Programming Assignment 5 Writeup

In this document, all defined functions and used macros, structs and functions from included headers in the source code will be listed below with full description.

1. Printk
   Defined in the Linux kernel source code.
   Signature:
   int printk(const char *fmt, ...);
   Parameters:
   The first parameter (fmt) takes a string, followed by any number of other parameters. A log level could be concatenated to the fmt string.
   Description:
   All strings are displayed in the kernel log buffer. The kernel log buffer can be seen by the user by typing `dmesg` in the command line interface. The function printk() could take any number of parameters just like printf( ). Printk() has the ability to parse log level strings which is concatenated to the fmt string, for example:
   printk(KERN_INFO “message: %s\n”, arg);
   The log level determines the message importance. If there is no log level used in the string the message is printed using log level: KERN_DEFAULT.

2. for_each_process
   Defined on line 601 in linux/sched/signal.h.
   Signature:
   for_each_process(struct task_struct* process)
   Parameters:
   This macro takes one parameter and it’s a pointer of type task_struct.
   Description:
   This macro is used to iterate over all active processes. It’s usually used to print processes information such as name, PID, state and more.

3. Struct task_struct
   Defined on line 1475 in linux/sched.h.
   Signature:
   Struct task_struct* task_list;
   Parameters:
   This is a data structure (no parameters).

Description:
This data struct contains the information of a process such as name, PID, state, priority, static priority, name priority and much more. It also has list of children and siblings processes if they exists as well as a pointer to the parent process.

4. Struct list_head
   Defined on line 178 in linux/types.h
   Signature:
   Struct list_head* list;
   Parameters:
   This is a data structure (no parameters).
   Description:
   This data struct is used to create a doubly linked list. It has only two pointers *next and *prev of type struct list_head to be able to traverse a doubly linked list.

5. list_for_each
   Defined on line 570 in linux/list.h.
   Signature:
   list_for_each(pos, head)
   Parameters:
   This macro takes two parameters. The first is a doubly linked list of type struct list_head and the second one is the list head.
   Description:
   This macro is used to iterate over a doubly linked list given a position and a head for a list. For this assignment, it will be used to traverse over child processes doubly linked list (if they exist).

6. list_entry
   Defined on line 351 in linux/list.h.
   Signature:
   list_entry(ptr, type, member)
   Parameters:
   This macro has three parameters. The first one is a pointer of type struct list_head, the second one is the type of the struct for this entry and the third one the name of the list in the struct entry.
   Description:
   This macro is used to fetch the list of children processes for each process. In this assignment, it’s used to find each child process struct task_struct and print the child process details.

7. module_param
   Defined on line 126 in linux/moduleparam.h
   Signature:
   Module_param(name, type, perm)
   Parameters:
   This macro has three parameters. The first one is the name of the variable, the second one is the type of the variable and the third is the permission of the corresponding file in sysfs
   Description:
   This macro allows the user to enter input parameters to the LKM. The variable and the module_param macro have to be declared in the global scope. In this assignment, this macro will be used to allow the used enter an integer number that represent a PID so that to print all the processes information with PID's greater than this integer value.