OS LAB – Week 5

Question:

Creating a header file ‘processInfo.h’ that will contain the necessary function declarations, structure declarations, macros, etc.

asmlinkage long sys\_listProcessInfo(void);

define your system call in ‘listProcessInfo.c’.

#include<linux/kernel.h>

#include<linux/init.h>

#include<linux/sched.h>

#include<linux/syscalls.h>

#include "processInfo.h"

asmlinkage long sys\_listProcessInfo(void) {

struct task\_struct \*proces;

for\_each\_process(proces) {

printk(

"Process: %s\n \

PID\_Number: %ld\n \

Process State: %ld\n \

Priority: %ld\n \

RT\_Priority: %ld\n \

Static Priority: %ld\n \

Normal Priority: %ld\n", \

proces->comm, \

(long)task\_pid\_nr(proces), \

(long)proces->state, \

(long)proces->prio, \

(long)proces->rt\_priority, \

(long)proces->static\_prio, \

(long)proces->normal\_prio \

);

if(proces->parent)

printk(

"Parent process: %s, \

PID\_Number: %ld", \

proces->parent->comm, \

(long)task\_pid\_nr(proces->parent) \

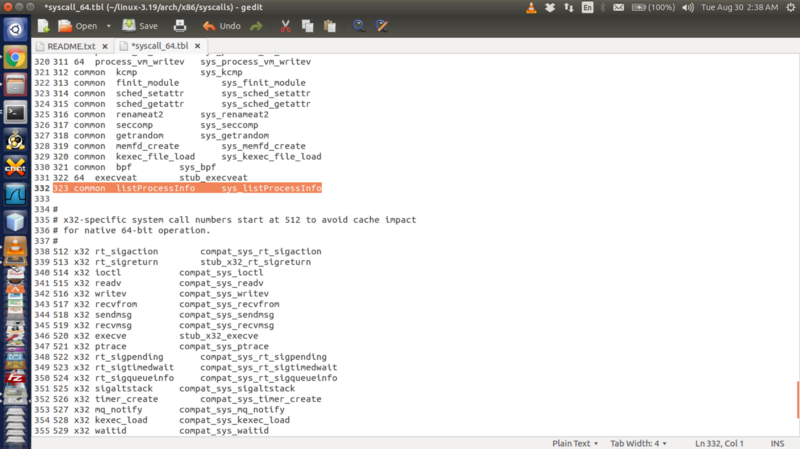
);

printk("\n\n");

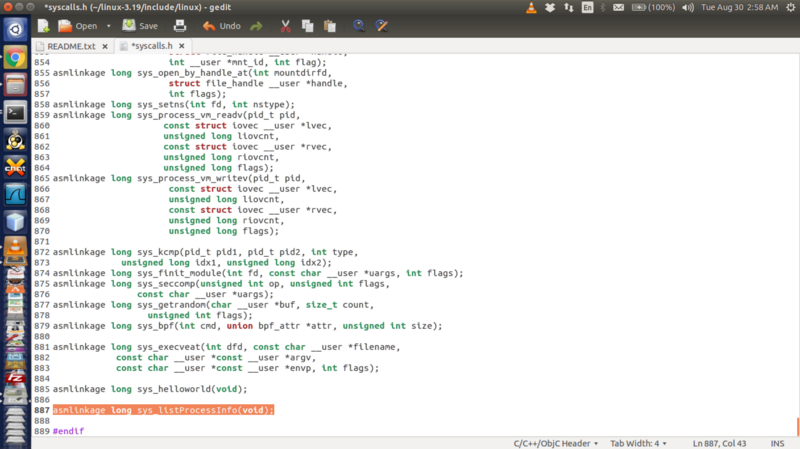
}

return 0;

Modifying necessary kernel files to integrate our system call



Altering the syscalls.h file, which can be found in /include/linux/ and adding the following line to the end of the file (before the #endif)

asmlinkage long sys\_listProcessInfo(void);

Testing the system call:

To test the system call write a simple ‘test.c’ function (it can be placed in any directory) as follows:

#include <stdio.h>

#include <linux/kernel.h>

#include <sys/syscall.h>

#include <unistd.h>

int main()

{ printf("Invoking 'listProcessInfo' system call");

long int ret\_status = syscall(323); // 323 is the syscall number

if(ret\_status == 0)

printf("System call 'listProcessInfo' executed correctly. Use dmesg to check processInfo\n");

else

printf("System call 'listProcessInfo' did not execute as expected\n");

return 0;

}

**Kernel Compilation**

