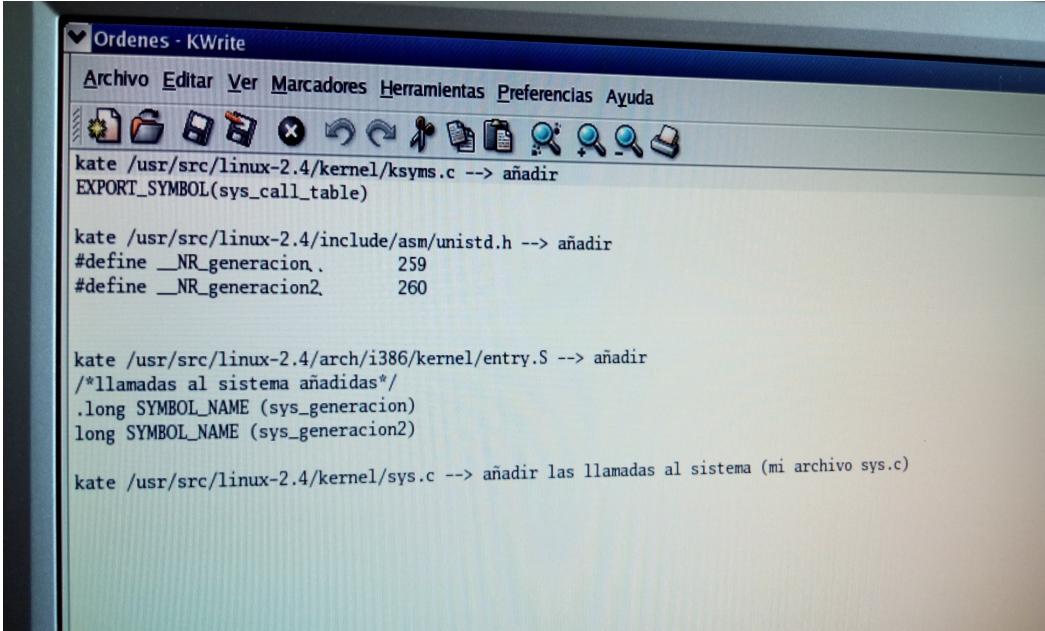


Guion Práctica 3 – José Antonio Guerrero Avilés

Para esta práctica he tenido que usar las siguientes ordenes para editar los ficheros correspondientes para la realización de la misma:



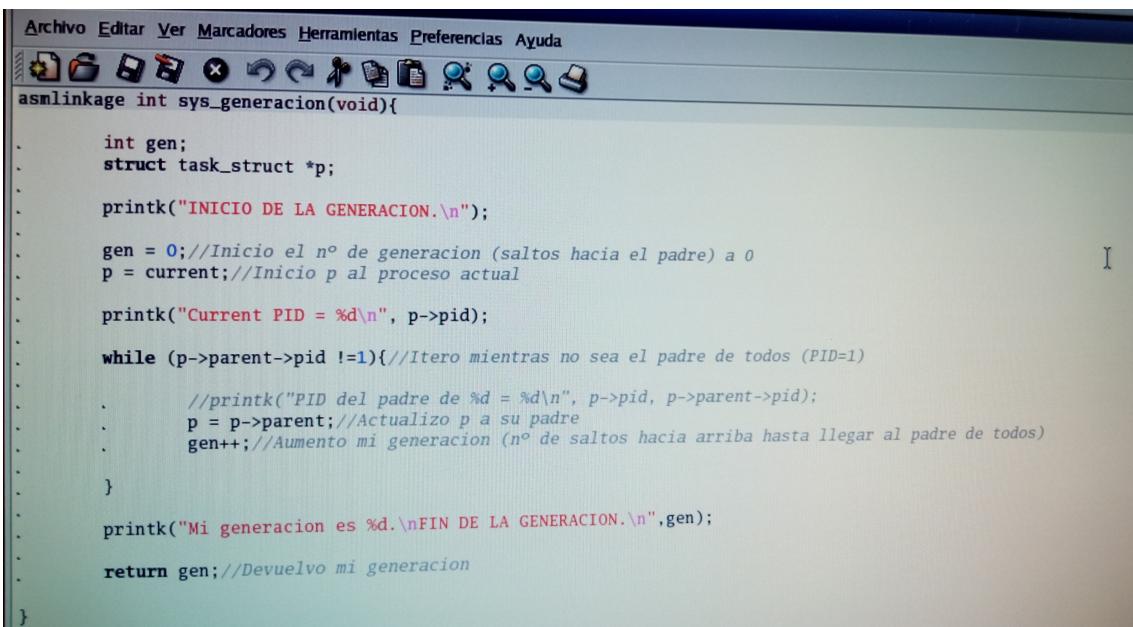
The screenshot shows a window titled "Ordenes - KWrite" with a menu bar including Archivo, Editar, Ver, Marcadores, Herramientas, Preferencias, and Ayuda. Below the menu is a toolbar with various icons. The main area contains a list of file paths and their modifications:

- kate /usr/src/linux-2.4/kernel/ksyms.c --> añadir
EXPORT_SYMBOL(sys_call_table)
- kate /usr/src/linux-2.4/include/asm/unistd.h --> añadir
#define __NR_generacion_ 259
#define __NR_generacion2 260
- kate /usr/src/linux-2.4/arch/i386/kernel/entry.S --> añadir
/*llamadas al sistema añadidas*/
.long SYMBOL_NAME (sys_generacion)
.long SYMBOL_NAME (sys_generacion2)
- kate /usr/src/linux-2.4/kernel/sys.c --> añadir las llamadas al sistema (mi archivo sys.c)

Como se puede ver, tenemos que exportar la tabla de símbolos y añadir la cabecera de las llamadas que queremos incluir y un identificador, que deberá continuar la numeración que ya hay empezada en el archivo unistd.h.

He implementado tanto la parte obligatoria (devolver la generación del proceso actual) como la parte optativa (devolver la generación de un proceso que le pasamos por argumento).

La primera corresponde al código de “sys_generacion” y puede verse en la siguiente imagen:



The screenshot shows a window titled "Archivos - KWrite" with a menu bar including Archivo, Editar, Ver, Marcadores, Herramientas, Preferencias, and Ayuda. Below the menu is a toolbar with various icons. The main area contains the C code for the sys_generacion function:

```
asmlinkage int sys_generacion(void){  
    .    int gen;  
    .    struct task_struct *p;  
    .  
    printk("INICIO DE LA GENERACION.\n");  

```

El código de la parte opcional se corresponde con “sys_generacion2” y puede verse en la siguiente imagen:

```
asm linkage int sys_generacion2(pid_t pid){//Le pase el PID que quiero encontrar
    int gen;
    struct task_struct *p;
    printk("INICIO DE LA GENERACION2.\n");
    gen = 0;//Inicia el nº de generacion (saltos hacia el padre) a 0
    int encontrado = 0;
    /*
    Una forma de hacerlo
    fcr_each_process(p){
        if(p->pid == pid){//haga algo
        else{}//otra cosa
    }*/
    //Otra forma de hacerlo:
    //Comienzo en init_task y recorro todos para encontrar el proceso cuya pid me pasan como argumento
    for (p = &init_task ; (p = next_task(p)) != &init_task && encontrado == 0; ){
        if (p->pid == pid){
            encontrado = 1;
            while (p->parent->pid !=1){//Itero mientras no sea el padre de todos (PID=1)
                //	printk("PID del padre de %d = %d\n", p->pid, p->parent->pid);
                p = p->parent;//Actualizo p a su padre
                gen++;//Aumento mi generacion (nº de saltos hacia arriba hasta llegar al padre de todos)
            }
        }
    }
    if (encontrado == 1){
        printk ("La generación del proceso con PID = %d es %d\nFIN DE LA GENERACION2.\n", pid, gen);
    } else{
        printk("La generación ha fallado. No existe ningún proceso con PID = %d.\n",pid);
        gen = -ERSCH;
    }
}
return gen;//Levuelvo la generación e el error
```

Una vez editados todos los archivos, recompilamos como en la práctica anterior y ya tendremos nuestras llamadas al sistema agregadas.

Para probarlas, debemos crear el programa con el siguiente código:

```
generacion.c - KWrite
Archivo Editar Ver Marcadores Herramientas Preferencias Ayuda
# include<stdio.h>
# include<stdlib.h>
# include<asm/unistd.h>
# include<errno.h>

int x;
_syscall1(generacion2,int,x);

main(){
    printf("Dame numero:\n");
    scanf("%d",&x);
    int res = generacion2(x);
    printf("Generacion: %d", res);
}
```

La imagen anterior es para el caso de la llamada “sys_generacion2”, que recibe parámetros, para el caso de la llamada “sys_generación” se simplificaría aún más ya que esta no recibe nada.

La llamada sería

```
_syscall(int,generacion);
```

y en el main simplemente mostraríamos el resultado de la llamada, que en nuestro caso, es la generación.