Sistemi Operativi
scripting, command line tools, streams

Sistemi Operativi 2018-2019

Marco Tessarotto

Unix Design Philosophy

- *This is the Unix philosophy: 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*)*
- **programmi che fanno una cosa sola e la fanno bene**; esistono strumenti separate per ciascuna attività (cat ls man …)
- **Gli strumenti per command line di Unix interoperano** bene perchè si completano a vicenda

Unix Design Philosophy

- la capacità di **gestire flussi di testo** è alla base della pipeline e ciò che consente di "accoppiare" piccoli e semplici comandi Unix per formare operazioni Unix più complesse e interessanti.
- Questa filosofia porta allo sviluppo di strumenti da riga di comando Unix ben formati che hanno le tre proprietà:
- Possono **ricevere l'input** dal terminale o attraverso una pipeline o leggendo un file di input fornito come argomento
- Possono **scrivere il loro output** sul terminale in modo che possa essere letto dall'utente o letto come input da un altro comando.
- Non scrivono le **informazioni di errore** sullo standard output per non creare confusione
- Questo processo di ricevere l'input dal terminale e scrivere output sul terminale è alla base della gestione di flussi di testo attraverso la pipeline.

Standard Input, Standard Output, and Standard Error (lec 03)

- Ogni programma Unix viene fornito con tre «file streams» standard o descrittori di file standard da cui leggere e scrivere. (stream: flusso)
- Standard Input (**stdin** file stream, file descriptor **0**): Il flusso di input (input stream) principale per leggere le informazioni provenienti da terminale (stdin: «standard input»)
- Standard Output (**stdout** file stream, file descriptor **1**): Il flusso (stream) di output principale per la stampa a terminale dell'output del programma (stdout: «standard output»)
- Standard Error (**stderr** file stream, file descriptor **2**): Il flusso (stream) di errore principale per la stampa a terminale di informazioni relative ad errori nell'elaborazione (stderr: «standard error»)

Esempio di script

- Scripting è sinonimo di programmazione con la differenza che è generalmente considerato più “leggero”

#!/bin/bash

echo "Hello World" # This is a comment

chmod +x helloworld.sh

- Esecuzione:

#> ./helloworld.sh

Hello World

(eserc.)

pipe

- La pipe (|) è un costrutto semantico della shell per collegare lo standard output di un programma allo standard input di un altro programma, quindi convogliando l'output all'input.
- *cat /proc/cpuinfo | more*
- *"cat" scrive al suo stdout → PIPE → "more" legge dal suo stdin*
- ***lo stdout di cat diventa lo stdin di more***
- Il fatto che l'input sia collegato all'output in una pipeline richiede effettivamente lo stderr perché se si dovesse verificare un errore lungo la pipeline, non si vuole che quell'errore si propagasse come input al prossimo programma nella pipeline. È necessario un meccanismo per segnalare l'errore al terminale all'esterno della pipeline e tale meccanismo è lo standard error.
- Pipe: la shell esegue tutti i programmi simultaneamente, collegando tra di loro i "tubi"(unidirezionali) stdout -> stdin

Redirecting stdin, stdout, and stderr

- Oltre a convogliare i flussi di file standard (stdin…) tramite pipe, si possono reindirizzarli a un file sul filesystem.
- I simboli di reindirizzamento sono > e <.
- cmd < input_file > output_file 2> error_file
- Ciò significherebbe che cmd leggerà l'input dal file input_file, tutto l'output che normalmente andrebbe a stdout verrà scritto in output_file e qualsiasi eventuale messaggio di errore verrà scritto in error_file. Notare che 2> indicano il reindirizzamento del descrittore di file 2 (stderr) verso un file.
- i reindirizzamenti > e < hanno sempre la precedenza su una pipe, l'ultimo reindirizzamento in una sequenza di > o < ha la precedenza.
- I reindirizzamenti di output troncheranno automaticamente il file verso cui viene reindirizzato. Cioè il file (se esistente) viene cancellato e ne creerà uno nuovo. Esistono situazioni in cui, invece, si desidera accodare alla fine del file, ad esempio log files. L'accodamento di output si specifica il simbolo >>
- cat input_file > out
- cat input_file >> out

Scrivere su stderr

- Uno script di bash è come gli altri strumenti a riga di comando, per quanto riguarda output, input e stderr. Vengono forniti automaticamente gli stessi nomi di file standard e i descrittori di file, oltre alla possibilità di reindirizzarli.
- Se si desidera scrivere direttamente l'errore standard nello script, si reindirizza l'output del comando echo.
- *echo "ERRORE: descrizione del problema" 1>&2 #<-- redirect file descriptor 1 to 2*
- i file standard hanno descrittori di file numerici, 0 per stdin, 1 per stdout e 2 per stderr. Quando si esegue il reindirizzamento, si sceglie lo stream in base al numero del descrittore

esercitazione

- Il file /etc/passwd contiene le informazioni di login (non le password) degli utenti sul sistema. Ogni riga è del tipo:

username groupid home directory

↓ ↓ ↓ ↓

marco:x:1003:1003:Marco Tessarotto,,,:/home/u140536:/bin/bash

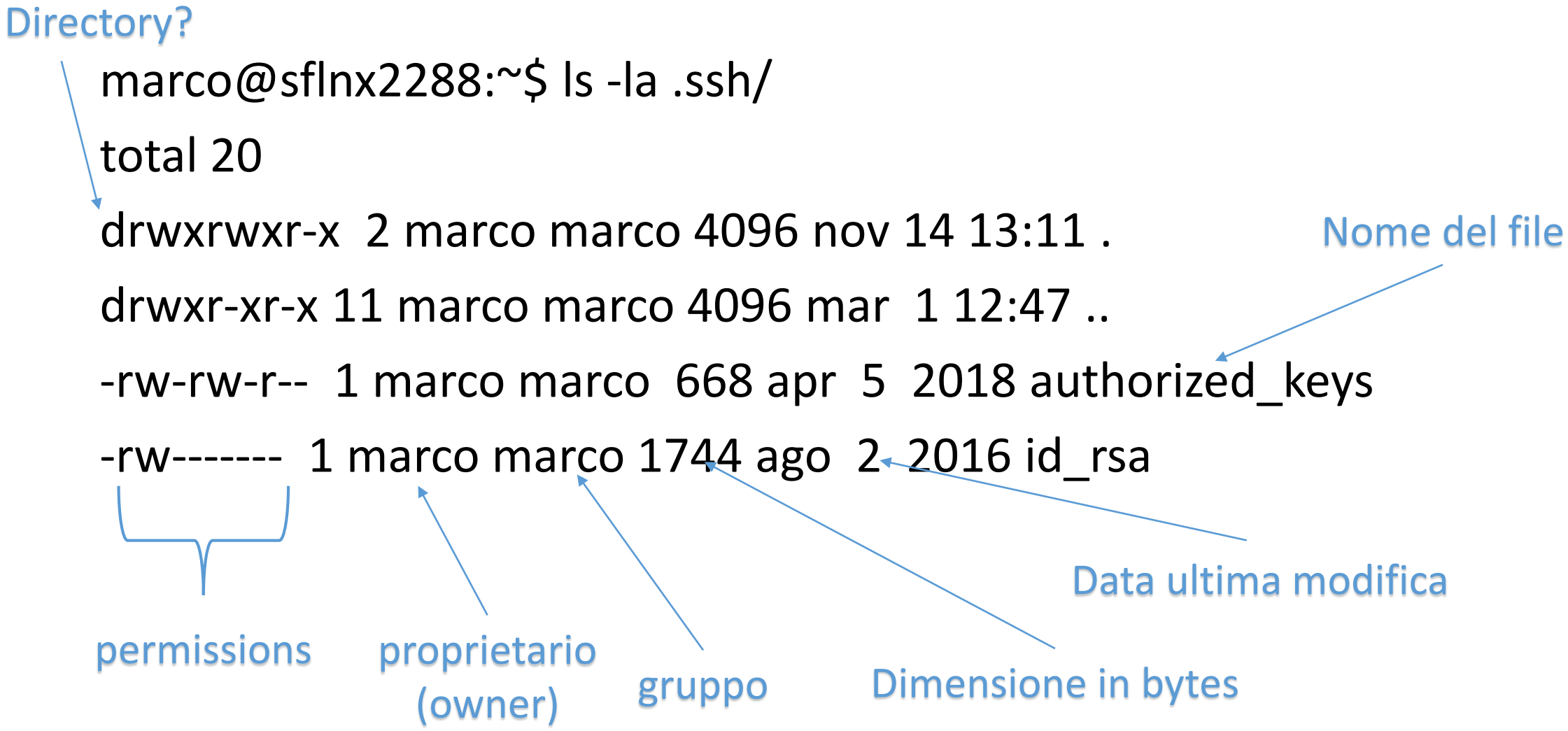
↑ ↑ ↑

uid name default shell

Alcune variabili di environment

- USER : the current user
- HOME : the home directory
- PWD : the current working directory
- SHELL : Il nome del programma di shell attualmente in esecuzione
- PATH è una variabile di environment in Linux e in altri sistemi operativi simili a Unix che istruisce la shell in quali directory cercare i file eseguibili (cioè i programmi pronti per l'esecuzione)

File Permissions and Ownership



ownership

- the owner/group and the permissions. The owner and the permissions are directly related to each other.
- Il proprietario (owner) di un file è l'utente che è direttamente responsabile del file ed ha uno status speciale rispetto ai permessi del file stesso
- Gli utenti possono essere raggruppati in gruppo, una collezione di utenti che condividono gli stessi permessi.
- marco@sflnx2288:~$ who am i
- marco pts/0   2019-02-01 11:19 (140.xx.xx.xxx)

Password e group files

- I gruppi sono definiti in due posizioni. Il primo è il file chiamato /etc/passwd che gestisce tutti gli utenti del sistema. Esempio di entry:
- marco@sflnx2288:~$ grep marco /etc/passwd
- marco:x:1003:1003:Marco Tessarotto,,,:/home/marco:/bin/bash
- Le prime due parti del file descrivono lo userid ed il groupid, che sono 1003 e 1003, rispettivamente.
- Questi numeri sono il vero nome del gruppo e dello username, ma Linux, per nostra comodità, li converte in nomi.
- L'informazione per tradurre userid in username è contenuta nel file /etc/passwd.

Password e file di gruppo

- L’informazione per tradurre il groupid numeric in group name si trova nel file /etc/group. Esempio di entry nel group file:
- marco@sflnx2288:~$ grep marco /etc/group
- sudo:x:27:ubuntu,pincop,marco
- www-data:x:33:pincop,marco,ftpuser
- lxd:x:110:ubuntu,pincop,marco,megaazienda
- marco:x:1003:
- admin:x:1004:pincop,marco,megaazienda

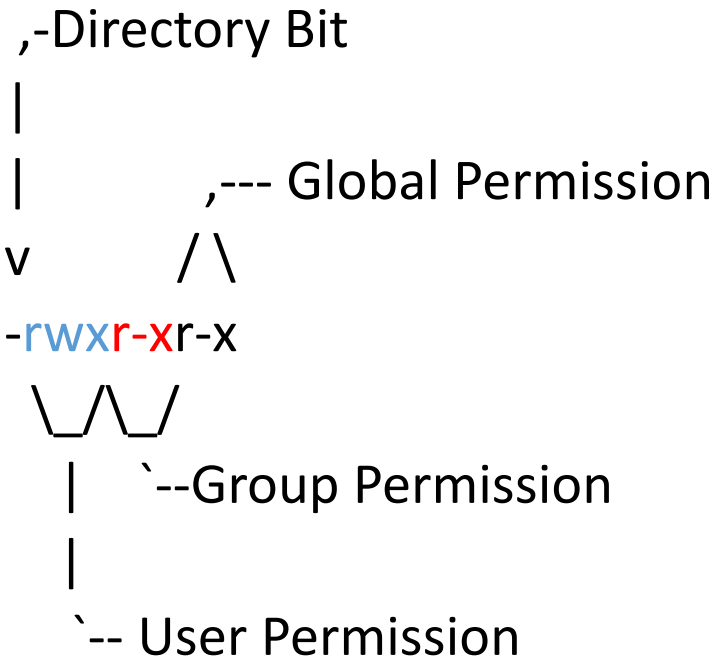
- Ad esempio, si può vedere che gli utenti picop, marco e ftpuser appartengono al gruppo www-data

File permissions

- Ora possiamo rivolgere la nostra attenzione alla stringa di autorizzazione. Un permesso è semplicemente una sequenza di 9 bit suddivisa in 3 ottetti di 3 bit ciascuno. Un ottetto è un numero di base 8 che va da 0 a 7 e 3 bit definiscono un ottetto in modo univoco poiché tutti i numeri compresi tra 0 e 7 possono essere rappresentati in 3 bit.
- All'interno di un ottetto, ci sono tre flag di autorizzazione, **lettura, scrittura ed esecuzione**. Questi sono spesso indicati con l'abbreviazione: r, w e x. L'impostazione di un'autorizzazione su «on» significa che il bit è 1. Quindi per un insieme di possibili stati di autorizzazione, possiamo definirlo univocamente con un numero ottale
- rwx -> 1 1 1 -> 7
- r-x -> 1 0 1 -> 5
- --x -> 0 0 1 -> 1
- rw- -> 1 1 0 -> 6

File permissions

- Un'autorizzazione file completa è costituita da una tupla di ottetti, nell’ordine permessi utente, gruppo e globale.



Esempio permessi

- Creare uno script helloworld.sh
- Provare ad eseguire:
- chmod 700 helloworld.sh
- chmod +x helloworld.sh

Cambiare «ownership»

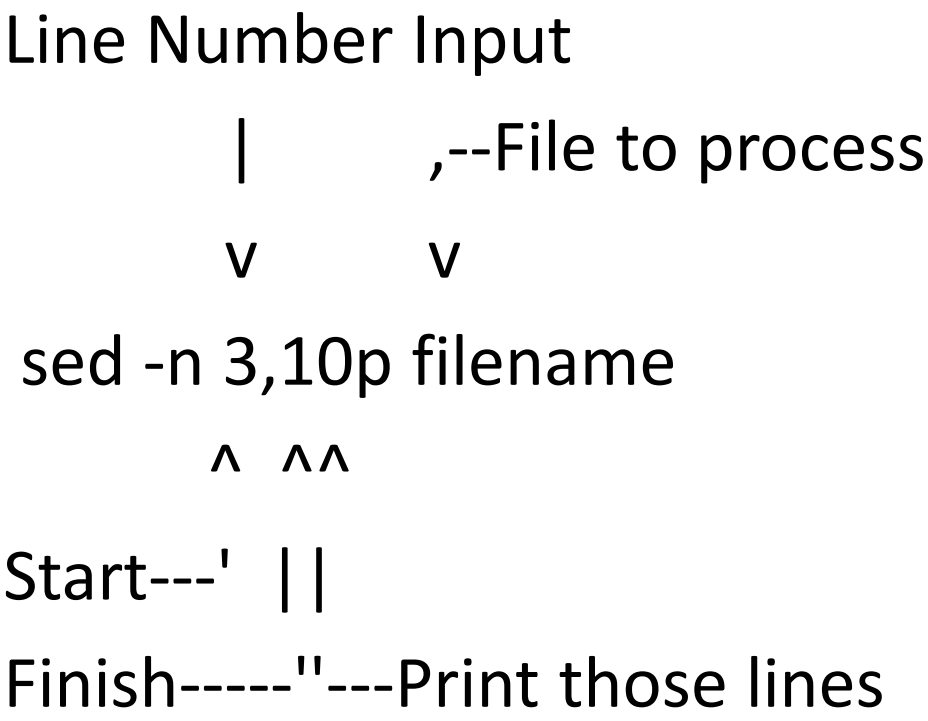
- come possiamo cambiare il proprietario ed il gruppo di un file?

- **chown** user file/directory : change owner of the file/directory to the user
- **chgrp** group file/directory : change group of the file to the group
- L'autorizzazione a cambiare il proprietario di un file è riservata solo al superuser per motivi di sicurezza. Tuttavia, la modifica del gruppo del file è riservata solo al proprietario.

Alcuni strumenti per riga di comando

- cat
- less
- more
- La differenza fondamentale tra i due visualizzatori di file è che less ti permette di andare avanti e indietro in un file mentre more ti permette solo di andare avanti nel file, uscendo alla fine.
- head
- tail

sed


Line Number Input
|
,--File to process
v
v
sed -n 3,10p filename
^ ^^
Start---' ||
Finish-----''---Print those lines

Grep

- Il comando grep elabora solo le linee che corrispondono a una condizione.
- grep utente /etc/passwd
- …

Pipelines con cut, wc…

- Il pezzo finale del puzzle per l'elaborazione dei file consiste nel prendere l'output dell'elaborazione di un file e impostarlo come input per un altro processo. Queste parti del processo possono essere incatenate insieme in una pipeline. Considera questa semplice pipeline di seguito:

- #> cat /proc/cpuinfo | head -10
- La pipe prende l'output di un comando e lo imposta come input di un altro. Nell'esempio sopra, l'output del comando cat è di stampare l'intero contenuto del file sull'input di head, che quindi stampa solo le prime 10 righe del suo input.

Word count (wc)

- grep utente /etc/passwd | wc -l
- La prima parte del comando stamperà solo le righe che contengono il pattern «utente», questo output viene quindi impostato come input per il comando wc, che è uno strumento da riga di comando per contare parole, linee e byte. L'opzione -l dice di stampare solo il conteggio delle righe, e quindi il comando sopra mostra il numero di linee.

cut

- Uno strumento molto utile per le pipe è “cut” che è usato per estrarre campi da un file formattato. esempio:

```
              ,--- Deliminatore
            __|_          ,-----Input File, or leave off to read from stdin
             /   \          v
#> cut -d "," -f 1 my_file | head -5
            \__/
             \__.– Field
```

- il deliminatore determina come il file va tagliato.
- (provare a creare un file csv con excel e vedere come è fatto)

esercizio

- Esercizio: scrivere uno script, allusers.sh, che legga il file /etc/passwd e mostri a video il campo “home directory” di tutti gli utenti
- [hint: usare cut]

/dev

- I file che troviamo in /dev non sono realmente file, ma «dispositivi» (devices) forniti dal sistema operativo. Un dispositivo generalmente si collega a un componente di input o output del sistema operativo.
- I tre dispositivi /dev/null, /dev/zero e /dev/urandom sono funzioni speciali del sistema operativo per fornire all'utente: uno «spazio nullo», una sorgente di «zero» (zero) e una sorgente di entropia (urandom).
- I dispositivi sono un modo per connettere lo spazio utente con lo spazio del kernel attraverso il file system.
- Ognuno dei file è il collegamento con alcuni processi del sistema operativo. Ad esempio, /dev/tty rappresenta un terminale aperto sul computer. /dev/ram si riferisce alla memoria del computer.
- /dev/dvd e /dev/cdrom permettono di accedere alle rispettive unità fisiche.

/dev

- /dev : percorso che significa che non si tratta di un file «standard», ma piuttosto di un dispositivo o un servizio fornito dal sistema operativo Linux
- /dev/null: se proviamo a leggere, non otteniamo nulla; se scriviamo dati, questi vengono fatti “scomparire”

Esempio: head -3 BAD_FILENAME 2> /dev/null

- /dev/zero: fornisce in lettura soltanto bytes di valore zero. Per esempio, per creare un file di lunghezza 20 bytes di valore zero:

head -c 20 /dev/zero > zero-20-byte-file.dat

- /dev/urandom: fornisce in lettura bytes random. Per esempio, per creare un file di lunghezza 20 bytes di valori casuali:

head -c 20 /dev/urandom > random-20-byte-file.dat