Command line kung fu

Bash, filtri & co.

Riccardo Binetti
punkerbino@gmail.com

Corsi GNU/Linux Avanzati 2014

“Il terminale? Nel 2014?”

Sì, il terminale viene usato anche nell’era delle GUI stilose

- È agnostico rispetto a distribuzione, ambiente grafico e OS[1]
- Si può usare da remoto e/o su una macchina senza schermo
- Automatizzazione molto facilitata
- Se nulla altro sembra andare, un terminale non si nega a nessuno

[1] Finché rimaniamo nei sistemi unix-like

“Il terminale? Nel 2014?”

Sì, il terminale viene usato anche nell’era delle GUI stilose

- È agnostico rispetto a distribuzione, ambiente grafico e OS[1]
- Si può usare da remoto e/o su una macchina senza schermo
- Automatizzazione molto facilitata
- Se nulla altro sembra andare, un terminale non si nega a nessuno

[1] Finché rimaniamo nei sistemi unix-like

“Il terminale? Nel 2014?”

Sì, il terminale viene usato anche nell’era delle GUI stilose

- È agnostico rispetto a distribuzione, ambiente grafico e OS[1]
- Si può usare da remoto e/o su una macchina senza schermo
- Automatizzazione molto facilitata
- Se nulla altro sembra andare, un terminale non si nega a nessuno

[1] Finché rimaniamo nei sistemi unix-like

“Il terminale? Nel 2014?”

Sì, il terminale viene usato anche nell’era delle GUI stilose

- È agnostico rispetto a distribuzione, ambiente grafico e OS[1]
- Si può usare da remoto e/o su una macchina senza schermo
- Automatizzazione molto facilitata
- Se nulla altro sembra andare, un terminale non si nega a nessuno

[1] Finché rimaniamo nei sistemi unix-like

“Il terminale? Nel 2014?”

Sì, il terminale viene usato anche nell’era delle GUI stilose

- È agnostico rispetto a distribuzione, ambiente grafico e OS[1]
- Si può usare da remoto e/o su una macchina senza schermo
- Automatizzazione molto facilitata
- Se nulla altro sembra andare, un terminale non si nega a nessuno

[1] Finché rimaniamo nei sistemi unix-like

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo n si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo `q` si esce dalla manpage
- `man man`

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo `n` si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo `q` si esce dalla manpage
- `man man`

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo n si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo `q` si esce dalla manpage
- `man man`

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo n si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo q si esce dalla manpage
- `man man`

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo n si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo `q` si esce dalla manpage
- `man man`

Un comando per domarli tutti

`man`

- Se siete in dubbio su cosa faccia un comando, digitate `man comando`. Vi risolverà buona parte dei dubbi e vi risparmierà un po' di "RTFM"
- La sintassi del comando sarà del tipo `comando [opzioni] [argomenti] argomento`. Tutto ciò che è tra parentesi quadre è opzionale, le opzioni si passano con uno o due trattini (- o --)
- Per cercare all'interno della man page basta scrivere `/terminecercato` e premere `Enter`. Premendo n si passa al prossimo match.
  - La ricerca non fa il wrap-around, arrivati alla fine della manpage premete `p` per tornare in cima
- Premendo `q` si esce dalla manpage
- `man man`

Muovere i primi passi

- `ls` – elenca i file nella cartella corrente
  - `ls nomecartella` -- elenca i file della cartella con quel nome
  - `-lah` – mostra i file incolonnati con maggiori informazioni (-l) includendo anche i file nascosti (-a) con le dimensioni in formato human-readable (-h)
- `cd nomecartella` – ci sposta nella directory
  - `.` è la directory corrente, `..` è la directory un livello più in su
- `pwd` – stampa la directory in cui ci si trova

Muovere i primi passi

- `ls` – elenca i file nella cartella corrente
  - `ls nomecartella` -- elenca i file della cartella con quel nome
  - `-lah` – mostra i file incolonnati con maggiori informazioni (-l) includendo anche i file nascosti (-a) con le dimensioni in formato human-readable (-h)
- `cd nomecartella` – ci sposta nella directory
  - . è la directory corrente, .. è la directory un livello più in su
- `pwd` – stampa la directory in cui ci si trova

Muovere i primi passi

- `ls` – elenca i file nella cartella corrente
  - `ls nomecartella` -- elenca i file della cartella con quel nome
  - `-lah` – mostra i file incolonnati con maggiori informazioni (-l) includendo anche i file nascosti (-a) con le dimensioni in formato human-readable (-h)
- `cd nomecartella` – ci sposta nella directory
  - . è la directory corrente, .. è la directory un livello più in su
- `pwd` – stampa la directory in cui ci si trova

Creiamo qualcosa

- `touch nomefile` – crea un file vuoto se non esiste
- `mkdir nomecartella` – crea una directory vuota
  - -p – crea tutte le directory necessarie (e.g. `mkdir a/b/c/` crea anche a e b se non esistono)

Nota importante

Il filesystem Linux, a differenza di quello Windows, è case sensitive.
Se create (o modificate) i file ricordate che `foo` $\neq$ `Foo` $\neq$ `FOO`

Creiamo qualcosa

- `touch nomefile` – crea un file vuoto se non esiste
- `mkdir nomecartella` – crea una directory vuota
  - -p – crea tutte le directory necessarie (e.g. `mkdir a/b/c/` crea anche a e b se non esistono)

Nota importante

Il filesystem Linux, a differenza di quello Windows, è case sensitive.
Se create (o modificate) i file ricordate che `foo` $\neq$ `Foo` $\neq$ `FOO`

Creiamo qualcosa

- `touch nomefile` – crea un file vuoto se non esiste
- `mkdir nomecartella` – crea una directory vuota
  - -p – crea tutte le directory necessarie (e.g. `mkdir a/b/c/` crea anche a e b se non esistono)

Nota importante

Il filesystem Linux, a differenza di quello Windows, è case sensitive.
Se create (o modificate) i file ricordate che `foo` $\neq$ `Foo` $\neq$ `FOO`

Io ti ho creato, io ti distruggo

- `rm nomefile` – cancella un file
  - `-r` – se il file è una cartella, cancella ricorsivamente il suo contenuto
- `rmdir nomecartella` – cancella una cartella solo se vuota
- Se siete paranoici: `shred nomefile` – sovrascrive un file con bit random 3 volte
  - `-u` – dopo averlo sovrascritto, lo cancella

Attenzione

Nel terminale, non esiste un "cestino". Una volta che avete rimosso un file, è perso per sempre.[2]

[2] A meno di strani magheggi con software forense

Io ti ho creato, io ti distruggo

- `rm nomefile` – cancella un file
  - `-r` – se il file è una cartella, cancella ricorsivamente il suo contenuto
- `rmdir nomecartella` – cancella una cartella solo se vuota
- Se siete paranoici: `shred nomefile` – sovrascrive un file con bit random 3 volte
  - `-u` – dopo averlo sovrascritto, lo cancella

Attenzione

Nel terminale, non esiste un "cestino". Una volta che avete rimosso un file, è perso per sempre.[2]

[2] A meno di strani magheggi con software forense

Io ti ho creato, io ti distruggo

- `rm nomefile` – cancella un file
  - `-r` – se il file è una cartella, cancella ricorsivamente il suo contenuto
- `rmdir nomecartella` – cancella una cartella solo se vuota
- Se siete paranoici: `shred nomefile` – sovrascrive un file con bit random 3 volte
  - `-u` – dopo averlo sovrascritto, lo cancella

Attenzione

Nel terminale, non esiste un "cestino". Una volta che avete rimosso un file, è perso per sempre.[2]

[2] A meno di strani magheggi con software forense

Io ti ho creato, io ti distruggo

- `rm nomefile` – cancella un file
  - `-r` – se il file è una cartella, cancella ricorsivamente il suo contenuto
- `rmdir nomecartella` – cancella una cartella solo se vuota
- Se siete paranoici: `shred nomefile` – sovrascrive un file con bit random 3 volte
  - `-u` – dopo averlo sovrascritto, lo cancella

Attenzione

Nel terminale, non esiste un "cestino". Una volta che avete rimosso un file, è perso per sempre.[2]

[2] A meno di strani magheggi con software forense

Modifiche di base ai file

- `cp sorgente destinazione` – Copia il file `sorgente` nel file `destinazione`
  - `-r` – Da usare per copiare una directory
- `mv sorgente destinazione` – Sposta il file sorgente in destinazione
  - Si può usare per rinominare i file se `destinazione` è nella stessa cartella di `sorgente`
- `ln -s sorgente /path/al/collegamento` – Crea un collegamento simbolico al file `sorgente` in `/path/al/collegamento`

Modifiche di base ai file

- `cp sorgente destinazione` – Copia il file `sorgente` nel file `destinazione`
  - `-r` – Da usare per copiare una directory
- `mv sorgente destinazione` – Sposta il file sorgente in destinazione
  - Si può usare per rinominare i file se `destinazione` è nella stessa cartella di `sorgente`
- `ln -s sorgente /path/al/collegamento` – Crea un collegamento simbolico al file `sorgente` in `/path/al/collegamento`

Modifiche di base ai file

- `cp sorgente destinazione` – Copia il file `sorgente` nel file `destinazione`
  - `-r` – Da usare per copiare una directory
- `mv sorgente destinazione` – Sposta il file sorgente in destinazione
  - Si può usare per rinominare i file se `destinazione` è nella stessa cartella di `sorgente`
- `ln -s sorgente /path/al/collegamento` – Crea un collegamento simbolico al file `sorgente` in `/path/al/collegamento`

Historia magistra vitae

- `Ctrl+p` o ↑ per andare indietro (comandi più vecchi) nella history dei comandi
- `Ctrl+n` o ↓ per avanti (comandi più recenti) nella history dei comandi
- `Ctrl+r` per effetturare la ricerca di una stringa nei comandi della history
- La history è salvata nel file `.bash_history` nella home dell'utente

Historia magistra vitae

- `Ctrl+p` o ↑ per andare indietro (comandi più vecchi) nella history dei comandi
- `Ctrl+n` o ↓ per avanti (comandi più recenti) nella history dei comandi
- `Ctrl+r` per effetturare la ricerca di una stringa nei comandi della history
- La history è salvata nel file `.bash_history` nella home dell'utente

Historia magistra vitae

- `Ctrl+p` o ↑ per andare indietro (comandi più vecchi) nella history dei comandi
- `Ctrl+n` o ↓ per avanti (comandi più recenti) nella history dei comandi
- `Ctrl+r` per effetturare la ricerca di una stringa nei comandi della history
- La history è salvata nel file `.bash_history` nella home dell'utente

Historia magistra vitae

- `Ctrl+p` o ↑ per andare indietro (comandi più vecchi) nella history dei comandi
- `Ctrl+n` o ↓ per avanti (comandi più recenti) nella history dei comandi
- `Ctrl+r` per effetturare la ricerca di una stringa nei comandi della history
- La history è salvata nel file `.bash_history` nella home dell'utente

Domande?

Demo!

Domande?

Demo!

Scriviamo qualcosa

Dobbiamo poter modificare in qualche modo i nostri file.
Text editors to the rescue!
Ne vediamo 3 in particolare:

- `nano`
- `vim`
- `emacs`

nano

"nano è il miglior text editor del mondo" cit. Nessuno

- Presente di default in tutte le maggiori distribuzioni
- Semplice da usare
- Piuttosto basilare
- Avviatelo sempre con l'opzione `-w` (altrimenti spezza le righe troppo lunghe)
- I comandi sono segnati in basso
  - ^X va interpretato come Ctrl+X

nano

"nano è il miglior text editor del mondo" cit. Nessuno

- Presente di default in tutte le maggiori distribuzioni
- Semplice da usare
- Piuttosto basilare
- Avviatelo sempre con l'opzione `-w` (altrimenti spezza le righe troppo lunghe)
- I comandi sono segnati in basso
  - ^X va interpretato come Ctrl+X

vim

"I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it."

- Editor modale
- I modi di utilizzo base sono: Normal, Insert e Command mode
  - Nella modalità Normal (a cui si accede premendo `Esc`) si può navigare all'interno del file di testo
  - Nella modalità Insert (a cui si accede premendo `i` in modalità Normal) si può modificare il file
  - Nella modalità Command (a cui si accede premendo : in modalità Normal) si possono inserire comandi o impostazioni per l'editor
- "Aiuto, ho aperto vim per sbaglio, come esco?"
  - :wq per uscire salvando il file, :q! per uscire senza salvare
- Questo è l'utilizzo base, si può usare in modo più avanzato[3]

[3] `http://www.viemu.com/vi-vim-cheat-sheet.gif`

vim

"I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it."

- Editor modale
- I modi di utilizzo base sono: Normal, Insert e Command mode
  - Nella modalità Normal (a cui si accede premendo `Esc`) si può navigare all'interno del file di testo
  - Nella modalità Insert (a cui si accede premendo `i` in modalità Normal) si può modificare il file
  - Nella modalità Command (a cui si accede premendo : in modalità Normal) si possono inserire comandi o impostazioni per l'editor
- "Aiuto, ho aperto vim per sbaglio, come esco?"
  - :wq per uscire salvando il file, :q! per uscire senza salvare
- Questo è l'utilizzo base, si può usare in modo più avanzato[3]

[3] `http://www.viemu.com/vi-vim-cheat-sheet.gif`

emacs

"emacs is a great operating system, lacking only a decent editor"

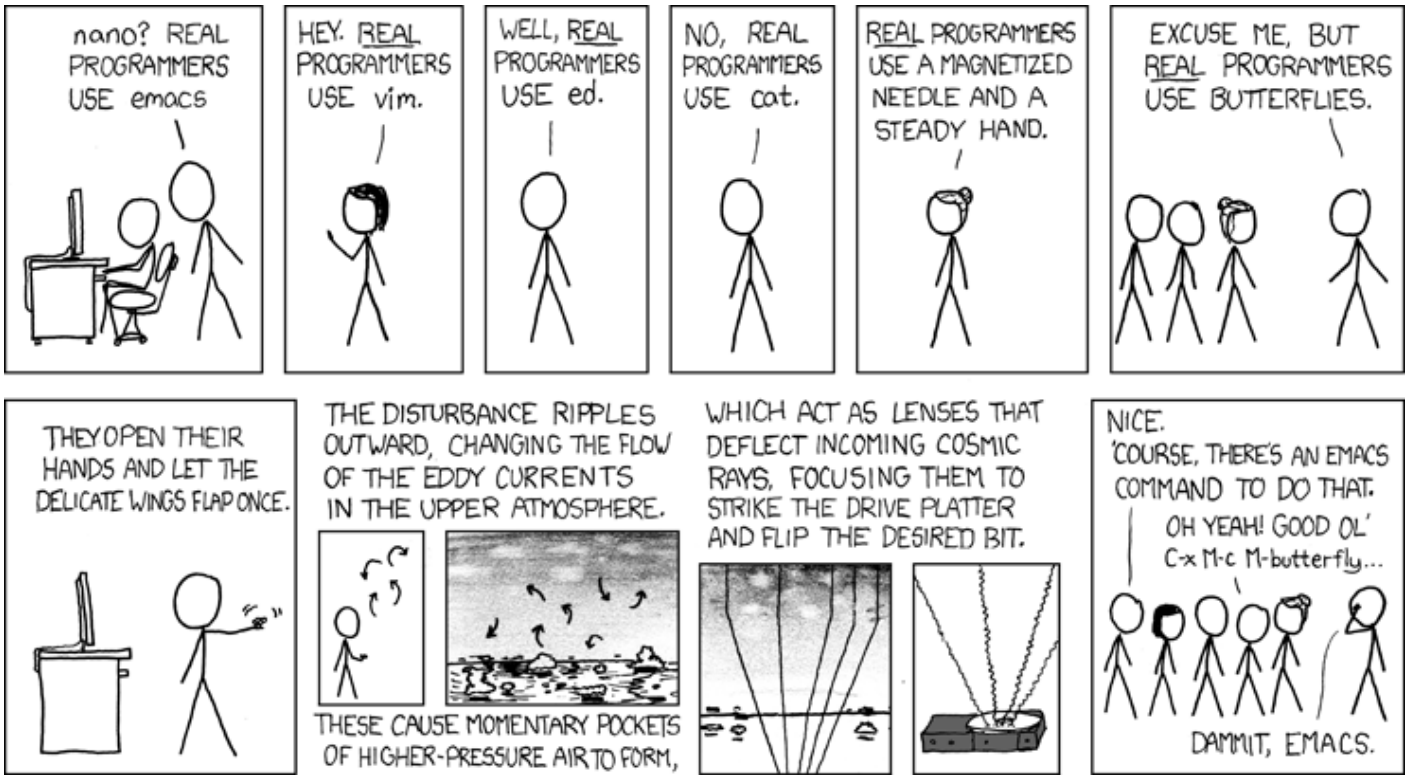
- GNU Emacs è un ambiente operativo che ha all'interno un interprete Lisp
- C'è una modalità per qualsiasi cosa (dal lettore mp3 allo psicologo)
- Il suo utilizzo si basa prevalentemente su combinazioni di tasti
- È estremamente personalizzabile e scriptabile
- Questa flessibilità si paga con un utilizzo di risorse maggiore

emacs

"emacs is a great operating system, lacking only a decent editor"

- GNU Emacs è un ambiente operativo che ha all'interno un interprete Lisp
- C'è una modalità per qualsiasi cosa (dal lettore mp3 allo psicologo)
- Il suo utilizzo si basa prevalentemente su combinazioni di tasti
- È estremamente personalizzabile e scriptabile
- Questa flessibilità si paga con un utilizzo di risorse maggiore

Ma quindi, qual è l'editor migliore?



Ma quindi, qual è l'editor migliore?

Se volete scatenare un flame tra utenti Linux, questa è un'ottima domanda da fare.
Non vi costa niente provarli tutti e scegliere qual è il migliore *per voi.*

Demo!

Ma quindi, qual è l'editor migliore?

Se volete scatenare un flame tra utenti Linux, questa è un'ottima domanda da fare.
Non vi costa niente provarli tutti e scegliere qual è il migliore *per voi.*

Demo!

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Bash++

- `cat` – concatena e stampa file su schermo
- `less` – stampa file su schermo con una finestra scorrevole, come la manpage
- `echo` – stampa il valore di un'espressione
- `locate` – cerca file in un database
  - Il database va aggiornato periodicamente con il comando `updatedb`
- `find` – cerca dei file all'interno di una gerarchia di cartelle

Standard streams

Ogni processo ha almeno 3 canali di comunicazione di default:

- stdin (0) – il canale che di default riceve l'input che l'utente scrive sul terminale
- stdout (1) – il canale che stampa l'output vero e proprio del programma, di default stampa su terminale
- stderr (2) – il canale che stampa gli errori del programma, anche questo di default stampa su terminale

Con pipe e redirezioni si possono connettere in vari modi questi canali, creando delle vere e proprie "catene di montaggio" di comandi bash che permettono di risolvere molti problemi unendo tanti blocchi semplici.

Standard streams

Ogni processo ha almeno 3 canali di comunicazione di default:

- stdin (0) – il canale che di default riceve l'input che l'utente scrive sul terminale
- stdout (1) – il canale che stampa l'output vero e proprio del programma, di default stampa su terminale
- stderr (2) – il canale che stampa gli errori del programma, anche questo di default stampa su terminale

Con pipe e redirezioni si possono connettere in vari modi questi canali, creando delle vere e proprie "catene di montaggio" di comandi bash che permettono di risolvere molti problemi unendo tanti blocchi semplici.

Standard streams

Ogni processo ha almeno 3 canali di comunicazione di default:

- stdin (0) – il canale che di default riceve l'input che l'utente scrive sul terminale
- stdout (1) – il canale che stampa l'output vero e proprio del programma, di default stampa su terminale
- stderr (2) – il canale che stampa gli errori del programma, anche questo di default stampa su terminale

Con pipe e redirezioni si possono connettere in vari modi questi canali, creando delle vere e proprie "catene di montaggio" di comandi bash che permettono di risolvere molti problemi unendo tanti blocchi semplici.

Standard streams

Ogni processo ha almeno 3 canali di comunicazione di default:

- stdin (0) – il canale che di default riceve l'input che l'utente scrive sul terminale
- stdout (1) – il canale che stampa l'output vero e proprio del programma, di default stampa su terminale
- stderr (2) – il canale che stampa gli errori del programma, anche questo di default stampa su terminale

Con pipe e redirezioni si possono connettere in vari modi questi canali, creando delle vere e proprie "catene di montaggio" di comandi bash che permettono di risolvere molti problemi unendo tanti blocchi semplici.

Standard streams

Ogni processo ha almeno 3 canali di comunicazione di default:

- stdin (0) – il canale che di default riceve l'input che l'utente scrive sul terminale
- stdout (1) – il canale che stampa l'output vero e proprio del programma, di default stampa su terminale
- stderr (2) – il canale che stampa gli errori del programma, anche questo di default stampa su terminale

Con pipe e redirezioni si possono connettere in vari modi questi canali, creando delle vere e proprie "catene di montaggio" di comandi bash che permettono di risolvere molti problemi unendo tanti blocchi semplici.

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando >> al posto di > in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando >> al posto di > in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando >> al posto di > in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando >> al posto di > in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando >> al posto di > in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Redirezione

- `comando < file` – connette stdin di un processo ad un file
- `comando > file` – connette stdout di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- `comando 2> file` – connette stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- comando `&> file` – connette stdout e stderr di un processo ad un file. Se il file esiste viene cancellato e sovrascritto
- Utilizzando $>>$ al posto di $>$ in uno degli ultimi 3 comandi si ottiene lo stesso risultato ma il risultato viene aggiunto al file se esistente
- `comando1 | comando2` – connette stdout di `comando1` a stdin di `comando2`

Esecuzione condizionale

- `comando1 && comando2` – esegue `comando2` se e solo se `comando1` ha successo (codice di ritorno $= 0$)
- `comando1 || comando2` – esegue `comando2` se e solo se `comando1` fallisce (codice di ritorno $\neq 0$)

Esecuzione condizionale

- `comando1 && comando2` – esegue `comando2` se e solo se `comando1` ha successo (codice di ritorno $= 0$)
- `comando1 || comando2` – esegue `comando2` se e solo se `comando1` fallisce (codice di ritorno $\neq 0$)

Alcuni filtri

- `cut`
- `sort`
- `uniq`
- `wc`
- `tee`
- `head e tail`
- `grep`

cut

- Estrae colonne delimitate da un carattere speciale da ogni riga di un file
- `-d` – specifica il delimitatore (default Tab)
- `-f` – specifica quale colonna estrarre (one-based)

sort

- Ordina le righe di un file
- `-k` – specifica quali colonne del file usare come chiave per l'ordinamento
- `-t` – specifica il delimitatore tra le colonne (default whitespace)

uniq

- Stampa le righe uniche di un file già ordinato
- `-c` – conta le occorrenze
- `-d` – mostra solo i duplicati
- `-u` – mostra solo i non duplicati

WC

- Conta righe, parole e caratteri
- `-l` – mostra solo il numero di righe
- `-w` – mostra solo il numero di parole
- `-c` – mostra solo il numero di caratteri

tee

- Connette il suo stdin allo stdin di due o più file
- Utile per mostrare l'output di un comando a schermo e darlo come input ad un altro comando

head e tail

- `head` mostra le prime 10 righe di un file, `tail` le ultime 10
- `-nX` – mostra le prime/ultime X righe
- `tail -f` – permette di "tenere d'occhio" un file a cui vengono continuamente aggiunte righe in coda (ad esempio un log)

grep

- Mostra le righe di un file che corrispondono ad un pattern (regular expression[4])
- `-v` – attiva il match invertito (mostra le righe che non corrispondono)
- `-i` – ricerca case insensitive
- `-c` – conta i match
- `-l` – mostra solo i nomi dei file con match
- `-r` – ricerca ricorsivamente all'interno dei file a partire da una cartella
- `-E` – usa le extended regular expression

[4] `https://xkcd.com/208/`

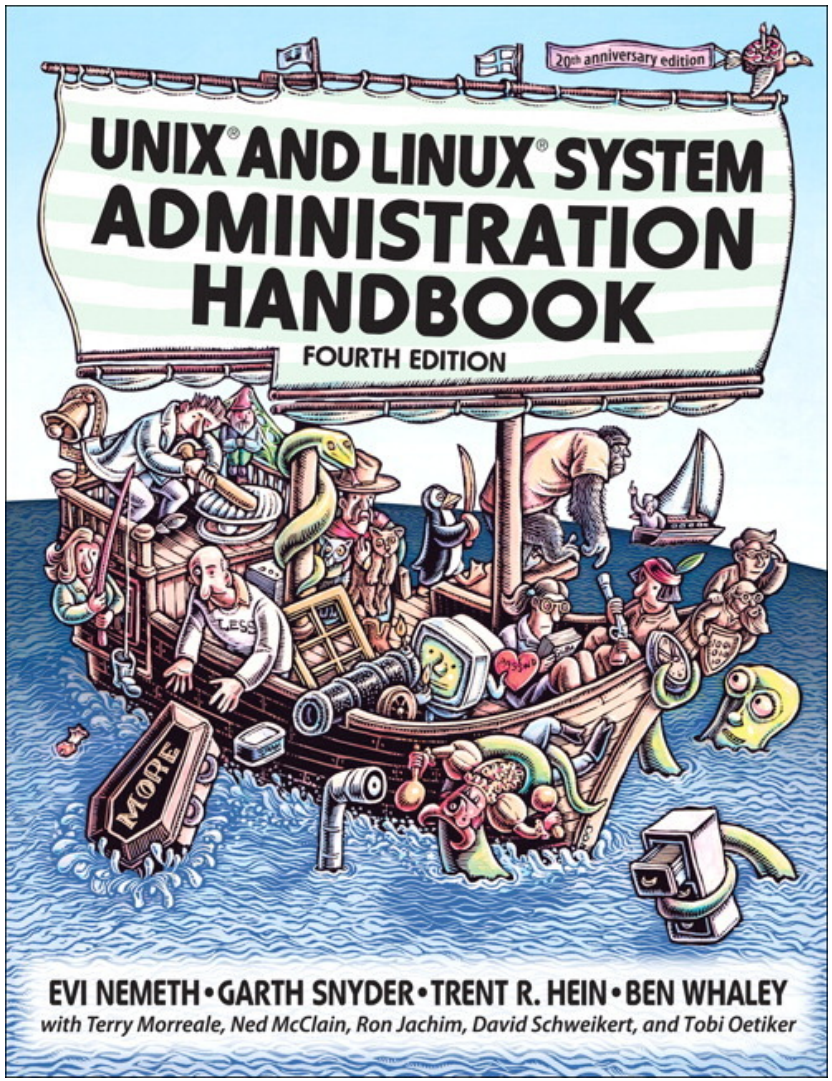
Domande?

Demo!

Domande?

Demo!

Per approfondire



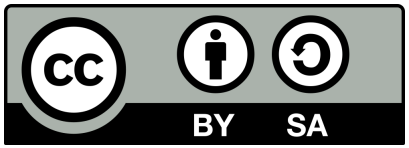
Google is your friend

Last call

Domande?

Goodbye and thanks for all the fish

Grazie per l'attenzione!



`http://www.poul.org`