



Práctica Final

LIN - Curso 2015-2016



Práctica Final



- Existen 2 variantes de la práctica final
 - Cada estudiante del mismo grupo tiene asociada una variante distinta
 - El listado con las variantes asignadas se encuentra en el Campus Virtual



Práctica Final



Parte A (sobre MV de Debian)

- **Variante 1:** Extender módulo del kernel de la Práctica 3A para que gestione múltiples listas enlazadas de enteros
- **Variante 2:** Extender módulo del kernel de la Práctica 3B para que gestione múltiples “FIFOs”

Parte B (sobre MV de Android-x86)

- 1 Hacer funcionar el módulo de la parte A en MV de Android-x86
- 2 Crear programa de usuario en C que interactúe con el módulo del kernel



Parte A (Ambas variantes)

- Al cargar el módulo, éste creará un directorio \$PROC_DIRECTORY en el sistema de ficheros /proc
 - Variante 1: PROC_DIRECTORY=/proc/multilist
 - Variante 2: PROC_DIRECTORY=/proc/multififo
- También se crearán dos entradas /proc en \$PROC_DIRECTORY:
 - 1 "default": sustituye a la entrada que existía en el módulo original
 - Mismo comportamiento aunque distinta implementación
 - Tendrá asociada una lista enlazada o FIFO (depende de variante)
 - 2 "control": permite crear/destruir entradas /proc en \$PROC_DIRECTORY con misma semántica que la entrada "default" pero con sus propias estructuras asociadas (lista enlazada o fifo privados)
 - Crear entrada:

```
$ echo create my_entry > $PROC_DIRECTORY/control
```
 - Eliminar entrada existente (como "default")

```
$ echo delete my_entry > $PROC_DIRECTORY/control
```

Requisitos y recomendaciones para Parte A

- Cada vez que se cree una entrada /proc escribiendo en la entrada de control, se creará también una estructura privada asociada a la entrada (lista enlazada o FIFO)
 - Algunas variables globales existentes en el módulo original serán ahora miembros de una estructura a definir en el módulo
 - La entrada default tendrá asociada también su propia estructura privada
 - Las callbacks de la entrada “default” así como aquellas de las entradas creadas dinámicamente se implementarán con las mismas funciones
 - Mismas callbacks pero sobre distintos datos
 - No se permite replicar código
- Al eliminar una entrada, la memoria asociada a la estructura correspondiente debe liberarse

Requisitos y recomendaciones para Parte A

- Se recomienda mantener una estructura de datos (p. ej., lista enlazada) para llevar la cuenta de las entradas /proc y estructuras privadas creadas dinámicamente
- Al descargar el módulo, todas las entradas /proc creadas (incluido el directorio) deben destruirse y la memoria asociada a las estructuras asociadas debe liberarse
- La implementación del módulo del kernel debe ser SMP-safe
 - Se valorará la eficiencia/calidad de la implementación
 - Nota: `remove_proc_entry()` es una función bloqueante
 - Bloquea al flujo del kernel invocador hasta que la entrada /proc deja de estar en uso (en el caso de que lo esté)



Parte A (V1): Ejemplo de ejecución

Terminal

```
kernel@debian:~/Final/ParteAV1$ sudo insmod multilist.ko
kernel@debian:~/Final/ParteAV1$ ls /proc/multilist
control  default
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ echo add 10 > /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/default
10
kernel@debian:~/Final/ParteAV1$ echo add 4 > /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ echo add 4 > /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/default
10
4
4
kernel@debian:~/Final/ParteAV1$ echo add 2 > /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ echo add 5 > /proc/multilist/default
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/default
10
4
4
2
5
```



Parte A (V1): Ejemplo de ejecución (cont.)



Terminal

```
kernel@debian:~/Final/ParteAV1$ echo create list0 > /proc/multilist/control
kernel@debian:~/Final/ParteAV1$ echo create list1 > /proc/multilist/control
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/list0
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/list1
kernel@debian:~/Final/ParteAV1$ for i in `seq 0 5`
> do
> echo add $i > /proc/multilist/list0
> done
kernel@debian:~/Final/ParteAV1$ echo add 1 > /proc/multilist/list1
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/list0
0
1
2
3
4
5
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/list1
1
kernel@debian:~/Final/ParteAV1$ cat /proc/multilist/default
10
4
4
2
5
```



Parte A (V1): Ejemplo de ejecución (cont.)



Terminal

```
kernel@debian:~/Final/ParteAV1$ echo delete no_existe > /proc/multilist/control
bash: echo: error de escritura: No existe el fichero o el directorio
kernel@debian:~/Final/ParteAV1$ echo delete default > /proc/multilist/control
kernel@debian:~/Final/ParteAV1$ ls /proc/multilist/
control  list0  list1
kernel@debian:~/Final/ParteAV1$ sudo rmmod multilist.ko
kernel@debian:~/Final/ParteAV1$
```





Parte A (V2): Ejemplo de ejecución

terminal 1

```
kernel@debian:~/Final/ParteAV2$ sudo insmod multififo.ko
kernel@debian:~/Final/ParteAV2$ ls /proc/multififo/
control  default
kernel@debian:~/Final/ParteAV2$ cd ~/FifoTest/
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/default -s < test.txt
kernel@debian:~/FifoTest$
```

terminal 2

```
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/default -r
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo
que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y
galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches,
duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura
los domingos, consumían las tres partes de su hacienda. El resto della concluían
sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mismo,
los días de entre semana se honraba con su vellori de lo más fino. Tenía en su casa
una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y
un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera...
kernel@debian:~/FifoTest$
```





Parte A (V2): Ejemplo de ejecución (cont.)

terminal 1

```
kernel@debian:~/FifoTest$ echo create fifoA > /proc/multififo/control
kernel@debian:~/FifoTest$ echo create fifoB > /proc/multififo/control
kernel@debian:~/FifoTest$ ls /proc/multififo
control default fifoA fifoB
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/fifoA -s
Esto va por fifoA
kernel@debian:~/FifoTest$
```

terminal 2

```
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/fifoA -r
Esto va por fifoA
kernel@debian:~/FifoTest$
```

terminal 3

```
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/fifoB -s
Esto va por fifoB
kernel@debian:~/FifoTest$
```

terminal 4

```
kernel@debian:~/FifoTest$ ./fifotest -f /proc/multififo/fifoB -r
Esto va por fifoB
kernel@debian:~/FifoTest$
```



Parte A (V2): Ejemplo de ejecución (cont.)



terminal

```
kernel@debian:~/Final/ParteAV2$ echo delete no_existe > /proc/multififo/control
bash: echo: error de escritura: No existe el fichero o el directorio
kernel@debian:~/Final/ParteAV2$ echo delete default > /proc/multififo/control
kernel@debian:~/Final/ParteAV2$ ls /proc/multififo
control  fifoA  fifoB
kernel@debian:~/Final/ParteAV2$ echo delete fifoB > /proc/multififo/control
kernel@debian:~/Final/ParteAV2$ ls /proc/multififo
control  fifoA
kernel@debian:~/Final/ParteAV2$ sudo rmmod multififo
```



Parte A: Pistas

Creación y destrucción de directorios en /proc

- La función `proc_mkdir()` permite crear un directorio `name` en el sistema de ficheros `/proc`
 - Retorna descriptor del directorio creado

```
struct proc_dir_entry *proc_mkdir(const char *name,  
                                  struct proc_dir_entry *parent);
```

- El descriptor devuelto se pasa como tercer parámetro de la función `proc_create()` o `proc_create_data()` para crear una entrada `/proc` que “cuelge” del directorio creado
- Para destruir un directorio dentro de `/proc` es preciso utilizar `remove_proc_entry()`
 - El directorio ha de estar vacío para que la eliminación tenga éxito
- Se proporciona módulo de ejemplo `clipboard_dir.c` que crea entrada `/proc/test/clipboard`

Parte A: Pistas (cont.)

Asociar datos de uso privado (`private_data`) a entrada `/proc`

- Al crear una entrada `/proc` con la función `proc_create_data()` es posible asociar cualquier tipo de datos privados a la entrada `/proc`
 - `private_data` puede ser un puntero a un entero, un puntero a una estructura, una cadena de caracteres,...

```
struct proc_dir_entry *proc_create_data(const char *name,
                                         umode_t mode,
                                         struct proc_dir_entry *parent,
                                         const struct file_operations *ops,
                                         void *private_data);
```

- Desde el código de las callbacks de una entrada `/proc` podemos recuperar el campo `private_data` a partir del parámetro `struct file*` de la siguiente forma:

```
ssize_t my_read_callback (struct file *filp, char __user *buf, size_t len,
                           loff_t *off) {
    mi_tipo_de_datos* private_data=(mi_tipo_de_datos*)PDE_DATA(filp->f_inode);
    ....
}
```





Parte A: Pistas (cont.)

Asociar datos de uso privado (`private_data`) a entrada `/proc`

- Este mecanismo permite usar una misma función callback para distintas entradas `/proc`
 - Misma operación, pero sobre distintos datos ...



Práctica Final: Parte B



Parte B

- 1** Mostrar al profesor en el laboratorio el funcionamiento del módulo de la parte A de la práctica sobre la máquina virtual de Android-x86
 - Para los alumnos que realicen la variante 2 de la práctica ...
 - El binario de Android-x86 para el programa `fifotest` se puede construir desde la MV de Debian con el siguiente comando:

```
$ gcc -m32 -static -g -Wall fifotest.c -o fifotest
```
- 2** Desarrollar un programa de usuario en C para Android-x86 que interactúe mediante llamadas al sistema con el módulo del kernel desarrollado en la parte A
 - Adjuntar fichero `README` en la entrega con breve descripción sobre funcionalidad del programa desarrollado
 - Para los alumnos que realicen la variante 2 de la práctica, no basta con mostrar el programa `fifotest` funcionando en la MV de Android-X86





Práctica Final: programa parte B

Objetivo

- El objetivo del programa es demostrar que los servicios que exporta el módulo del kernel (p. ej.: entradas /proc) funcionan correctamente
 - Queda a elección del alumno el tipo de procesamiento que realice el programa de usuario
 - No se permite usar `system()` ni `exec()`

En esta parte se valorarán los siguientes aspectos:

- 1 Efectividad y completitud de los casos de uso/prueba del módulo llevados a cabo por el programa
- 2 Originalidad de la propuesta
- 3 Nivel de complejidad del programa C para Android-x86



Práctica Final: Parte B



Consideraciones adicionales

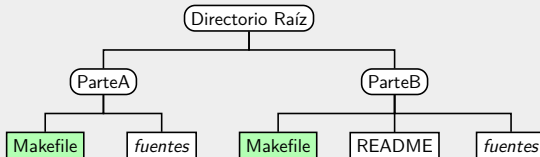
- La máquina virtual de Android lleva un kernel de 32 bits
 - El código del kernel (fork de Linux v3.10.52) puede consultarse a través de <https://srcxref.dacya.ucm.es>
- Recomendable depurar el programa de usuario en Linux y luego llevarlo a Android



Entrega de la práctica

- Entrega a través del Campus Virtual
 - Entrega hasta el **8 de febrero a las 16:50**
- Defensa individual obligatoria el 8 de febrero a las 17:00h en Lab 3
- Esta práctica es “opcional” (Máximo 2.5 puntos+)
 - Se permiten “entregas parciales” (p. ej., sólo un apartado)

Estructura entrega (en un fichero comprimido .tar.gz o .zip)





LIN - Práctica Final
Versión 0.2

©J.C. Sáez

*This work is licensed under the Creative Commons **Attribution-Share Alike 3.0 Spain License**. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/es/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.*

*Esta obra está bajo una licencia **Reconocimiento-Compartir Bajo La Misma Licencia 3.0 España de Creative Commons**. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.*

Este documento (o uno muy similar) está disponible en
<https://cv4.ucm.es/moodle/course/view.php?id=62472>

