programa de BECAS
DGTIC
de FORMACIÓN en TIC

Seguridad Informática

Lenguaje C y llamadas al Sistema.

Fernando C.

Temario.

- ***Manejo de archivos.***
  - ***Flujos.***
  - ***Descriptores de archivos.***
- ***Uso de bibliotecas externas.***
  - ***Matemáticas con C.***
  - ***Networking con C.***
  - ***Cifrado con C.***
  - ***Bases de datos con C.***
  - ***Interfaces graficas con C.***
- Interacción de C con el sistema operativo.
- Clasificación de llamadas al sistema.
- Implementación de llamadas al sistema.
- Módulos de kernel.
- “Programación segura”.

Manejo de archivos.

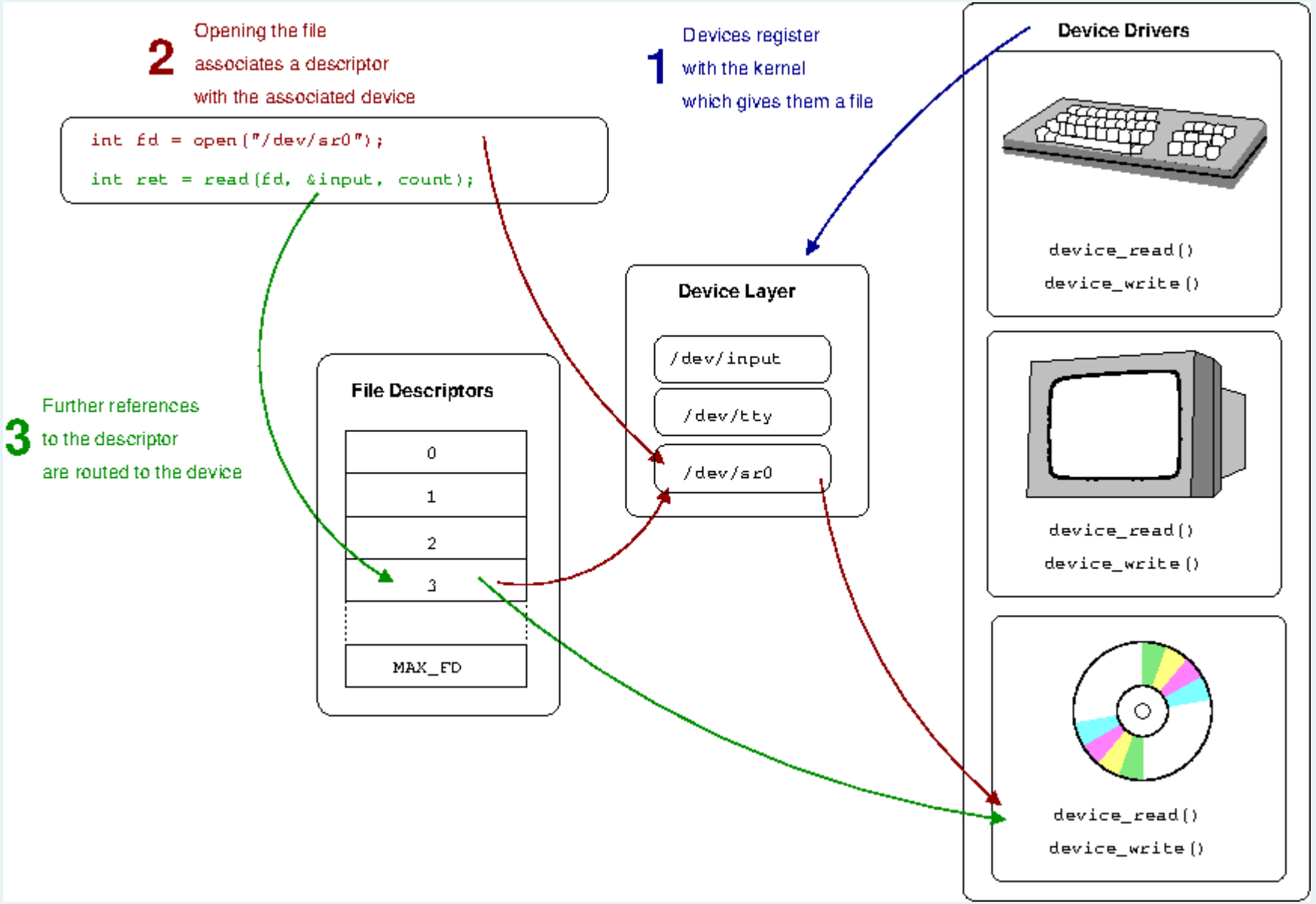
- Existen dos formas de acceder a un archivo, por medio de flujos y por medio de descriptores de archivos.

- Flujos se puede llevar a cabo importando la biblioteca file.h y utilizando sus funciones. Pero esto se puede encontrar muy fácilmente en cualquier tutorial en línea.

- Para este curso y para comenzar con llamadas al sistema, emplearemos descriptores de archivos.

¿Qué es un descriptor de archivo?

- Generalmente son usados en sistemas operativos POSIX(UNIX, BSD, GNU/Linux).
- Un descriptor de archivo es un concepto propio de un sistema operativo y se define como “Un file descriptor es un entero provisto por el kernel, usualmente no muy alto, que representa algo a lo que se puede mandar bytes, o desde donde se puede leerlos.”


2
Opening the file
associates a descriptor
with the associated device
int fd = open("/dev/sr0");
int ret = read(fd, &input, count);
1
Devices register
with the kernel
which gives them a file
Device Drivers
device_read()
device_write()
Device Layer
/dev/input
/dev/tty
/dev/sr0
File Descriptors
0
1
2
3
MAX_FD
3
Further references
to the descriptor
are routed to the device
device_read()
device_write()
device_read()
device_write()

Funciones de lectura para trabajar con file descriptors.

- open()
- close()
- read()
- write()

- man 2 open

- http://fcastaneda.herokuap.com/c/dia4.zip

Practica 14. Números aleatorios.

Hay dos formas de generar números aleatorios, como los egresados de alguna licenciatura o como los integrantes de PBSI.

Recuerda que todo es un archivo, lee los bytes necesarios del "device" /dev/random en tu equipo, para generar un numero aleatorio, que pueda ser alojado en un entero.

La función read() recibe descriptor de archivo, dirección de la variable en la que se almacenaran los bytes leídos, y los bytes a leer.

Good luck.

Si quisiéramos especificar los bytes que leerá read()

```
off_t fsize;

fsize = lseek(fd, 0, SEEK_END);

char * tmp = (int *)malloc(fsize);

read(fd,&tmp,fsize);
```

Pero pueden utilizar las funciones estándar, por comodidad, afortunadamente alguien creo fopen()

Bases de datos.

sqlite3.h (?) -> apt-get install libsqlite3-dev sqlite3

http://zetcode.com/db/sqlitec/

```
#include <sqlite3.h>
#include <stdio.h>
int main(void) {
    printf("%s\n", sqlite3_libversion());
}
```

Practica 15. Recuerdas tu base de becarios?

Utiliza tu practica de la base de datos de becarios, cuya salida redirigías a un archivo, pero ahora crea una base de datos real.

El diseño no es importante, pero la tengo que poder consultar por medio de un cliente.

"Criptografia, cifrados y demas..."

- openssl
- heartbleed
- -lcrypt
- Md5.c

Practica 16. Hash, hash, hash...

Modifica el archivo md5.c para que pueda obtener el hash md5 de un archivo pasado como argumento por línea de comandos.

Sockets

Es una abstracción de software de algo que representa un punto final entre la conexión de dos computadoras.

Un socket es también un archivo, por lo que se identifica con un descriptor de archivo.

fd = socket(int dominio, int tipo, int protocolo);

Todo esta definido en /usr/include/sys/sockets.h

Y en Todo esta definido en /usr/include/sys/sockets.h

- RFC 791
- tcpdump -X -l
- Man in the middle, sniffers. ☺

Practica 17. CHAT. (OPCIONAL)

Puedes crear un cliente que se comunique con el "servidor" creado en clase?

Good luck.

CONTENEDORES!

Les han hablado mucho de virtualización, cuéntenme un poco.

Chroot() <- Jaulas, ¿Es virtualización?

*Contenedores :D