Ozan Can Altıok 21001016
Metin Can Siper 21001399
Umut Utku Çalış 21000227

CS342 OPERATING SYSTEMS PROJECT #4 REPORT

How The Module is Implemented

Firstly, we learned the kernel version of our Linux system by typing `uname -r` command. Then, we researched the structure of the kernel, with the version we have. At the final stage, we printed out the information we're asked, using proper structs and their properties. For the code implementation of the code, please see the glossary.

How The Module is Tested

To test the module, the list of currently working processes were inspected using `ps aux` command. Then, an arbitrary process ID is selected among those processes. We chose the process with the PID 2286, then the module is inserted into the kernel with the pid parameter 2286 (`insmod ./process_info.ko pid=2286`). The output is printed into a text file, then the virtual memory part was compared to the actual memory mapping of the process 2286 (the map was generated by typing `cat /proc/2286/maps`). The VM results of both maps and the module output gives the same addresses. The outputs are included in the glossary part.

The Information Presented in the Module

– The PID of process
– The currently opened files by the process
– Virtual memory information
– Start, end and sizes of code, data, main arguments and the environment variables
– Total virtual memory area and the number of frames used
– Virtual memory areas (start, end and the size), together with the stack area
– File system information
– Root directory
– Working directory

Glossary

- Module C Code

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/sched.h>
#include <linux/rcupdate.h>
#include <linux/fdtable.h>
#include <linux/fs.h>
#include <linux/fs_struct.h>
#include <linux/dcache.h>
#include <linux/slab.h>
#include <linux/kernel.h>
#include <linux/errno.h>
#include <linux/stat.h>
#include <linux/mm.h>
```

```
#include <linux/highmem.h>
#include <asm/pgtable.h>

#define BUFSIZE 100

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Ozan Can Altiok, Metin Can Siper, Umut Utku Calis");

static int pid = 0;

module_param( pid, int, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
MODULE_PARM_DESC( pid, "PID of the process");

static int __init processinfo_init(void) {
	printk( KERN_INFO "CS342 Project 4: Kernel Module¥n");
	printk( KERN_INFO "Starting module...¥n");

	struct task_struct *task = current;
	struct task_struct *desiredTask = NULL;

	for_each_process( task) {
		if ( task->pid == pid) {
			desiredTask = task;
		}
	}

	if ( desiredTask != NULL) {
		printk( KERN_INFO "--A process is found with the PID = %d--¥n", pid);
		printk( KERN_INFO "--The curently opened files information--¥n");
		struct fdtable *filesTable;
		struct path fPath;
		char *filePath;
		char *buffer = (char *) kmalloc( GFP_KERNEL, BUFSIZE * sizeof( char) );

		filesTable = files_fdtable( desiredTask->files);

		int i = 0;
		while ( filesTable->fd[i]) {
			fPath = filesTable->fd[i]->f_path;
			filePath = d_path( &fPath, buffer, BUFSIZE * sizeof( char) );
			printk( KERN_INFO "¥t%s¥n", filePath);
			i++;
		}

		printk( KERN_INFO "--Memory Management Information--¥n" );

		struct mm_struct* mm = desiredTask->mm;
		printk( KERN_INFO "[CODE START]¥t[CODE END]¥t[CODE SIZE]¥n");
		printk( KERN_INFO "%lx¥t¥t%lx¥t%lu¥n", mm->start_code,
			mm->end_code, mm->end_code - mm->start_code );

		printk( KERN_INFO "[DATA START]¥t[DATA END]¥t[DATA SIZE]¥n");
```

```
            printk( KERN_INFO "%lx¥t¥t%lx¥t%lu¥n¥n", mm->start_data,
                   mm->end_data, mm->end_data - mm->start_data );

            printk( KERN_INFO "¥nNotice: The stack data will be written in virtual memory
part.¥n" );

            printk( KERN_INFO "[ARG START]¥t[ARG END]¥t[ARG SIZE]¥n");
            printk( KERN_INFO "%lx¥t¥t%lx¥t%lu¥n¥n", mm->arg_start,
                   mm->arg_end, mm->arg_end - mm->arg_start );

            printk( KERN_INFO "[ENV START]¥t[ENV END]¥t[ENV SIZE]¥n");
            printk( KERN_INFO "%lx¥t¥t%lx¥t%lu¥n¥n", mm->env_start,
                   mm->env_end, mm->env_end - mm->env_start );

            printk( KERN_INFO "Total VM area = %lu¥n", mm->total_vm);
            printk( KERN_INFO "Number of frames used by the process = %lu¥n¥n",
get_mm_rss( mm) );

            struct vm_area_struct *mmap = mm->mmap;

            printk( KERN_INFO "--Virtual Memory Information--¥n" );
            printk( KERN_INFO "[VM START]¥t[VM_END]¥t[VM_SIZE]");
            while( mmap != NULL )
            {
                   if( mmap -> vm_next == NULL ) {
                          printk( KERN_INFO "¥nThe stack information of the process:¥n");
                          printk( KERN_INFO "[STACK START]¥t[STACK END]¥t[STACK SIZE]¥n" );
                   }
                   printk( KERN_INFO "%lx¥t¥t%lx¥t%lu¥n", mmap -> vm_start,
                          mmap -> vm_end, mmap -> vm_end - mmap -> vm_start );
                   mmap = mmap -> vm_next;
            }

            printk( KERN_INFO "--The filesystem information--¥n");
            struct fs_struct *filesStruct = desiredTask->fs;
            printk( KERN_INFO "¥tRoot: %s¥n", filesStruct->root.dentry->d_name.name);
            printk( KERN_INFO "¥tWorking Directory: %s¥n", filesStruct->pwd.dentry->d_name.name);

            /* -- THIS PART IS ABOUT THE PAGE TABLE INFORMATION
             * -- BUT THERE ARE SOME STRANGE ERRORS
             * -- SO WE COMMENTED THIS PART
            printk( KERN_INFO "--Page Table Information--¥n");
            pgd_t *PGD;
            pud_t *PUD;
            pmd_t *PMD;
            pte_t *PTE;
            struct page *processPage = NULL;
            int j, k, l, m;

            //printk( KERN_INFO "Totalhighpages = %llu¥n", totalhigh_pages);
            //printk( KERN_INFO "Pud Size = %llu¥n", sizeof( pgd_t) );
            for ( m = 0; m < totalhigh_pages; m++ ) {
                    PGD = pgd_offset( mm, sizeof( pgd_t) * m);
                    if ( !pgd_none( *PGD) && !pgd_bad( *PGD) ) {
                          for ( j = 0; j < totalhigh_pages; j++ ) {
                                PUD = pud_offset( PGD, sizeof( pud_t) * j);
                                if ( !pud_none( *PUD) && !pud_bad( *PUD) ) {
                                      for ( k = 0; k < totalhigh_pages; k++ ) {
                                            PMD = pmd_offset( PUD, sizeof( pmd_t) * k);
                                            if ( !pmd_none( *PMD) && !pmd_bad( *PMD) ) {
                                                  for ( l = 0; l < totalhigh_pages; l++ ) {
                                                        PTE = pte_offset_map( PMD,
sizeof( pte_t) * l);

                                                        if ( PTE) {
                                                              processPage =
pte_page( *PTE);

                                                              if ( processPage) {
                                                                    printk( KERN_INFO
"¥tPage frame struct address = %p", processPage);
                                                              }
                                                              pte_unmap( PTE);
                                                        }
                                                  }
                                            }
                                      }
                                }
                          }
                    }
              }

              if ( !processPage) {
                    printk( KERN_INFO "There is no page table information.¥n");
              }
              */
       }
       else {
              printk( KERN_INFO "There is not a process with PID = %d, exiting...¥n", pid);
       }

       return 0;
}

static void __exit processinfo_exit( void) {
       printk( KERN_INFO "The module successfully removed.¥n");
}

module_init( processinfo_init);
module_exit( processinfo_exit);
```

- Makefile

```
obj-m += process_info.o

all:
       make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

```
clean:
	make -C /lib/mobules/$(shell uname -r)/build M=$(PWD) clean
```

- Output Generated by A Process

```
[  795.923506] --A process is found with the PID = 2286--
[  795.923508] --The curently opened files information--
[  795.923515]       socket:[15417]
[  795.923519]       socket:[15417]
[  795.923524]       /dev/null
[  795.923527] --Memory Management Information--
[  795.923529] [CODE START] [CODE END]    [CODE SIZE]
[  795.923534] 8048000             804a2f88952
[  795.923536] [DATA START] [DATA END]    [DATA SIZE]
[  795.923540] 804bf00             804c140576
[  795.923540]
[  795.923545]
[  795.923545] Notice: The stack data will be written in virtual memory part.
[  795.923549] [ARG START]  [ARG END]     [ARG SIZE]
[  795.923552] bfe348a6            bfe348b7      17
[  795.923552]
[  795.923556] [ENV START]  [ENV END]     [ENV SIZE]
[  795.923559] bfe348b7            bfe34fda      1827
[  795.923559]
[  795.923563] Total VM area = 605
[  795.923566] Number of frames used by the process = 185
[  795.923566]
[  795.923570] --Virtual Memory Information--
[  795.923573] [VM START]   [VM_END]      [VM_SIZE]
[  795.923578] 8048000             804b00012288
[  795.923582] 804b000             804c0004096
[  795.923585] 804c000             804d0004096
[  795.923588] 9998000             99b9000135168
[  795.923592] b75cf000            b75da000      45056
[  795.923595] b75da000            b75db000      4096
[  795.923598] b75db000            b75dc000      4096
[  795.923602] b75dc000            b75e6000      40960
[  795.923605] b75e6000            b75e7000      4096
[  795.923608] b75e7000            b75e8000      4096
[  795.923611] b75e8000            b75fd000      86016
[  795.923615] b75fd000            b75fe000      4096
[  795.923618] b75fe000            b75ff000      4096
[  795.923621] b75ff000            b7601000      8192
[  795.923624] b7601000            b7608000      28672
[  795.923627] b7608000            b7609000      4096
[  795.923630] b7609000            b760a000      4096
[  795.923634] b760a000            b760c000      8192
[  795.923637] b760c000            b77b5000      1740800
[  795.923640] b77b5000            b77b7000      8192
[  795.923643] b77b7000            b77b8000      4096
[  795.923646] b77b8000            b77bb000      12288
[  795.923650] b77bb000            b77bd000      8192
[  795.923653] b77bd000           b77be000      4096
[  795.923656] b77be000           b77bf000      4096
[  795.923659] b77d0000           b77d2000      8192
[  795.923662] b77d2000           b77d3000      4096
[  795.923666] b77d3000           b77f3000      131072
[  795.923669] b77f3000           b77f4000      4096
[  795.923672] b77f4000           b77f5000      4096
[  795.923674]
[  795.923674] The stack information of the process:
[  795.923678] [STACK START][STACK END]   [STACK SIZE]
[  795.923681] bfe13000           bfe35000      139264
[  795.923684] --The filesystem information--
[  795.923687]       Root: /
[  795.923689]       Working Directory: /
[  797.764310] The module successfully removed.
```

- The Actual VM Mapping

```
08048000-0804b000 r-xp 00000000 08:01 7680       /usr/lib/libvte9/gnome-pty-helper
0804b000-0804c000 r--p 00002000 08:01 7680       /usr/lib/libvte9/gnome-pty-helper
0804c000-0804d000 rw-p 00003000 08:01 7680       /usr/lib/libvte9/gnome-pty-helper
09998000-099b9000 rw-p 00000000 00:00 0          [heap]
b75cf000-b75da000 r-xp 00000000 08:01 132328     /lib/i386-linux-gnu/libnss_files-2.19.so
b75da000-b75db000 r--p 0000a000 08:01 132328     /lib/i386-linux-gnu/libnss_files-2.19.so
b75db000-b75dc000 rw-p 0000b000 08:01 132328     /lib/i386-linux-gnu/libnss_files-2.19.so
b75dc000-b75e6000 r-xp 00000000 08:01 132338     /lib/i386-linux-gnu/libnss_nis-2.19.so
b75e6000-b75e7000 r--p 00009000 08:01 132338     /lib/i386-linux-gnu/libnss_nis-2.19.so
b75e7000-b75e8000 rw-p 0000a000 08:01 132338     /lib/i386-linux-gnu/libnss_nis-2.19.so
b75e8000-b75fd000 r-xp 00000000 08:01 132322     /lib/i386-linux-gnu/libnsl-2.19.so
b75fd000-b75fe000 r--p 00015000 08:01 132322     /lib/i386-linux-gnu/libnsl-2.19.so
b75fe000-b75ff000 rw-p 00016000 08:01 132322     /lib/i386-linux-gnu/libnsl-2.19.so
b75ff000-b7601000 rw-p 00000000 00:00 0
b7601000-b7608000 r-xp 00000000 08:01 132324     /lib/i386-linux-gnu/libnss_compat-2.19.so
b7608000-b7609000 r--p 00006000 08:01 132324     /lib/i386-linux-gnu/libnss_compat-2.19.so
b7609000-b760a000 rw-p 00007000 08:01 132324     /lib/i386-linux-gnu/libnss_compat-2.19.so
b760a000-b760c000 rw-p 00000000 00:00 0
b760c000-b77b5000 r-xp 00000000 08:01 132253     /lib/i386-linux-gnu/libc-2.19.so
b77b5000-b77b7000 r--p 001a9000 08:01 132253     /lib/i386-linux-gnu/libc-2.19.so
b77b7000-b77b8000 rw-p 001ab000 08:01 132253     /lib/i386-linux-gnu/libc-2.19.so
b77b8000-b77bb000 rw-p 00000000 00:00 0
b77bb000-b77bd000 r-xp 00000000 08:01 132406     /lib/i386-linux-gnu/libutil-2.19.so
b77bd000-b77be000 r--p 00001000 08:01 132406     /lib/i386-linux-gnu/libutil-2.19.so
b77be000-b77bf000 rw-p 00002000 08:01 132406     /lib/i386-linux-gnu/libutil-2.19.so
b77d0000-b77d2000 rw-p 00000000 00:00 0
b77d2000-b77d3000 r-xp 00000000 00:00 0          [vdso]
b77d3000-b77f3000 r-xp 00000000 08:01 132229     /lib/i386-linux-gnu/ld-2.19.so
b77f3000-b77f4000 r--p 0001f000 08:01 132229     /lib/i386-linux-gnu/ld-2.19.so
b77f4000-b77f5000 rw-p 00020000 08:01 132229     /lib/i386-linux-gnu/ld-2.19.so
bfe14000-bfe35000 rw-p 00000000 00:00 0          [stack]
```