**CPTS 360 Programming Assignment 3: Kernel Module Documentation**

**Author**: Darasimi Ogunbinu-Peters  
**Date**: 11/7/2024

**1. Overview**

This documentation describes the implementation of a Linux Kernel Module for tracking CPU usage of registered user processes. The module allows user applications to register their process IDs (PIDs) and periodically updates their CPU time usage, removing entries when a process terminates.

The module utilizes:

* The /proc filesystem for interfacing with user applications.
* Kernel timers and workqueues for periodic updates.
* A linked list to store and manage information about each registered process.

**2. Function Descriptions**

**kmlab\_init**

This function is executed when the module is loaded. It initializes the /proc/kmlab directory and its status file for interfacing with user applications. It also sets up a timer to trigger the timer\_callback function every 5 seconds.

**timer\_callback**

This function is executed by the kernel timer every 5 seconds. It iterates through the linked list of registered processes and updates their CPU time using get\_cpu\_use(). If a process has terminated, it removes the process from the list.

**kmlab\_write**

This function registers a user process by adding its PID to the linked list. When a user application writes to /proc/kmlab/status, this function reads the PID and stores it in a new linked list entry.

**kmlab\_read**

This function is triggered when a user reads from /proc/kmlab/status. It formats and outputs the current CPU times for all registered PIDs, displaying each as "PID [pid]: CPU Time [time] ms."

**kmlab\_exit**

This function is executed when the module is removed. It deletes all entries in the linked list, removes the /proc/kmlab directory and status file, and cancels the timer.

**3. Implementation Details**

**Linked List**

The linked list is defined using LIST\_HEAD(proc\_head). Each entry is an instance of the proc\_list structure, which contains the PID and the CPU time for each process.

**Registering and Removing Processes**

Processes register by writing to /proc/kmlab/status, which adds them to the linked list. The timer\_callback function removes terminated processes by checking if get\_cpu\_use() returns -1.

**CPU Time Updates**

The kernel timer triggers timer\_callback, which updates CPU times every 5 seconds. Updated times are recorded in the linked list.

**Directory and File Creation**

The /proc/kmlab directory and status file are created in kmlab\_init. They are removed in kmlab\_exit.

**4. Testing and Output Verification**

**Test Setup**

1. Compiled userapp.c to create the user application executable userapp.
2. Loaded the kernel module with insmod kmlab.ko.
3. Registered processes by executing ./userapp 10 & and ./userapp 15 &.

**Observed Output**

**/proc/kmlab/status**

Example output after running cat /proc/kmlab/status:

PID 16467: CPU Time 7328000000 ms

PID 16503: CPU Time 6900000000 ms

This output shows registered PIDs and their CPU times, which are updated every 5 seconds.

**dmesg Logs**

dmesg logs confirmed CPU time updates and process removals. Example log:

[ 4611.650131] Registered PID 16467

[ 4614.381607] Updated CPU time for PID 16467: 2572000000 ms

[ 4622.491134] Registered PID 16503

[ 4629.738927] Updated CPU time for PID 16503: 6900000000 ms

[ 4634.859788] Removed terminated process PID 3827912381

**5. Challenges and Solutions**

1. **proc\_create Compatibility**: The assignment required creating a /proc interface, but proc\_create required adaptation to use proc\_ops for compatibility with the kernel version.
2. **Memory Management and Process Removal**: Safely removing terminated processes required list\_for\_each\_entry\_safe to avoid issues when modifying the list during iteration.
3. **Testing Output Consistency**: Ensuring consistent output in /proc/kmlab/status involved frequent testing with userapp to confirm accurate registration and removal.