές ματαβλητές μέσω του τύπου atomic_t ο οποίος κρατάει μια ακέραια τιμή. Προσφέρει, δηλαδή, την δυνατότητα σε κάποιες λειτουργίες να διαβάσουν και να γράψουν σε αυτήν την ατομική μεταβλητή ταυτόχρονα Η χρήση των ατομικών μεταβλητών μπορεί να γίνει για την αποκλειστική πρόσβαση σε έναν πόρο του συστήματος όπως μία συσκευή.

2.5. **atomic_read**: η συνάρτηση αυτή δέχεται ως όρισμα έναν pointer τύπου atomic_t και διαβάζει την τιμή του.

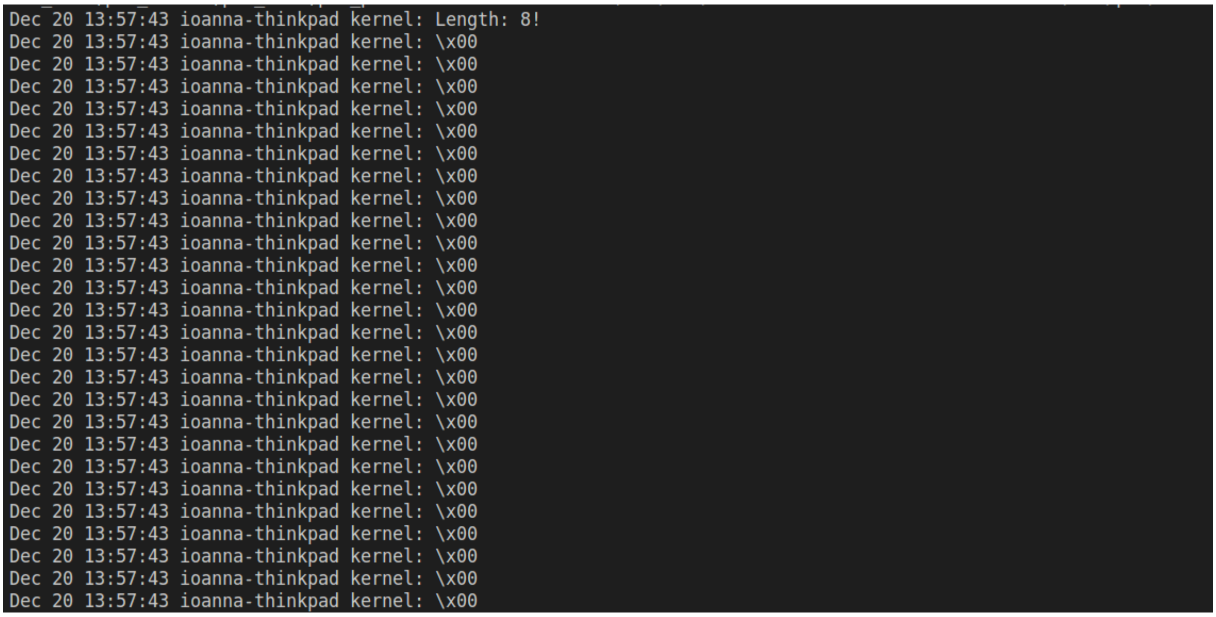
3. Άσκηση 2

Σε αυτήν την άσκηση μας ζητείται να γράψουμε ένα module το οποίο το οποίο όταν φορτωθεί θα δεσμεύει μνήμη μεγέθους 4096 βψτες και στη συνέχεια θα τυπώνει τα περιεχόμενά της. Ο κώδικας του module είναι ο εξής:

```c
MODULE_DESCRIPTION("Memory allocation");
MODULE_AUTHOR("Me");
MODULE_LICENSE("GPL");

char *my_memory ;


static int my_init(void)
{

        my_memory = kmalloc (4096 * sizeof (char), GFP_KERNEL );
        if (!my_memory) printk("Error in allocating memory!");

        int length = sizeof(my_memory)/sizeof(my_memory[0]);

        printk("Length: %d!\n", length);

        for (int i=0; i<50; i++){
            printk("%c ", my_memory[i]);
        }
        return 0;
}

static void my_exit(void)
{
        kfree(my_memory);
        printk("Memory freed\n");
}

module_init(my_init);
module_exit(my_exit);
```

Σχήμα 2: Παράδειγμα χρήσης της kmalloc

Στην συνάρτηση my_init() δεσμεύουμε την ζητούμενη μνήμη κατά την φόρτωση του module, χειροζόμενοι κατάλληλα την περίπτωση αδυναμίας δέσμευσης.
Στην συνέχεια, με την my_exit() ελευθερώνουμε την μνήμη που δεσμεύσαμε, όταν θα εκφορτώσουμε το module .



Σχήμα 3: Τα logs του συστήματος

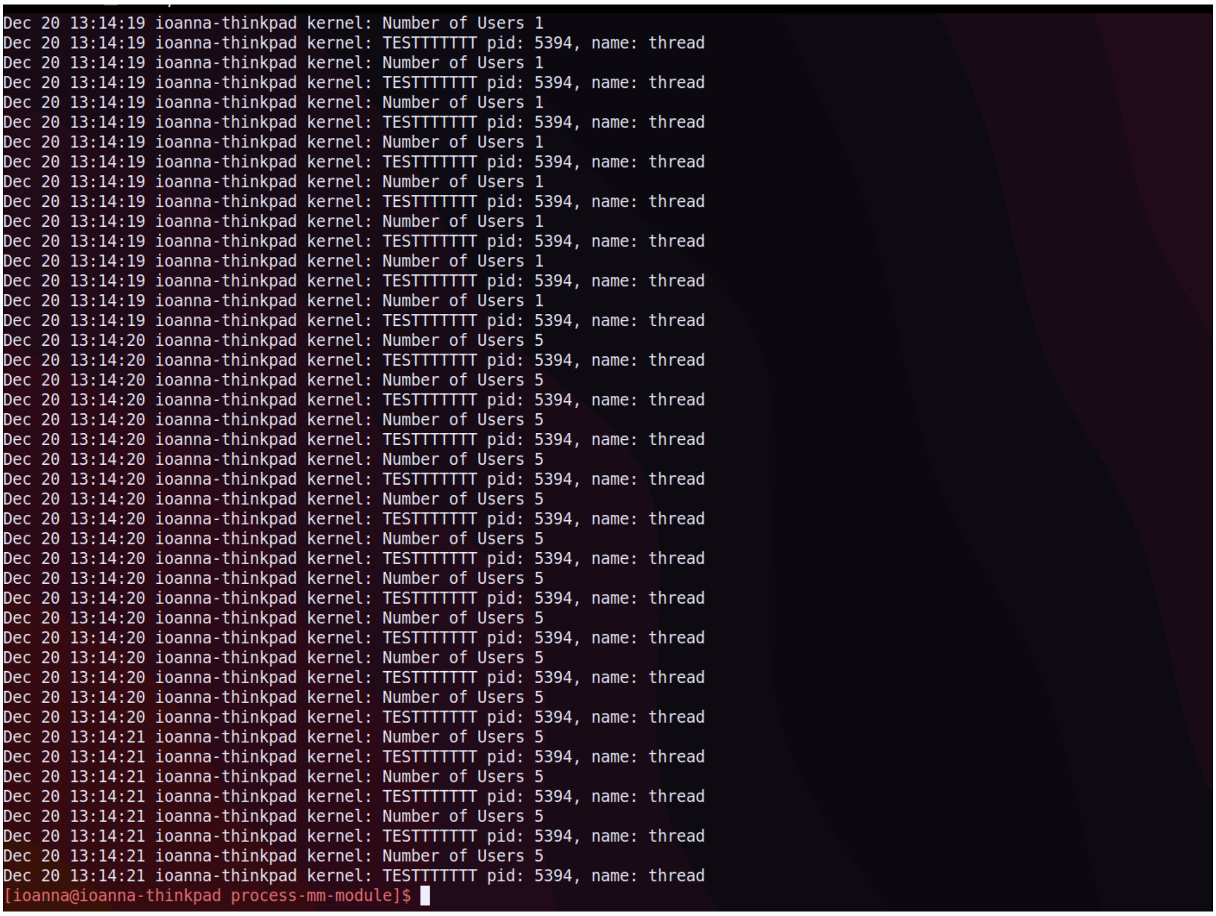
4. Άσκηση 3

Στην τελευταία άσκηση μας ζητείται να φτιάξουμε ένα κερνελ μοδυλε το οποίο θα βρίσκει το task_struct * που θα αντιστοιχεί σε μία διεργασία με ένα δοθέν PID και να εξετάσουμε το mm μέλος του task_struct *. Με απλά λόγια, θέλουμε να τυπώσουμε τον αριθμό των διεργασιών που μοιράζονται μία συγκεκριμένη θέση μνήμης.
Μέσα στο αρχείο threads.c ορίζουμε αρκετά μεγάλο sleep_time. Αρχικά κάνουμε compile το αρχείο threads.c. Έπειτα το εκτελούμε και παίρνουμε το PID της διεργασίας και το ορίζουμε κατά τη φόρτωση του module (process-mm-module.ko), όπως περιγράφεται στην εκφώνηση.
Το ζητούμενο επιτυγχάνεται τυπώνοντας το task->mm->mm_users, ο οποίος είναι ο αριθμός των ¨χρηστών' που έχει πρόσβαση σε αυτή τη θέση μνήμης.
Το task->mm->mm_users είναι μία μεταβλητή τύπου atomic_t, δηλαδή κραταάει μία ακέραια τιμή.
Η προσθήκη που απαιτείται είναι η εξής:

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

    /* 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)
    {
        /* TODO: print the number of processes accesing the process' memory. */
        printk("Number of Users %ld\n",task->mm->mm_users);
        printk("TESTTTTTT pid: %d, name: %s\n", task->pid, task->comm);
    }

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

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

```
PROBLEMS 5  OUTPUT  DEBUG CONSOLE  TERMINAL
PID: 4788
[ioanna@ioanna-thinkpad process-mm-module]$ ./thread
PID: 4891
[ioanna@ioanna-thinkpad process-mm-module]$ gcc -o thread threads.c
[ioanna@ioanna-thinkpad process-mm-module]$ ./thread
PID: 5042
[ioanna@ioanna-thinkpad process-mm-module]$ gcc -o thread threads.c
[ioanna@ioanna-thinkpad process-mm-module]$ ./thread
PID: 5394
```

Σχήμα 4: process_mm_module.c



```
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:19 ioanna-thinkpad kernel: Number of Users 1
Dec 20 13:14:19 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:20 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:20 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:21 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:21 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:21 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:21 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:21 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:21 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
Dec 20 13:14:21 ioanna-thinkpad kernel: Number of Users 5
Dec 20 13:14:21 ioanna-thinkpad kernel: TESTTTTTTT pid: 5394, name: thread
[ioanna@ioanna-thinkpad process-mm-module]$
```

Σχήμα 5: Τα logs του συστήματος

Παρατηρούμε ότι αρχικά έχουμε μόνο την διεργασία μας με PID 5394 και στην αυνέχεια έχουμε 5 χρήστες, δηλαδή την διεργασία και τους threads (τα οποία θα έχουν προφανώς το ίδιο PID με την διεργασία, αφού πρόκειται για νήματα).