PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
4ta práctica (tipo a)
(Primer semestre de 2022)

Horarios 0781, 0782

Duración: 1 h. 50 min.
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 1 (6 puntos – 30 min.) Su respuesta debe estar en la carpeta **INF239_078X_P4_P1_Buzón** de la **Práctica 4** en PAIDEIA **antes de las 09:40**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_41.txt`. Por ejemplo, `20171903_41.txt`.

(*By Michael Kerrisk*) "... In addition to maintaining a link count for each i-node, the kernel also counts open file descriptions for the file. If the last link to a file is removed and any processes hold open descriptors referring to the file, the file won't actually be deleted until all of the descriptors are closed. This is a useful feature, because it permits us to link a file without needing to worry about whether some other process has it open. (However, we can't reattach a name to an open file whose link count has fallen to 0). In addition, we can perform tricks such as creating an opening a temporary file, unlinking it immediately, and then continuing to use it within our program, relying on the fact that the file is destroyed only when we close the file descriptor – either explicitly, or implicitly when the program exits."

Ahora considere el siguiente programa:

```
$ cat t_unlink
#include <sys/stat.h>
#include <fcntl.h>
#include "tlpi_hdr.h"

#define CMD_SIZE 200
#define BUF_SIZE 1024

int
main(int argc, char *argv[])
{
    int fd, j, numBlocks;
    char shellCmd[CMD_SIZE];
    char buf[BUF_SIZE];

    if (argc < 2 || strcmp(argv[1], "--help") == 0)
        usageErr("%s temp-file [num-1kB-blocks] \n", argv[0]);

    numBlocks = (argc > 2) ? getInt(argv[2], GN_GT_0, "num-1kB-blocks")
                           : 100000;

    fd = open(argv[1], O_WRONLY | O_CREAT | O_EXCL, S_IRUSR | S_IWUSR);
    if (fd == -1) errExit("open");

    if (unlink(argv[1]) == -1) errExit("unlink");

    for (j = 0; j < numBlocks; j++)
        if (write(fd, buf, BUF_SIZE) != BUF_SIZE) fatal("partial/failed write");

    snprintf(shellCmd, CMD_SIZE, "df -k `dirname %s`", argv[1]);
    system(shellCmd);
    if (close(fd) == -1) errExit("close");
    printf("********** Closed file descriptor\n");
    system(shellCmd);
    exit(EXIT_SUCCESS);
}
```

```
$ ./t_unlink /tmp/tfile 1000000
Filesystem           1K-blocks       Used Available Use% Mounted on
/dev/sda10             5245020    3204044   2040976  62% /
********** Closed file descriptor
Filesystem           1K-blocks       Used Available Use% Mounted on
/dev/sda10             5245020    2201128   3043892  42% /
```

a) (3 puntos – 15 min.) Explique cuál es el objetivo del programa y comente las operaciones para lograr este objetivo durante su ejecución.

b) (3 puntos – 15 min.) Supongamos que el archivo `tfile` se crea en el sistema de archivo ext2 (un número de bloque en el *inode* ocupa 4 bytes) con los bloques de 4KB, ¿se usaría el bloque indirecto doble para este archivo? ¿El indirecto triple? Aproximadamente, ¿cuántos bloques administrativos se usarían para los bloques de datos del archivo `tfile`?

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 20.3 "Una"

Profesor del curso: V. Khlebnikov

Lima, 24 de junio de 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
4ta práctica (tipo a)
(Primer semestre de 2022)

Horarios 0781, 0782

Duración: 1 h. 50 min.
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 2 (6 puntos – 30 min.) Su respuesta debe estar en la carpeta **INF239_078X_P4_P2_Buzón** de la **Práctica 4** en PAIDEIA **antes de las 10:20**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_42.txt`. Por ejemplo, `20171903_42.txt`.

Un determinado sistema operativo gestiona la memoria virtual mediante paginación por demanda. La dirección lógica tiene 24 bits, de los cuales 14 indican el número de página. La memoria física tiene 5 marcos. El algoritmo de reemplazo de páginas es el LRU LOCAL (los marcos están asignados de forma **local a los procesos**, es decir, poseen una *Política de Reemplazo Local*: un grupo de marcos está dedicado a cada proceso y dentro de estos marcos del grupo se aplica el algoritmo, por proceso), y el algoritmo se ha implementado mediante un contador asociado a cada página que indica el instante de tiempo en que se referenció la página por última vez. Las tablas de páginas después del instante 15 son:

Tabla de páginas proceso A

	Marco	Bit de validez	Contador
0	1	v	10
1	2	v	15
2	-	i	6
3	-	i	5

Tabla de páginas proceso B

	Marco	Bit de validez	Contador
0	0	v	7
1	-	i	2
2	-	i	3
3	3	v	4
4	4	v	11

En los siguientes 10 instantes se generan las siguientes direcciones lógicas (todas las direcciones son decimales):

instante 16: (A, 2900)
instante 17: (B, 1200)
instante 18: (A, 1850)
instante 19: (A, 3072)
instante 20: (B, 527)
instante 21: (B, 2987)
instante 22: (A, 27)
instante 23: (A, 2000)
instante 24: (B, 4800)
instante 25: (B, 1500)

Indique las direcciones físicas (en decimal) generadas en estos instantes.

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 24 de junio de 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

4ta práctica (tipo a)
(Primer semestre de 2022)

Horarios 0781, 0782

Duración: 1 h. 50 min.
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 3 (8 puntos – 30 min.) Su respuesta debe estar en la carpeta **INF239_078X_P4_P3_Buzón** de la **Práctica 4** en PAIDEIA **antes de las 11:00**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_43.txt`. Por ejemplo, `20171903_43.txt`.

Raspberry Pi 3 Model B es una computadora de una sola placa de tamaño de una tarjeta de crédito. Su procesador es 1.2GHz 64-bit quad-core ARMv8, la memoria de 1GB, con salida HDMI, 4 puertos USB, Ethernet, WiFi, Bluetooth y MicroSD de almacenamiento. Y su precio es US$ 35. El sistema operativo principal es Raspbian, una versión de Debian, pero trabaja con muchos sistemas operativos tanto basados en Linux como en otros. En MicroSD viene la aplicación NOOBS (New Out of Box Software), la utilidad que facilita la instalación de diferentes sistemas operativos. Inicialmente MicroSD tiene una partición FAT32:

```
$ sudo fdisk /dev/mmcblk0
Disk /dev/mmcblk0: 29,3 GiB, 31444697088 bytes, 61415424 sectors
Sector size (logical/physical): 512 bytes / 512 bytes
Device         Boot Start      End  Sectors  Size Id Type
/dev/mmcblk0p1       8192 61415423 61407232 29,3G  c W95 FAT32 (LBA)
```

Pero con el primer encendido NOOBS instala Raspbian en MicroSD creando 5 particiones:

```
$ sudo fdisk /dev/mmcblk0
Disk /dev/mmcblk0: 29,3 GiB, 31444697088 bytes, 61415424 sectors
Sector size (logical/physical): 512 bytes / 512 bytes
Device         Boot   Start      End  Sectors  Size Id Type
/dev/mmcblk0p1         8192  2289062  2280871  1,1G  e W95 FAT16 (LBA)
/dev/mmcblk0p2      2289063 61415423 59126361 28,2G  5 Extended
/dev/mmcblk0p5      2293760  2359293    65534   32M 83 Linux
/dev/mmcblk0p6      2359296  2488319   129024   63M  c W95 FAT32 (LBA)
/dev/mmcblk0p7      2490368 61415423 58925056 28,1G 83 Linux
```

Ahora nos interesa la partición 5, la más pequeña, y haremos su *dump* desde el inicio (el *offset* `00000000` se indica en la 1ra columna):

```
$ sudo hexdump -C /dev/mmcblk0p5

00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|   Boot block
*
00000400  00 20 00 00 fc 7f 00 00  66 06 00 00 5f 76 00 00  |. ......f..._v..|   Superblock
00000410  f2 1f 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000420  00 20 00 00 00 20 00 00  00 08 00 00 5d 64 21 58  |. ... ......]d!X|
00000430  5d 64 21 58 05 00 ff ff  53 ef 01 00 01 00 00 00  |]d!X....S.......|
```

Después de **`Boot block`** (bloque #0), siempre con el *offset* `0x400` está el superbloque que tiene el número mágico en `0x438`, e indica (en `0x418`) que el tamaño de bloque es 2^0K = 1024 (`0x400`) bytes (1024 << log_block_size). La partición es de 32MB, o 65534 sectores de 512 bytes, por eso el superbloque indica (en `0x404: fc 7f 00 00`) que el sistema de archivo tiene `0x7ffc` (32764) bloques y (en `0x400`) tiene `0x2000` (8192) *i-nodes*.

Al superbloque sigue el bloque que contiene la tabla de descriptores de grupos de bloques (*block group descriptor table*) (el *offset* `0x800`):

```
00000800  82 00 00 00 86 00 00 00  8a 00 00 00 69 1b f2 07  |............i...|   Group
00000810  02 00 04 00 00 00 00 00  00 00 00 00 f2 07 d9 c8  |................|   descriptors
00000820  83 00 00 00 87 00 00 00  8a 01 00 00 7c 1f 00 08  |............|...|
00000830  00 00 05 00 00 00 00 00  00 00 00 00 00 08 16 d6  |................|
00000840  84 00 00 00 88 00 00 00  8a 02 00 00 00 1c 00 08  |................|
00000850  00 00 05 00 00 00 00 00  00 00 00 00 00 08 2c 1a  |..............,.|
00000860  85 00 00 00 89 00 00 00  8a 03 00 00 7a 1f 00 08  |............z...|
00000870  00 00 05 00 00 00 00 00  00 00 00 00 00 08 0d 78  |...............x|
00000880  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

que describe 4 grupos de bloques (en `0x800`, `0x820`, `0x840`, `0x860`) porque, según el superbloque, cada grupo de bloque es de 8192 bloques (`0x2000` en `0x420`). Las entradas (de 32 bytes cada una) de esta tabla indican que el *blocks bitmap* del grupo 0 está en el bloque # `0x82`, del grupo 1 está en el bloque # `0x83`, del grupo 2 está en el bloque # `0x84`, y del grupo 3 está en el bloque # `0x85`. El *inodes bitmap* del grupo 0 está en el bloque # `0x86`, del grupo 1 está en el bloque # `0x87`, del grupo 2 está en el bloque # `0x88`, y del grupo 3 está en el bloque # `0x89`. El *inodes table* del grupo 0 está en el bloque # `0x8a`, del grupo 1 está en el bloque # `0x18a`, del grupo 2 está en el bloque # `0x28a`, y del grupo 3 está en el bloque # `0x38a`.

Esto es el *blocks bitmap* del grupo 0:

```
000...00  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
000...90  ff ff 7f 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000...a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

(a) (3 puntos) ¿Cuál es el *offset* es este *bitmap*? (Realice y presente el cálculo en hexadecimal sin calculadora.)

(b) (3 puntos) Si el grupo 0 es de 8192 (`0x2000`) bloques, ¿cuántos bloques de este grupo están libres según su *bitmap*? (Realice y presente el cálculo en hexadecimal sin calculadora.) Este valor está indicado en el descriptor del grupo.

Esto es el *dump* de *inodes table* del grupo 0:

```
000...00  00 00 00 00 00 00 00 00  05 00 00 00 05 00 00 00  |................|
000...10  05 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000...20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000...80  ed 41 00 00 00 04 00 00  5d 64 21 58 df 02 00 00  |.A......]d!X....|
000...90  df 02 00 00 00 00 00 00  00 00 03 00 02 00 00 00  |................|
000...a0  00 00 08 00 07 00 00 00  0a f3 01 00 04 00 00 00  |................|
000...b0  00 00 00 00 00 00 00 00  01 00 00 00 8a 04 00 00  |................|
000...c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

(c) (2 puntos) ¿Cuál es el *offset* de esta tabla de *inodes*? (Realice y presente el cálculo en hexadecimal sin calculadora.)

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 24 de junio de 2022