INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

SISTEMAS OPERATIVOS

David Díaz Araujo

Práctica 11

Administración de dispositivos de E/S

- Alumnos:
  1) Arcos Hermida Aldo Alejandro (
  2) Chávez Becerra Bianca Cristina
  3) Islas Osorio Enrique (N° lista 2(
  4) Juárez Cabrera Jessica (N° lista
  5) Palmerin García Diego (N° lista
- Grupo: 4CM1
- Materia: Sistemas operativos
- Profesor: Araujo Díaz David

Índice

Objetivo

Describir como se administran los dispositivos de Entrada/Salida en los sistemas operativos LINUX/UNIX.

Descripción

En Linux todos los dispositivos de E/S se deben ver como archivos y se deben utilizar con las mismas llamadas a sistema *open*, *read* y *write* que se utilizan para acceder a todos los archivos ordinarios. Sin embargo, los dispositivos se integran al sistema de archivos como *archivos especiales*. A cada dispositivo de E/S le asigna un nombre de ruta, generalmente en */dev*. No se requiere ningún mecanismo especial para realizar operaciones de E/S. Linux divide estos archivos en dos categorías: tres clases: dispositivos de bloque, dispositivos de caracteres y dispositivos de red.

Los archivos de bloque consisten en una secuencia de bloques enumerados que se pueden direccionar y utilizar por separado, suele utilizarse para discos. Los archivos especiales de caracteres se utilizan para dispositivos que reciben o envían un flujo de caracteres, como teclados, impresoras, redes, etc.

A cada archivo especial se le asocia un driver de dispositivo que se encarga de manejar el dispositivo correspondiente. Cada driver tiene un número de dispositivo mayor con el cual se identifica. Si un driver acepta varios dispositivos, cada uno tiene un número de dispositivo menor que lo identifica. Estos números en conjunto, especifican cada uno de los dispositivos de E/S en forma única.

Como se mencionó anteriormente, cada dispositivo de E/S en un sistema Linux tiene asociado un archivo especial y se implementa mediante una colección de drivers de dispositivos. Los drivers aíslan al resto del sistema del hardware, se dividen en dos partes que forman parte del kernel de Linux y se ejecutan en modo kernel.

La administración de dispositivos, es la administración de todos los recursos del hardware disponible, tanto los estándar que viene de fábricas, como las que se van agregando para hacer más poderosa o actualizar la PC. Todo dispositivo necesita presentarse al sistema operativo, agregando un pequeño programa que permite su uso. Este hp programa es llamado controlador. De aquí el controlador es un software que utiliza el sistema operativo para especificar de hardware, como puede ser cualquier dispositivo.

Los puntos de vista respecto al hardware de E/S son muy distintos para cada persona, dependiendo del campo en el que trabaje. Los ingenieros eléctricos lo ven en términos de chips, alambres, fuentes de potencia, motores y todos los demás componentes físicos que constituyen el hardware. Los programadores tienen en cuenta la interfaz en relación con el software: los comandos que el hardware acepta, las funciones que realiza y los errores que puede informar. En este libro nos interesa la programación de los dispositivos de E/S, no su diseño, construcción ni mantenimiento, así que nuestra atención se limitará a la forma como el hardware se programa, no a cómo funciona internamente.

Los dispositivos de E/S se pueden dividir a grandes rasgos en dos categorías: dispositivos por bloques y dispositivos por caracteres. Un dispositivo por bloques almacena información en bloques de tamaño fijo, cada uno con su propia dirección. Los tamaños de bloque comunes van desde 512 bytes hasta 32 768 bytes. La propiedad esencial de un dispositivo por bloques es que es posible leer o escribir cada bloque con independencia de los demás. Los discos son los dispositivos por bloques más comunes. Si lo analizamos con cuidado, la frontera entre los dispositivos que son direccionales por bloques y los que no lo son no se hallan bien definida.

En teoría los dispositivos de e/s se comunicarían con la CPU por los buses del sistema

Dado que son muy heterogéneos sería muy costoso que la CPU los manejase directamente

Los dispositivos están conectados a una pieza de hardware llamada controlador de dispositivo (a veces controladora o adaptador)

El controlador de dispositivo admite comandos abstractos de la CPU y se encarga de transmitirlos al dispositivo

Se libera a la CPU de tareas de muy bajo nivel

La comunicación con el sistema UNIX se da mediante un programa de control llamado SHELL. Este es un lenguaje de control, un intérprete, y un lenguaje de programación, cuyas características lo hacen sumamente flexible para las tareas de un centro de cómputo. Como lenguaje de programación abarca los siguientes aspectos:

- Ofrece las estructuras de control normales: secuenciación, iteración condicional, selección y otras.

- Paso de parámetros.

- Sustitución textual de variables y Cadenas.

- Comunicación bidireccional entre órdenes de shell.

El shell permite modificar en forma dinámica las características con que se ejecutan los programas en UNIX:

Las entradas y salidas pueden ser redireccionadas o redirigidas hacia archivos, procesos y dispositivos;

Es posible interconectar procesos entre sí.

Prueba de escritorio

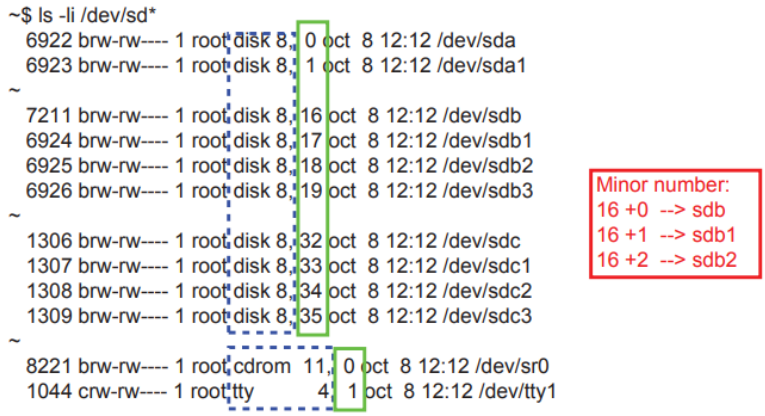
En UNIX los dispositivos aparecen como un fichero (típicamente en el directorio /dev). En algunos sistemas fichero es un enlace simbólico a donde está realmente el fichero del dispositivo.

Tienen asignado un índice, en donde además de indicar si es un dispositivo de bloque o de carácter, figuran dos números (mayor número y menor número) que indican que manejador de dispositivo se utiliza para acceder a ´él y que unidad dentro de las manejadas por dicho manejador.

```
# ls -li /dev/sda /dev/sda1 /dev/sda2 /dev/audio
4232 crw-rw---T+ 1 root audio 14, 4 Jan  7 18:51 /dev/audio
3106 brw-rw---T  1 root disk   8, 0 Jan  7 18:51 /dev/sda
3110 brw-rw---T  1 root disk   8, 1 Jan  7 18:51 /dev/sda1
3111 brw-rw---T  1 root disk   8, 2 Jan  7 18:51 /dev/sda2
```

Para acceder a los dispositivos se pueden utilizar las mismas llamadas que para el acceso a los ficheros (open, read, write) siempre que el proceso que lo haga tenga los privilegios adecuados.

ls -li /dev/...
Shows permissions (brw-rw----), owner (root), group (disk), major device number (8), minor device number (0), date (mars 9 - french, no year), hour (07:56) and device name (guess :-).
When accessing a device file, ...
... the major number selects which device driver is being called to perform the input/output operation. This call is being done with the minor number as a parameter and it is entirely up to the driver how the minor number is being interpreted. The driver documentation usually describes how the driver uses minor numbers.



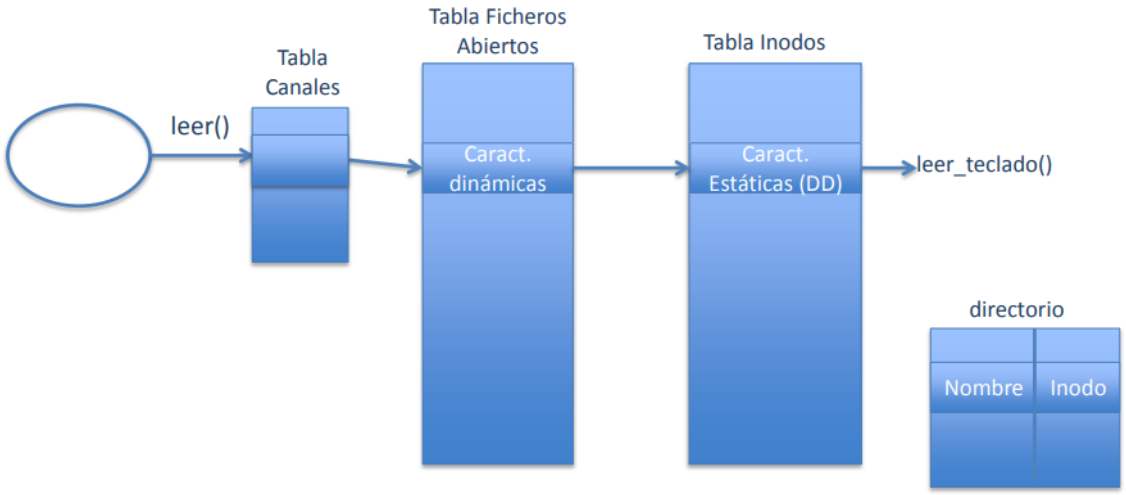


Tabla Ficheros Abiertos
Tabla Canales
Tabla Inodos
leer()
Caract. dinámicas
Caract. Estáticas (DD)
leer_teclado()
directorio
Nombre
Inodo

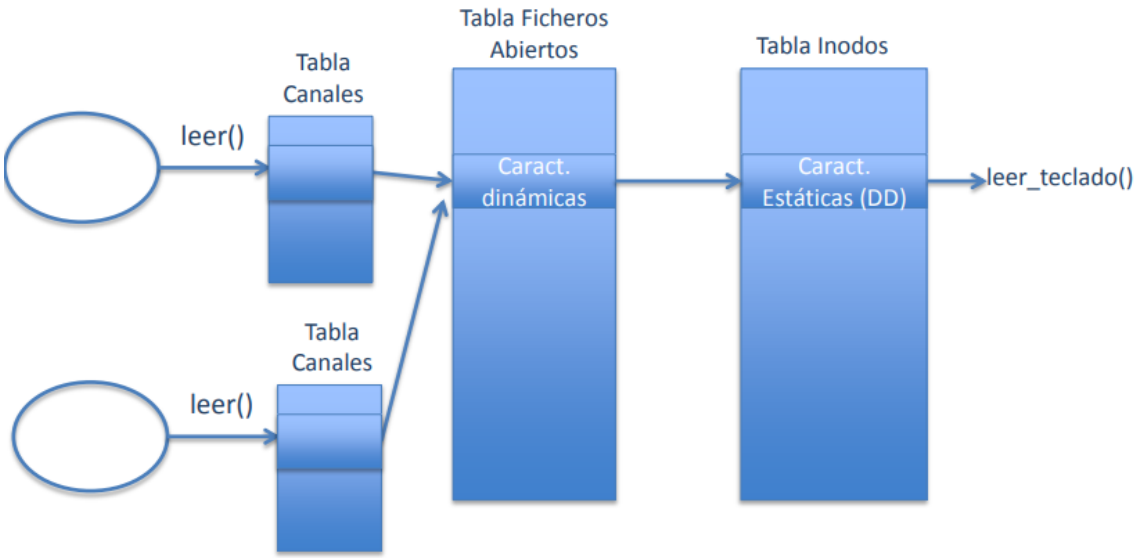
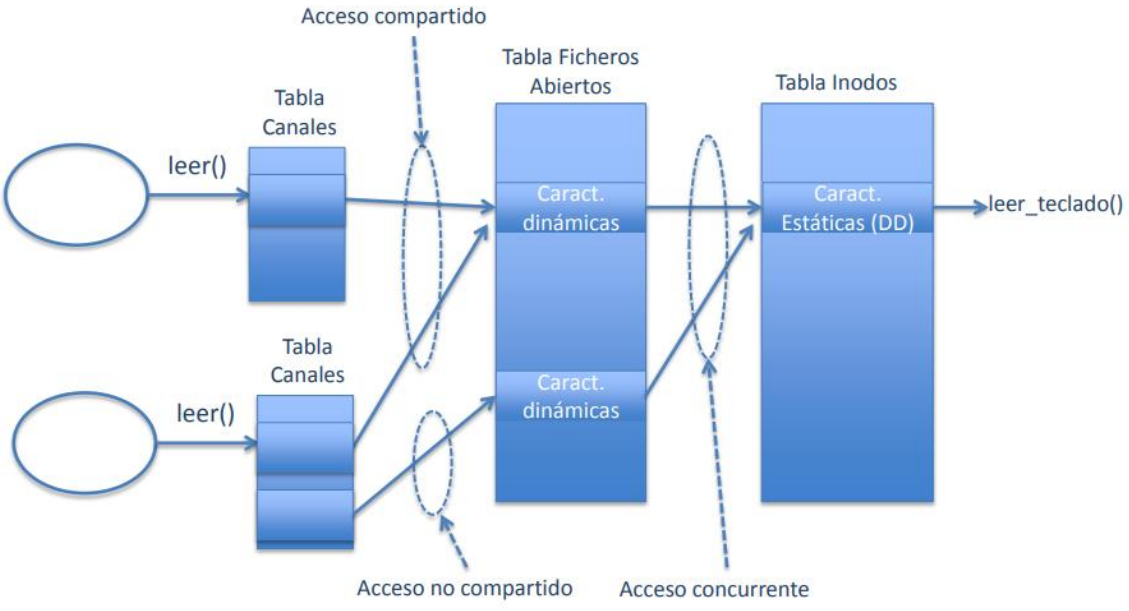


Tabla Ficheros Abiertos
Tabla Canales
Tabla Inodos
leer()
Caract. dinámicas
Caract. Estáticas (DD)
leer_teclado()
Tabla Canales
leer()


Acceso compartido
Tabla Ficheros Abiertos
Tabla Canales
Tabla Inodos
leer()
Caract. dinámicas
Caract. Estáticas (DD)
leer_teclado()
Tabla Canales
leer()
Caract. dinámicas
Acceso no compartido
Acceso concurrente

Dispositivos lógicos accesibles a través del Sistema de ficheros

– Ficheros especiales (normalmente situados en /dev)

• /dev/hda1

• /dev/audio0

• /dev/nul

– Se utilizan con las primitivas normales

(open,read,write,...)

• Se crean mediante mknod

– Asigna dos numeros especiales al fichero: major y minor

• Relaciona dispositivo lógico con dispositivo físico

– Asigna el tipo de entrada/salida: por bloques o por carácteres

Código fuente

```java
public class Mostrarinformacion {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        caracteristicas objeto=new caracteristicas();
        objeto.setVisible(true);
    }
}
```

```java
package callthesystem;

public class Callthesystem {

    public static void main(String[] args){
        // TODO code application logic here
        try{
            String []cmd={"shutdown","-s","-t","10"};
            //Runtime.getRuntime().exec(cmd);
        }catch(Exception e){
            System.out.println("error "+e);
        }

        leeryguarda conector=new leeryguarda();
        conector.setVisible(true);
    }
}
```

```java
package callthesystem;

public class Callthesystem {

    public static void main(String[] args){
        // TODO code application logic here
        try{
            String []cmd={"shutdown","-s","-t","10"};
            Runtime.getRuntime().exec(cmd);
        }catch(Exception e){
            System.out.println("error "+e);
        }
```

Prueba de pantalla

```
debby@debian:~/Desktop$ touch file.txt
debby@debian:~/Desktop$ ls
entsal.pdf  file.txt
debby@debian:~/Desktop$ sudo mv file.txt /dev/null
debby@debian:~/Desktop$ ls
entsal.pdf
debby@debian:~/Desktop$ cd /dev/null
bash: cd: /dev/null: Not a directory
```

Conclusiones

Una de las características principales de un SO es el control de todos los dispositivos de E/S de la computadora, esto comprende desde la transferencia entre diversos niveles de memoria hasta la comunicación entre los periféricos, además de que la interfaz que presente sea sencilla y fácil de implementar. El SO tiene la obligación de controlar todos los dispositivos E/S para cumplir con objetivos fundamentales para su uso.

Los dispositivos E/S abarcan un considerable rango de velocidades, por esta razón se le impone una presión considerable al software con el fin recibir un buen desempeño sobre muchas ordenes de magnitud.

Referencias

Área de memoria compartida Posix - programador clic. (s. f.). programerclick. Recuperado, de https://programmerclick.com/article/73871146075/

M. D. L. L. R. MARTINEZ, SISTEMAS OPERATIVOS, U.N.N.E ARGENTINA, 2015.

Silberschatz, A. G. (2005). *Fundamentos de sistemas operativos. Séptima edición.* Madrid: McGraw-Hill.

Stallings, W. (2005). *Sistemas operativos: Aspectos internos y principio de diseño. Quinta edición. .* Madrid: Pearson Educación.

Tanenbaum, A. S. (2009). *Sistemas Operativos Modernos. Tercera edición.* México: Pearson Educación.