

Sistemas Operativos

Departamento de Computación - FCEyN - UBA
Primer cuatrimestre de 2025

Nombre y apellido: Pedro Fuentes Uruguay

Nº orden: 11 L.U.: 1089/22 Cant. hojas: 1

1	2	3	Nota
20	20	30	70

(A)

Segundo parcial - 1er cuatrimestre de 2025

- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido, LU y número de orden. Hojas sin identificar no serán corregidas. Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas. Entregue esta hoja junto al examen, la misma no se incluye en la cantidad total de hojas entregadas. La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen dos notas: I (Insuficiente): 0 a 64 pts y A (Aprobado): 65 a 100 pts.

Ejercicio 1.(35 puntos)

Escribir el pseudocódigo de un programa que recorra todos los directorios en un sistema de archivos Ext2 y encuentre todos los i-nodos cuya cuenta de enlaces duros sea de dos o más. Para cada uno de estos archivos, debe listar todos los nombres de archivos que apuntan a él. Considerar que ningún directorio ocupará más de dos bloques, y que no pueden existir enlaces duros a directorios.

Se cuenta con la constante BLOCK_SIZE denotando el tamaño de un bloque de disco. También se cuenta con las siguientes estructuras y funciones ya implementadas:

```
struct Ext2FSDirEntry {  
    unsigned int inode;  
    unsigned short record_length;  
    unsigned char name_length;  
    unsigned char file_type; // d: directory, r: regular file, l: symbolic link  
    char name[];  
};  
struct Ext2FSInode {  
    unsigned short mode;  
    unsigned short uid;  
    unsigned int size;  
    unsigned int atime;  
    unsigned int ctime;  
    unsigned int mtime;  
    unsigned int dtime;  
    unsigned short gid;  
    unsigned short links_count;  
    unsigned int blocks;  
    unsigned int flags;  
    unsigned int os_dependant_1;  
    unsigned int block[15];  
    unsigned int generation;  
    unsigned int file_acl;  
    unsigned int directory_acl;  
    unsigned int faddr;  
    unsigned int os_dependant_2[3];  
};  
unsigned int get_block_address(struct Ext2FSInode * inode, unsigned int block_number);  
void read_block(unsigned int block_address, unsigned char * buffer);  
struct Ext2FSInode * load_inode(unsigned int inode_number);  
char * strncpy(char * dest, const char * src, size_t n);
```

Ejercicio 2.(35 puntos)

Se desea implementar el driver para una pantalla táctil. Un programa de usuario podrá comunicarse con este driver para conocer los eventos de toque que ocurrán en la pantalla, y así identificar distintos gestos. Un gesto comienza cuando se toca la pantalla táctil, dura todo el tiempo que se mantiene el contacto, y termina al dejar de tocar.

El dispositivo genera una interrupción IRQ_TOUCH cuando detecta un toque en la pantalla (al iniciar un gesto) y actualiza automáticamente los registros TCH_COORD con un entero que representa las coordenadas (x, y) del contacto, TCH_STATUS con el valor TCH_PRESS y TCH_TIME con el valor 0. Cuando el gesto termina, se actualizan *muy rápidamente* los registros TCH_TIME con un valor numérico que representa la cantidad de milisegundos que duró el gesto y TCH_STATUS con el valor TCH_UP. Mientras dura el gesto, el registro TCH_COORD es actualizado con los cambios de ubicación del toque y TCH_TIME es actualizado con la cantidad de tiempo transcurrido desde que se inició el gesto.

Cuando una aplicación haga uso del input de la pantalla táctil, deberá poder identificar si se trata de un click, un deslizamiento, o una pulsación prolongada. El driver deberá suministrar la información necesaria para ello.

Además, la pantalla permite colorear pixeles utilizando los registros PRINT_COORD para la coordenada del pixel, PRINT_COLOR para el color a utilizar y PRINT_OK para efectuar el coldeo al escribir el valor PRINT.

- Escribir las funciones necesarias del driver para lograr el funcionamiento pedido. Tener en cuenta que la lectura debe ser bloqueante mientras no esté ocurriendo ningún gesto.
- Escribir el pseudocódigo de un programa de usuario que se comunique con el driver para detectar los siguientes patrones de gestos:
 - **Click:** se detecta un único punto de contacto sin movimiento y con una duración muy corta (menos de 500 ms).
 - **Deslizamiento:** se detecta un punto de contacto inicial y un movimiento continuo (sin soltar).
 - **Pulsación prolongada:** se detecta un único punto de contacto sin movimiento y con una duración mayor al click.

Al detectar un click, se debe colorear el pixel tocado en color AZUL. Si se trata de una pulsación prolongada, será de VERDE. En caso de detectar un desplazamiento, cada punto detectado debe ser coloreado de ROJO.

Se cuenta con las funciones `int getX(int coord)` y `int getY(int coord)` que permiten obtener las coordenadas x e y respectivamente a partir del valor que se encuentra en TCH_COORD. A su vez, la función `int genCoord(int x, int y)` toma dos coordenadas y genera el valor necesario para el registro PRINT_COORD.

Ejercicio 3.(30 puntos)

Dado el siguiente código:

- Indicar qué vulnerabilidad tiene. Mostrar un ejemplo de entrada que resulte en un resultado inesperado, y un ejemplo de entrada que rompa la ejecución del programa.
- Proponer una solución posible para la vulnerabilidad indicada.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    int valor = 5;
    char buffer_uno[8], buffer_dos[8];

    strcpy(buffer_uno, "uno");
    strcpy(buffer_dos, "dos");

    printf("[ANTES] buffer_dos está en %p y contiene \'%s\'\n", buffer_dos, buffer_dos);
    printf("[ANTES] buffer_uno está en %p y contiene \'%s\'\n", buffer_uno, buffer_uno);
    printf("[ANTES] valor está en %p y es %d (0x%08x)\n", &valor, valor, valor);

    printf("\n[STRCPY] copiando %d bytes en buffer_dos\n", strlen(argv[1]));
    strcpy(buffer_dos, argv[1]);

    printf("[DESPUÉS] buffer_dos está en %p y contiene \'%s\'\n", buffer_dos, buffer_dos);
    printf("[DESPUÉS] buffer_uno está en %p y contiene \'%s\'\n", buffer_uno, buffer_uno);
    printf("[DESPUÉS] valor está en %p y es %d (0x%08x)\n", &valor, valor, valor);
}
```

1)

~~Void ej1 () {~~~~char *current;~~~~Dentry *to-visit = DentryList (loadMode(2)); // 2 nodes.~~~~while (to-visit.length >= 1) {~~~~extern Dentry~~~~current = to-visit.pop();~~~~if (current->dev~~~~addEntries To (DentryList + to-add, EX2FS_INODE * mode) {~~~~EX2FS_Dentry * current = (EX2FS_Dentry *) malloc (2 * BLOCK_SIZE);~~~~current->addr~~~~readBlock (getBlockAddress (mode, modeBlock[0]), current);~~~~readBlock (getBlockAddress (mode, modeBlock[1]), current + BLOCK_SIZE);~~

↳ ¿Y si ocupa 1 solo bloque?

~~while (current->file-type == "d" || current->record-length == 0) {~~~~if ((current->file-type == "d") ||~~~~strcmp (current->file-type, "d") == 0) {~~~~TO-ADD.append (current->entry);~~~~TO-ADD.append (CURRENT-DENTRY);~~~~}~~~~current += current->record-length; }~~

}

→ si la dentry acaba en "null"; especificar ya terminado

No es válido asumir

que el resto del bloque es cero. inode.size te dice la suma total de todos los dirEntries.

void EJ1() {

Dentry LSL * To-visit = Dentry LSL(); // Inicializa una lista vacía
add_dentries_To (To-visit), load-mode (2);
while (! To-visit->empty) {

EXT2FS_DirEntry * current-dentry = To-visit.pop();

EXT2FS_Imode * current-dentry-mode = lnd-mode (current-dentry->mode);

~~if (current-dentry->file_type == 1)~~

~~if (STRNCMP (current-dentry->file_Type, "R", 1) == 0)~~
~~current-dentry-mode->links_count = 1;~~

// Si es un modo con 2 o más referencias, busca sus archivos

read-name-fn (current-dentry->mode); // Para el resto de modos

{ else if (STRNCMP (current-dentry->file-Type, "d", 1)) {

search_dentries (current-dentry, To-visit);

}

}

```
void search_directory(EXT2FS_DirEntry *current_dir, DirectoryList *to-add) {
```

EXT2FS mode * prev-mode = load-mode (prev-dir -> mode);

EXT2FS_DirEntry * current = (EXT2FS_DirEntry *) malloc (2 * BLOCK_SIZE);

read_block (get-block-address (prev-mode, prev-mode -> block[0]), current);

read-block (get-block-address (prev-mode, prev-mode -> block[1]), current);

```
while (current < 2 * BLOCK_SIZE && current -> record-length != 0) {
```

// Si es directorio, lo agrega a to-add. Si archivo, lo agrega a queue.

```
if (strcmp (current -> file-type, "d", 1)) {
```

to-add.append (current);

```
else if (strcmp (current -> file-type, "r", 1)) {
```

EXT2FS mode * curr-mode = load-mode (current -> mode);

if (curr-mode -> links-count > 1) {

find-nodes-from (current -> mode);

current += current -> record-length; add-dentry-to() y ej().

Implementaste otra vez

curr += curr -> record-length; add-dentry-to() y ej().

Podrias reutilizar las

funciones que ya tienes.

```
void FIND-NAMES-FROM (unsigned int target) {
```

DirectoryList * to-vist = DirectoryList();

add-directory-to (to-vist, load-mode(2));

```
while (!to-vist.empty()) {
```

EXT2FS_DirEntry * current = to-vist.pop();

```
if (strcmp (current -> file-type, "r", 1) && current -> mode == target) {
```

printf ("%s", current -> name); // printea solo esos bytes, es COMPATNF?

// Asumo que hay una función que lo hace.

// Si queremos más info.

else if (STRENCMP(curread->file-type, "d", 1)) {

// Quiero las todas sus direcciones. Alos archivs tienen el tipo de

// A los directorios agregarlos a TO-VISIT.

EXT2FSNODE * curread = head_node(curread->node);

~~Estas repiten~~ EXT2FSDIRENTRY * data = (EXT2FSDIRENTRY *) malloc(2 * sizeof(struct dirent));

Muchó código. read_block(get_block_address(~~ext2fs_node~~, curread->node, block[0]), data);
read_block(get_block_address(curread->node, curread->node->block[1]), data + sizeof(struct dirent));

while (data[0] <= 2 * sizeof(struct dirent) && data[0] >= read_length) {

if

if (STRENCMP(data->file-type, "d", 1)) {

TO-VISIT_APPEND(data);

~~else if (STRENCMP(data->file-type, "r", 1)) {~~

~~else~~

~~EXT2FSNODE * data_node = head_node(file_node);~~

~~data_node->~~

~~if (STRENCMP(data->file-type, "r", 1) && data->node == target) {~~

~~PRINTNF("%s", data->name-length, data->name);~~

~~}~~

~~data += data->record->length;~~

La idea está bien pero es una mala implementación.

Me hubiese gustado que detectes que estás escribiendo lo mismo 3 veces y que hayas buscado una solución más elegante y concisa.

~~Al final~~

1) Pido disculpas por lo desordenado. No tuve tiempo de pararlo en bruto.

También por el código repetido.

2) Estas líneas deberían sumar el resto de bytes. Es decir, algo así.

PRINTNF("%d %s", data->name-length ^{+ sizeof(struct dirent)}, data->name, data->name);

L, f

OCCURRÉ EN EL CASO DE ②

2) Para emplear, asumir que hay un solo usuario, por lo tanto no habrá condiciones de carrera.
~~void driver_init()~~

~~L, (se lo expusiste)~~

negro - ira (JRD+TOUCH, handler_init(), etc);

SEM_C SEM-GESTO = SEM(); → esto tiene que definirlo
~~desconocido~~ global en el driver.

Falta driver_remove

void HANDELR_INICIO_GESTO (INT_SIE) {

// Asumo que TCH_STATUS andarán entre TCH_UP y TCH_PRESS.

Cuidado! Si hay actividad en la
 puerta ANTES de emplear el
 SEM-GESTO.SIGNAL();

Muy vacío de recursos, podrías ocurrir
 que se quede pue hasta el fin del pos o no
 que corresponda.

// EL READING DE 1 COORDENADA

INT DRIVER_READ (ML & data_user) {

// Si no hay un gesto en curso, error.

If (IN(TCH_STATUS) == TCH_UP) {

* SEM-GESTO.WAIT(); // Bloqueante!

// Así ya habrá sido leído en gesto, pero sigue.

If (IN(TCH_STATUS) == TCH_PRESS) {

ML * data_kernel = (ML *) malloc (4 * sizeof (ML)); // Devuelve X,y, Encap, UP

ML coord = IN(TCH-COORD);

ML time = IN(TCH-TIME);

ML x = getX(coord);

ML y = getY(coord);

memcpy (data_kernel, &x, sizeof (ML));

// return

y es para cargar memoria
a kernel.

```
memory( data-kernel + syloff(w), &y, sizeoff(w) ); // Copia solo "syloff" bytes.  
memory( data-kernel + 2 * sizeoff(w), &true, sizeoff(w) ); //  
Copy - To - user ( data-kernel, data-user, sizeoff(w) + 4 ); // Ahora se la copia al user  
return 1; → debemos reponer 4 + sizeoff(w)
```

// Si la operación es pospuesta irá al final en TCH_STATUS
return 0;

// El write es de a 1 coordenada.

```
void driver - write( int * data-user value ) {  
    int x = data-kernel[0];  
    int y = data-kernel[1];  
    int color = data-kernel[2];  
    int coord = genCoord(x,y);  
    Copy - FROM - USER( DATA - KERNEL, DATA - USER, 3 * sizeoff(w) );
```

int x = data-kernel[0];

int y = data-kernel[1];

int color = data-kernel[2];

int coord = genCoord(x,y);

OUT(PRINT - COORD, &coord);

OUT(PRINT - COLOR, &color);

OUT(PRINT - OK, &PRINT); // VAR. GLOBAL

~~KFREE(DATA - KERNEL);~~

② Acá me faltó devolverle al usuario que el driver terminó. Sera lo mismo que
en ① y cuando ② ~~TCH STATUS~~ ~~TELL 2000~~ en el "STATUS"

③ Lo mismo. Pusola para 1 en el "STATUS"

Pedro Fuentes Orfey

20/8/22

Hojas.

47

CÓDIGO DE USUARIO

(PERDÓN POR LA DESPROLIFICACIÓN, LLEGUÉ MÁS CON EL TIEMPO Y ME DI CUENTA DE COSAS MUY AL FINAL)

$$N = 3 * \text{sizeof}(w)$$

INT MAIN () {

INT FD-DEV = OPEN ("DEV/PANTALLA");

INT * DATA-READ = (INT *) MALLOC (SIZEOF N);

INT * DATA-WRITE = (INT *) MALLOC (N); INT curr-color = 0;

INT PREV-X = 0; INT PREV-Y = 0; INT PREV-TIME = 0; ^{int zero_en-curso_f0} ~~int zero_en-curso_f0~~

WHILE (TRUE) {

READ (FD-DEV, & DATA-READ);

INT X = DATA-READ[0]; INT Y = DATA-READ[1]; INT TIME = DATA-READ[2];

DATA-WRITE[0] = X; DATA-WRITE[1] = Y;

DATA-WRITE[2] = curr-color;

curr-color =

entiendo que
DATA-READ[4] es
el slotus.

~~if (gesto_en_curso == 1 & DATA-READ[4] == 1) {~~
~~// No es el gesto levor. Solo el cursor si era un slotus desplazamiento~~
~~// DATA-WRITE(2) = DATA-WRITE(1); write(FD-DW, &DATA-WRITE[1]);~~
~~} else {~~
~~// No es desplazamiento. Espero a que termine~~
 }
else if (gesto_en_curso == 0 & DATA-READ[4] == 1) {
 // Empieza un gesto
 // Si los coords son distintos, es desplazamiento.
 if (prev-x != x || prev-y != y) {
 // am_color = ROJO;
 // gesto_en_curso = 1; DATA-WRITE(2) = am_color; while (FD-DW, DATA-WRITE[2])
 // el cursor no recuerda las direcciones
 }
 else if (gesto_en_curso == 1 & DATA-READ[4] == 0) {
 // Termina el gesto! Vlo fuera de los tres orde
 if (Reto < 500) {
 // DATA-WRITE(2) = AZUL
 // DATA-WRITE(FD-DW, &DATA-WRITE[1]);
 // (prev-x != x) || (prev-y != y)
 }
 if (am_color == ROJO) { // Vla desplazamiento
 DATA-WRITE(2) = ROJO;
 }
 else if (Reto < 500) {
 DATA-WRITE(2) = AZUL;
 }
 else {
 DATA-WRITE(2) = VERDE;
 }
 }
}
N Sigue...

Pedro Fuentes Ulofberg

1088172

Hoja 6

11

// ACTUALIZO VARIABLES

PREV-X = X;

PREV-Y = Y;

PREV-TIME = TIME;

~~if (TIME > PREV-TIME) {~~

cero_en_censo = DATA-ROBO[4];

return 0;

De vuelta al trabajo por el quinto...

- ④ La primera vez que lee prev-x y prev-y están en 0, va a considerar que es el punto (0,0) en parte de un desplazamiento.

30/30

Pedro Fuentes Uffey

1088122

Hoja 7

27

3)a) La vulnerabilidad está en la linea `strcpy(buffer-dos, argv[1]);`
 Esto puede resultar en un buffer overflow ya que en ningún momento se checa cuál es la longitud del buffer pasado por parámetro.

Vamos un ejemplo que resulte en un resultado inesperado. Supongamos que "uno" esté en algo distinto de 5, o que "buffer-uno" esté en algo distinto de "uno". Podríamos lograr esto pasando un string de más de 8 caracteres por parámetros como "BUFFER2CAMBiADO" (tienen 13 caracteres). Luego, buffer-uno tendría algo distinto a "uno". Esto lo veremos con la pila del programa, que se ve así luego de declarar las variables ✓

RETURN ADDRESS	
VALOR [7]	
VALOR [0]	
BUFFER-UNO [7]	
: BUFFER-UNO [0]	
BUFFER-DOS [7]	
: BUFFER-DOS [0]. en	

Uffey

Luego, para hacer algo que rompa la ejecución podríamos escribir los siguientes caracteres (uno para plan la return address). Podría poner algo que se interprete como una dirección inválida (o alguna en la cual yo no tenga acceso) y me saldría con segmentación fail. Entonces, un string de 20 bytes de longitud (algunas veces suficiente (20 bytes de los variables + 8 de la return address, ansiendo que denga una dirección)) ✓