Лабораторная работа №3 - Kernel API — работа с памятью.

Цель работы: Знакомство с приемами программирования ОС GNU/Linux, создание простого модуля ядра системы, работа с памятью.

Аппаратное и программное обеспечение: PC, ОС GNU/Linux, Virtualbox.

Для выполнения данной лабораторной работы необходимо установить ОС GNU/Linux в рамках системы виртуализации VirtualBox.

Источники информации:
http://www.google.ru
https://lwn.net/Kernel/LDD3/
https://github.com/martinezjavier/ldd3

Контрольные вопросы:

1. Что такое текущий процесс current в GNU/Linux.
2. Что такое параметры модуля ядра ОС GNU/Linux.
3. Как выделяется память в модулях ядра ОС GNU/Linux.

Содержание работы.

Задание 1. В рабочем каталоге пользователя создать каталог **lab3**. В данном каталоге создать файл исходного кода модуля ядра с именем lab3_ivanov.c (вместо ivanov вставить свою фамилию транслитерацией).

*Задание 2***.** На основе результата выполнения предыдущей лабораторной работы, составить программу модуля ядра системы с одним параметром (int). Данный параметр должен использоваться для передачи значения PID процесса системы. Необходимо создать структуру task_info, содержащую поле для PID процесса, и для временной метки. При загрузке модуля необходимо выделить память для экземпляра данной структуры с помощью kmalloc(). При выгрузке, модуль должен выводить строку, содержащую данный номер процесса и количество системных тиков, прошедших с момента загрузки системы. Также, при выгрузке, необходимо очистить память с помощью функции kfree(). Фрагмент кода иллюстрирующий частичное решение такой задачи представлен ниже.

```
struct task_info {
      pid_t pid;
      unsigned long timestamp;
};
static struct task_info *ti1;
static struct task_info *task_info_alloc(int pid)
{
      struct task_info *ti;
      ti = kmalloc(sizeof(*ti), GFP_KERNEL);
      if (ti == NULL)
            return NULL;
      ti->pid = pid;
      ti->timestamp = jiffies;
```

```
        return ti;
}
```

Дополнить имена функций, определенных в данном модуле, фамилией студента, выполняющего работу (транслитерацией, например была функция lkm_example_init — стала lkm_example_init_ivanov).

Задание 3. В соответствии с таблицей вариантов, модифицировать программу.

№ варианта	Задание
1	Добавить к структуре task_info еще один элемент, строковый тип. При вызове task_info_alloc присвоить этому элементу в качестве значения фамилию (транслитом) студента, выполняющего работу. При загрузке модуля вызвать task_info_alloc для текущего процесса (current→pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
2	Добавить к структуре task_info еще один элемент, строковый тип. При вызове task_info_alloc присвоить этому элементу в качестве значения фамилию (транслитом) студента, выполняющего работу. При загрузке модуля вызвать task_info_alloc для родительского (по отношению к текущему) процесса (current->parent->pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
3	Добавить к структуре task_info еще один элемент, строковый тип. При вызове task_info_alloc присвоить этому элементу в качестве значения фамилию (транслитом) студента, выполняющего работу. При загрузке модуля вызвать task_info_alloc для процесса, который последует за текущим (next_task(current)->pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
4	Добавить к структуре task_info еще один элемент, строковый тип. При вызове task_info_alloc присвоить этому элементу в качестве значения фамилию (транслитом) студента, выполняющего работу. При загрузке модуля вызвать task_info_alloc для процесса, который будет после следующего за текущим (next_task(next_task(current))->pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
5	Добавить к структуре task_info еще один элемент, тип int. При

	вызове task_info_alloc присвоить этому элементу в качестве значения PID родительского (по отношению к текущему) процесса. При загрузке модуля вызвать task_info_alloc для текущего процесса (current→pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
6	Добавить к структуре task_info еще один элемент, тип int. При вызове task_info_alloc присвоить этому элементу в качестве значения PID процесса, который последует за текущим. При загрузке модуля вызвать task_info_alloc для текущего процесса (current→pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
7	Добавить к структуре task_info еще один элемент, строковый тип. При вызове task_info_alloc присвоить этому элементу в качестве значения фамилию (транслитом) студента, выполняющего работу. При загрузке модуля вызвать task_info_alloc для процесса с PID=1 и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
8	Добавить к структуре task_info еще один элемент, тип int. При вызове task_info_alloc присвоить этому элементу в качестве значения количество системных тиков, прошедших с момента загрузки системы для процесса с PID=0. При загрузке модуля вызвать task_info_alloc для процесса, который последует за текущим (next_task(current)->pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
9	Добавить к структуре task_info еще один элемент, тип int. При вызове task_info_alloc присвоить этому элементу в качестве значения количество системных тиков, прошедших с момента загрузки системы для процесса с PID=2. При загрузке модуля вызвать task_info_alloc для процесса, который будет после следующего за текущим (next_task(next_task(current))->pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).
10	Добавить к структуре task_info еще один элемент, тип int. При вызове task_info_alloc присвоить этому элементу в качестве значения количество системных тиков, прошедших с момента загрузки системы для процесса с PID=3. При загрузке модуля вызвать task_info_alloc для текущего процесса

	(current→pid) и вывести по нему аналогичную информацию при выгрузке модуля (номер процесса и количество системных тиков, прошедших с момента загрузки системы).

Сохранить исходный код программы, продемонстрировать работу программы преподавателю.

Содержание отчета.

1. Титульный лист с указанием номера и наименования работы, ф.и.о. студента, номера учебной группы.
2. Исходный код программы в соответствии с заданиями.
3. Снимки экрана терминала, на которых видно процесс компиляции и тестирования модуля ядра.
   Вместе с отчетом приложить файл исходного кода модуля, а также Makefile!