[**CSCE340101 - Operating systems**](https://blackboard.aucegypt.edu/webapps/blackboard/execute/launcher?type=Course&id=_67070_1&url=) **| Fall 2018**

**Manar Abdelatty | 900152684**

**Project 2**

**Part 1: Introduction to Kernel Modules**

* **Skeleton**

Kernel modules were developed using the skeleton provided in the textbook. Firstly, module\_init() function is defined as an entry point to when the module is loaded to the kernel and module\_exit() as an exit point to when the module is removed from the kernel. These two functions are registered using the *module\_init* and *module\_exit*  macros.

* **Creating Proc Entry**

In order to create a module entry in the proc pseudo file system, the *proc\_create()* is called in the module’s entry point such that whenever the module is loaded to the Linux kernel an entry in the proc is created. In the module’s exit point, *remove\_proc\_entry()*is called to remove its entry from the proc.

In the *proc\_read()* function, information from the kernel is copied to the user space buffer.

1. **Jiffies Module**

This module prints the current value of the jiffies, number of timer interrupts since the system was booted, when the user runs the command: cat /proc/jiffies. The jiffies value to be printed is copied to the user space buffer in the *proc\_read()* function and is printed on the terminal.

1. **Seconds Module**

This module prints the amount of seconds passed since the module was loaded till the user enters the following command: cat /proc/seconds

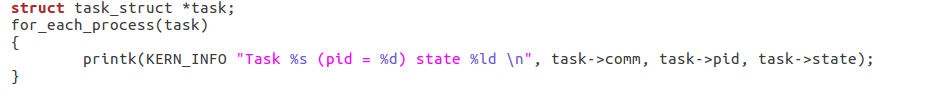
The number of seconds elapsed is calculated according to the following equation:

In the module’s entry point, the value of the jiffies is saved in a global variable *jiffies\_init*. When the proc*\_read* function is called the initial jiffies value is subtracted from the current jiffies value and the result is divided by the *HZ* value which defines the frequency of the timer interrupts.

**Part 2: Listing Tasks**

1. **Iterating over Tasks Linearly**

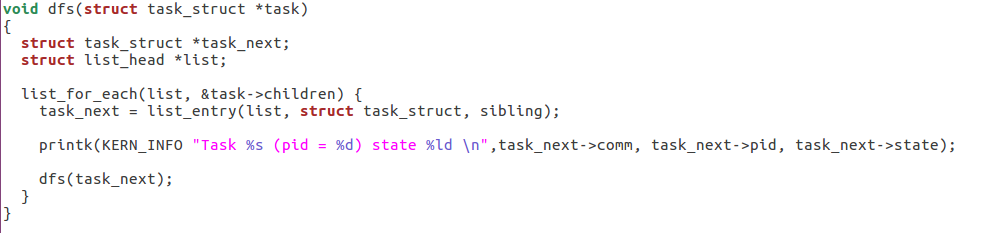
In this part, a Linux kernel module (linear) is created to list the current running tasks linearly. The *for\_each\_process* macro is used to iterate over the tasks in the module’s entry point as shown below.



The macro takes a pointer *task* that points to a struct of type *task\_struct*. The task struct contents, task command, process id and state, are accessed and then printed in the kernel buffer.

1. **Iterating over Tasks with a Depth First Search Tree**

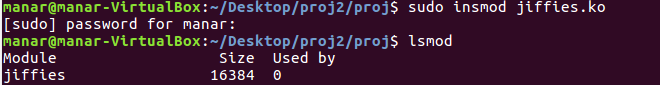
In the DFS module, iterating over tasks is done with depth first search tree. The *DFS()* recursively calls *list\_for\_each* macro on each task and its children.



*DFS()* is called in the entry point of the module on the *init\_task* pointer.

**Test Cases**

1. **Jiffies Module**
2. Inserting module to the kernel



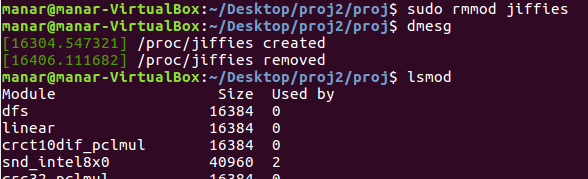
1. Checking dmesg buffer after insertion



1. Running Cat



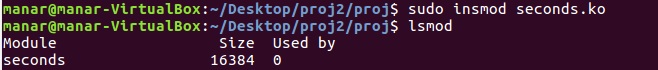
1. Removing module



1. Making sure that module file is removed from proc

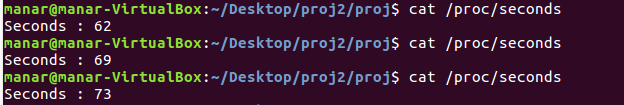


1. **Seconds Module**
2. **Insertion**

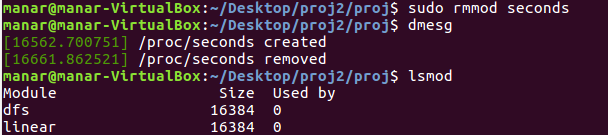
****

****

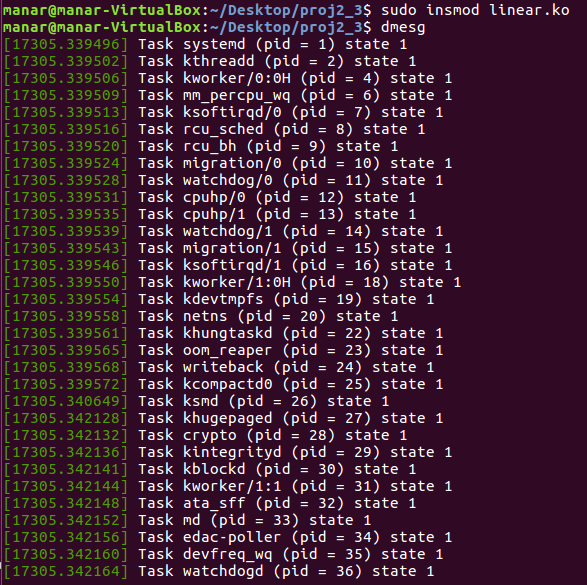
1. **Running Cat**

****

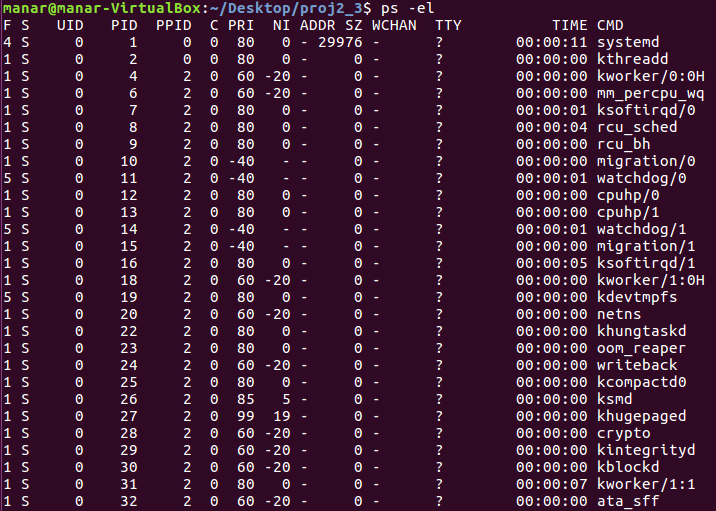
1. **Removing Module**

****

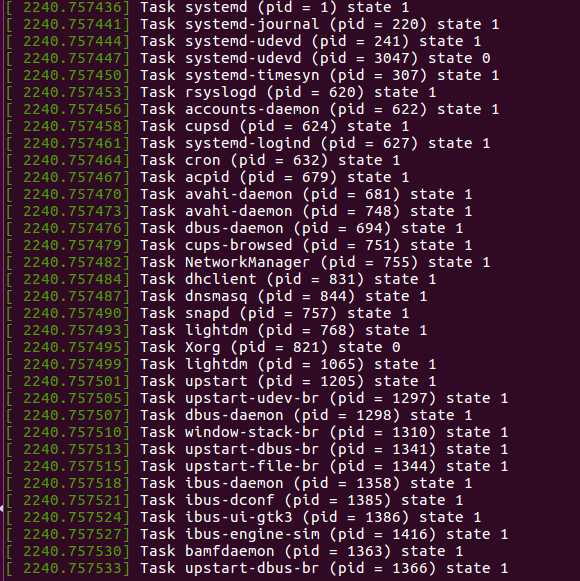
1. **Linear Listing**

****

**2- Ps –el**

****

1. **DFS**

****