

# SO sesión 7: gestión de entrada/salida (open, read, write...)

domingo, 17 de enero de 2021

19:29

Para leer en el man	Descripción básica	Opc
<b>mknod</b>	Comando que crea un fichero especial	
<b>insmod</b>	Comando que inserta un módulo en el kernel	
<b>rmmod</b>	Comando que descarga un módulo del kernel	
<b>lsmod</b>	Comando que muestra el estado de los módulos cargados en el kernel	
<b>sudo</b>	Comando que permite ejecutar un comando como root	
<b>open</b>	Abre un fichero o dispositivo	
<b>write</b>	Llamada a sistema para escribir en un dispositivo virtual	
<b>read</b>	Llamada a sistema para leer de un dispositivo virtual	
<b>grep</b>	Comando que busca patrones en un fichero o en su entrada estándar si no se le pasa fichero como parámetro	
<b>ps</b>	Comando que muestra información sobre los procesos en ejecución	
<b>strace</b>	Lista las llamadas a sistema ejecutadas por un proceso	

Programa < entrada.txt (redireccionar la entrada)

---

iones a consultar

c,p

---

-c

-e, -o

-e, -c

---

Programa > salida.txt (redireccionar la salida)

Ls -l | grep es ( | es una pipe sin nombre, redirecciona la salida  
la entrada de grep es)

```
char buffer[64];
int num,i=0;
// Cuando el usuario apriete CtrlD el read devolverá 0
// Como no conocemos cuantas cifras tiene el número, hay que leerlo con un bucle
while (read(0,&buffer[i],1)>0) i++;
buffer[i]='\0';
num=atoi(buffer);
```

Lee un número de la consola (en ASCII) y luego lo convierte en entero

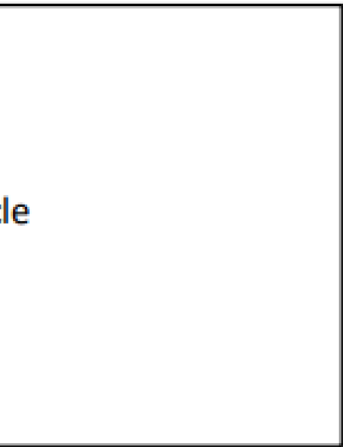
```
char buff[64];
int num=10562;
sprintf(buff,"%d",num);
write(1,buff,strlen(buff));
```

Para escribir un numero por la consola, debemos transformarlo a una cadena de caracteres primero, y luego hacer un write

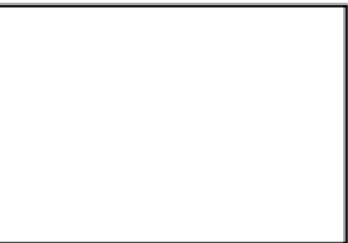
## Open: Creació

- Els fitxers especials han d'existir abans de poder accedir
- S'han d'especificar els **permission\_flags**

a de ls -l a



n un



lo a una



- És una OR (|) de: S\_IRWXU, S\_IRUSR, S\_IWUSR, etc
- Els fitxers de dades es poden crear a la vegada que fem l'open
  - En aquest cas hem d'afegir el flag O\_CREAT (amb una `access_mode`)
- No hi ha una crida a sistema per eliminar dades d'un fitxer, només les podem esborrar totes. Si volem truncar en continuar afegirem el flag O\_TRUNC a l'`access_mode`
  - Ex1: `open("X",O_RDWR|O_CREAT, S_IRUSR|S_IWUSR)`, el fitxer no existia el crea, si existia no té efecte
  - Ex2: `open("X",O_RDWR|O_CREAT|O_TRUNC, S_IRUSR|S_IWUSR)`, el fitxer no existia el crea, si existia s'alliberen les dades i es posa el tamany a 0 bytes.

1.40

## Open

- Per tant, com s'associa un nom a un dispositiu virtual?
- `fd = open(nom, access_mode[,permission_flags])`
  - Vincula nom de fitxer a dispositiu virtual (descriptor de fitxer)
    - ▶ A més, permet fer un sol cop les comprovacions de permisos, ja es pot executar *read/write* múltiples cops sense tornar a comprovar
  - Se li passa el **nom** del dispositiu i retorna el dispositiu virtual (un *descriptor* o canal) a través del qual es podran realitzar operacions de lectura/escriptura, etc
  - El **access\_mode** indica el tipus d'accés. Sempre ha d'indicar un o més d'aquests tres com a mínim :

open:  
(a OR de bits) a  
parcialment,  
ngut d'un fitxer  
S\_IWUSR) → Si  
S\_IRWXU) → Si  
s seves dades y

---

) ;  
fitxer o canal)  
proteccions. Un  
cops però ja no  
virtual (*fd: file*  
les operacions  
anar un

- ▶ O\_RDONLY (lectura)
- ▶ O\_WRONLY (escriptura)
- ▶ O\_RDWR (lectura i escriptura)

1.39

El major identifica el tipo de driver y el minor identifica la instancia concreta.

Para redireccionar terminales: `./es1 > /dev/pts/1`

While => espera activa, consume CPU

En un read, si no escribes nada, hace una espera activa, no consume CPU

Errno indica error => habitual (`errno == EINTR`), indica interrupció per signal

## E/S i execució concurrent (3)

- Si un procés està bloquejat en una operació d'E/S i l'operació és interrompuda per un signal podem tenir dos comportaments:
  - Després de tractar el signal, l'operació interrompuda es reinicia i el procés continuarà bloquejat a l'operació
  - Després de tractar el signal, l'operació retorna error i la variable `errno` contindrà el valor `EINTR`
- El comportament depèn de:
  - La gestió associada al signal:
    - ▶ si flag `SA_RESTART` a la sigaction → es reinicia l'operació
    - ▶ en cas contrari → l'operació retorna error

ancia

nsume

mpido por

3)



ció és  
ts diferents:  
s reinicia (el

a `errno` pren per

operació



▶ en cas contrari → l'operació retorna error

- L'operació que s'està fent (per exemple, mai es reinicia sobre sockets, operacions per a esperar signals, etc.)

- Exemple de com protegir la crida al sistema segons el mode de comportament:

```
while ( (n = read(...)) == -1 && errno ==
```

1.52

Cat /proc/devices => dispositivos instalados en la maquina

Ls -l /dev => carpeta con los devices

Lsmod => modulos en la maquina

en operacions

del de

EINTR );