

LPI 103.4. Utilitzar fluxos, canonades i redireccionaments

Wiki:

http://acacha.org/mediawiki/index.php/LPI_103.4

Objectius

103.4. Utilitzar fluxos, conductes i redireccions



- **Objectiu:** Els candidats han de ser capaços de redirigir fluxos i connectar-los per tal de processar dades de text de forma eficient. Les tasques inclouen redirigir l'entrada, la sortida i l'error estàndard i l'ús de canonades (pipes) per tal d'enviar la sortida d'una ordre a l'entrada d'un altra ordre, utilitzant la sortida d'una ordre com a argument d'un altra ordre i enviar la sortida d'una ordre tant a un fitxer com a a la sortida estàndard.
- **Pes:** 4



Àrees Clau de Coneixement:

- Redirigir l'entrada estàndard, la sortida estàndard i l'error estàndard.
- Utilitzar canonades per enllaçar la sortida d'una instrucció amb l'entrada d'una altra ordre.
- Utilitzar la sortida d'una ordre com a argument per a una altra ordre.
- Enviar la sortida tant cap a la sortida estàndard (stdout) com cap a un fitxer.



La següent és una llista parcial de fitxers, termes i utilitats utilitzades:

- [tee](#)
- [xargs](#)



Apunts: [LPI 103.4](#). Utilitzar fluxos, conductes i redireccions

♦ Flux de dades (stream)

- ♦ És un conjunt d'informació que es desplaça d'un origen a una destinació
- ♦ Els flux de dades són bàsics per tal de realitzar tasques complexes combinant programes senzills.

♦ 3 fluxes de dades estàndard a Linux

- ♦ **Entrada estàndard (stdin):** És el flux d'entrada de les ordres i aplicacions. Habitualment és el teclat.
- ♦ **Sortida estàndard (stdout):** És el flux de sortida de les ordres i aplicacions. Normalment és la terminal.
- ♦ **Error Estàndard (stderr):** Linux proporciona un segon tipus de flux de dades de sortida. La idea és tenir una sortida per aquelles dades amb alta prioritat, com per exemple els missatges d'error.

♦ Internament, el nucli els tracta com fitxers:

```
$ ls -la /dev/std*  
lrwxrwxrwx 1 root root 15 2009-09-27 14:54 /dev/stderr -> /proc/self/fd/2  
lrwxrwxrwx 1 root root 15 2009-09-27 14:54 /dev/stdin -> /proc/self/fd/0  
lrwxrwxrwx 1 root root 15 2009-09-27 14:54 /dev/stdout -> /proc/self/fd/1
```

- ♦ L'entrada i sortida d'un programa o ordre són tractats a Linux com flux de dades.
- ♦ L'entrada per defecte és el teclat i la sortida per defecte és la terminal (o millor dit, l'emulador de terminal)
- ♦ Aquest flux de dades es poden manipular per tal d'adaptar-los a les nostres necessitats.
- ♦ La potència del sistema prové de la possibilitat de poder modificar els fluxs de dades estàndard i per exemple poder utilitzar com a entrada estàndard un fitxer, o un dispositiu de maquinari, el resultat d'un altre comanda, etc.

♦ Resumida en una frase

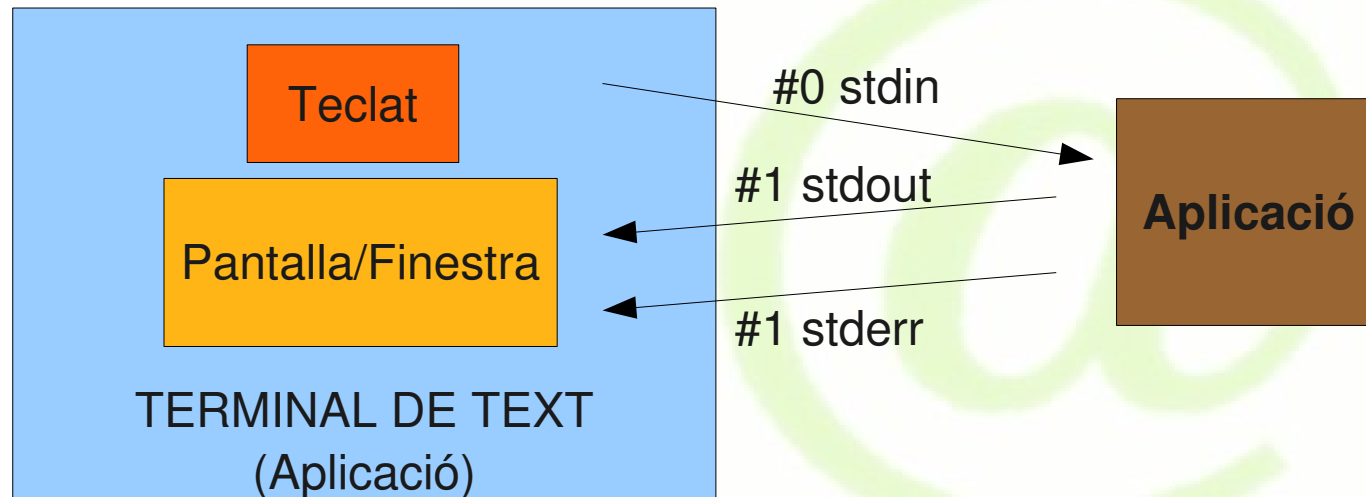
Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

Doug McIlroy, creador del conductes Unix

- ♦ Programes petits amb una sola utilitat
- ♦ Eficiència des programes
- ♦ Les operacions complexes es realitzen combinant programes
- ♦ Es treballa amb fluxes de dades de text
- ♦ Filosofia Linux a la wikipedia anglesa

Descriptors de fitxers

♦ Descriptors de fitxers



```
$ lsof | grep pts
bash      20198  sergi    0u      CHR      136,0    2 /dev/pts/0
bash      20198  sergi    1u      CHR      136,0    2 /dev/pts/0
bash      20198  sergi    2u      CHR      136,0    2 /dev/pts/0
bash      20198  sergi   255u    CHR      136,0    2 /dev/pts/0
lsof      21074  sergi    0u      CHR      136,0    2 /dev/pts/0
lsof      21074  sergi    2u      CHR      136,0    2 /dev/pts/0
grep      21075  sergi    1u      CHR      136,0    2 /dev/pts/0
grep      21075  sergi    2u      CHR      136,0    2 /dev/pts/0
```

♦ Permeten modificar els flux de dades estàndard



- ♦ Podem modificar l'entrada o la sortida estàndard que utilitza una ordre o aplicació.
- ♦ Redirreccionar la sortida d'una ordre cap a un fitxer:

```
$ echo Hola! > hola.txt  
$ cat hola.txt  
Hola!
```

- ♦ 2 tipus principals
 - Redirecció de la sortida: >
 - Redirecció de l'entrada: <

Operadors de redirecció (bash)

◆ Redireccions amb bash

Operador	Descripció
>	Crea un nou fitxer contenint la sortida estàndard (no la d'error!) de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents)
>>	Afegeix (append) un fitxer la sortida estàndard (no la d'error!). Si el fitxer no existeix es crea. Les noves dades s'afegeixen al final del fitxer i no s'eliminen les dades existents.
2>	Crea un nou fitxer contenint la sortida d'error estàndard de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents)
2>>	Afegeix (append) a un fitxer la sortida d'error estàndard (no la d'error!). Si el fitxer no existeix es crea. Les noves dades s'afegeixen al final del fitxer i no s'eliminen les dades existents.
&>	Crea un nou fitxer contenint la sortida estàndard i la sortida d'error estàndard de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents)
<	Envia les dades del fitxer especificat a la dreta com a entrada estàndard de la comanda de l'esquerra.
<<	Accepta el text de les següents línies com a entrada estàndard. També conegut com here document ( heredoc) 
<>	El fitxer especificat a la dreta es tant la entrada estàndard com la sortida estàndard de la comanda de l'esquerra.

- ◆ \$ man bash
 - Secció REDIRECTION



Redireccions. Exemples

- ♦ Enviar tot a enlloc:

```
$ grep -n -r "linux" / Prova &> /dev/null
```

 - **/dev/null** és una espècie de forat negre.

- ♦ Redireccionar només els errors:

```
$ grep -n -r "linux" / 2>/home/sergi/grep-errors.txt
```

- ♦ Append:

```
$ ls >> file_list.txt
```

- ♦ Ordenar un fitxer:

```
$ sort < file_list.txt
```

- ♦ Ordenar un fitxer i guardar el resultat en un altre fitxer

```
$ sort < file_list.txt > sorted_file_list.txt
```

Heredoc (here document)

- ◆ **Permet treballar amb múltiples línies (documents)**

- ◆ <<INDICADOR_FINAL DE FITXER

```
$ cat <<PROVA
> dsadsasadsad
> sadddsadsa
> dsadsadsa
> sdadsadsa
> PROVA
dsadsasadsad
sadddsadsa
dsadsadsa
sdadsadsa
```

```
$ cat <<EOF
> Això és un document
> de múltiples línies
> un altre línia
> EOF
Això és un document
de múltiples línies
un altre línia
```

- ◆ S'utilitza en múltiples llenguatges de programació (PHP, C, Java...)

♦ Conductes (pipes)

- ♦ Els conductes són un mecanisme que permet enllaçar la sortida estàndard d'una ordre amb l'entrada estàndard d'una altra ordre.
- ♦ Permeten concatenar l'execució de diverses comandes:

```
$ comanda1 | comanda2 | comanda3 | comanda4 | ...
```

- Aquest tipus de conductes s'anomenen conductes de dades.

♦ Exemple

- ♦ Comprovar si està instal·lat un paquet Debian

```
$ dpkg -l | grep manpages
ii  manpages                                3.15-1
    Manual pages about using a GNU/Linux system
```

- ♦ Fixeu-vos que es pot implementar un conducte mitjançant un fitxer intermediari:

```
$ dpkg -l > intermediari.txt ; grep manpages < intermediari.txt;
$ rm intermediari.txt
```

- ♦ Visualitzar fitxers

```
$ cat fitxer | more
$ cat fitxer | less
```

Conductes amb nom

- ♦ **Conducte amb nom (aka FIFO - First In First Out)**
 - ♦ És una extensió del concepte tradicional de conducte (pipe) i s'utilitza en sistemes Unix-like com un mecanisme més de comunicació entre processos (IPC Inter Proces Communication).
 - ♦ Un conducte tradicional és "sense nom" per que existeix anònimament i nomé existeix mentrestant els processos existeixen. Un conducte amb nom, es creat pel sistema operatiu i persisteix més enllà del temps de vida dels processos que l'utilitzen.



Conductes amb nom

- ♦ Els conductes amb nom són un tipus de fitxer i es reconeixen per que la sortida de l'ordre **ls -l** els marca com a **fitxers de tipus p**:

- ♦ **Es poden crear amb mkfifo o mknod**

```
$ mkfifo prova  
$ ls -l prova  
prw-r--r-- 1 sergi sergi 0 2009-10-11 17:29 prova
```

- ♦ **Exemple. Conducte compressor**

```
$ mkfifo conducte_compresor  
$ gzip -9 -c < conducte_compresor > sortida.gz  
    En una altra terminal podem executar:  
$ cat fitxer > conducte_comprimidor
```

Ordre tee



Tee-shirt



Tee-pipe

♦ Dividir la sortida estàndard

- ♦ Per tal que sigui mostrada per pantalla i guardada en tants fitxers com s'indiquin.
- ♦ Normalment s'utilitza conjuntament amb conductes (pipes) de la següent forma:

```
$ ls -la | tee ls.txt
```

- Us mostrarà el resultat de la comanda `ls -la` i guardarà aquest resultat al fitxer `ls.txt`.
- Per defecte `tee` sobreescriu els fitxers que indiquem (elimini els continguts previs si el fitxer ja existia). Per evitar aquest comportament podeu utilitzar el paràmetre `-a`.



Ordre xargs

♦ Xargs

- ♦ Permet construir arguments complexes de comandes.
- ♦ La millor forma d'entendre com funciona xargs és veure un exemple:

```
$ find . -type d -name ".svn" -print | xargs rm -rf
```

- Esborra de forma massiva fitxers que compleixen amb un patró (fitxer .svn a l'exemple)
- Té moltes opcions. Convertir llista en comes en una llista

```
$ echo "foo,bar,baz" | xargs -d, -L 1 echo
foo
bar
baz
```



Reconeixement 3.0 Unported

Sou lliure de:



copiar, distribuir i comunicar públicament l'obra



fer-ne obres derivades

Amb les condicions següents:



Reconeixement. Heu de reconèixer els crèdits de l'obra de la manera especificada per l'autor o el llicenciador (però no d'una manera que suggereixi que us donen suport o rebeu suport per l'ús que feu l'obra).

- Quan reutilitzeu o distribuïu l'obra, heu de deixar ben clar els termes de la llicència de l'obra.
- Alguna d'aquestes condicions pot no aplicar-se si obteniu el permís del titular dels drets d'autor.
- No hi ha res en aquesta llicència que menyscabi o restringeixi els drets morals de l'autor.

Advertiment

Els drets derivats d'usos legítims o altres limitacions reconegudes per llei no queden afectats per l'anterior
Això és un resum fàcilment llegible del **text legal** (la llicència completa).

<http://creativecommons.org/licenses/by/3.0/deed.ca>