

Sistemi Operativi

Leonardo Mengozzi

Titoletti indice link a rispettive sezioni, in alto a sinistra "[←Indice](#)" link a pagina Indice.

Contents

1 Bash	1
Comandi Variabili Ambiente	1
Comandi File speciali	1
Comandi Directory	1
Comandi Controllo Comandi	2
Comandi Scripting	3
Comandi Tilde Expansion	4
Comandi Privilegi	4
Comandi Subshell	5
Comandi Vari	5
2 Programmazione Concorrente	6
Comandi Multi-processo	6
Comandi PIPE	7
Comandi Multi-thread	7
2.1 SJF	8

1 Bash

Variabili Ambiente	Descrizione
PATH	Modificabile, sequenza di percorsi assoluti, divisi da ":", di directory contenenti eseguibili (lanciabili senza digitare path). Ricerca secondo ordine specificato in PATH, si ferma a primo eseguibile con nome uguale. Eventuale ErrorNotFoutd. Altre variabili d'ambiente: \$HOME, \$USER, \$SHELL, \$TERM.
env	Visualizza l'elenco delle variabili d'ambiente.

File speciali	Descrizione
/etc/passwd	Righe sono info ogni utente divise da ":".
/etc/shadow	Righe sono password utente codificate.
/etc/group	Righe sono info ogni gruppo divise da ":".
/usr/bin/passwd	Cambia la pass utente.

Directory	Descrizione
<code>cd percorso</code>	Sposta logicamente in una diversa directory, secondo un path assoluto o relativo.
<code>mkdir nomeDir</code>	Crea una nuova directory.
<code>touch nomeFile.estensione</code>	Crea un file vuoto nella directory corrente.
<code>rmdir nomeDir</code>	Rimuove una directory solo se è vuota.
<code>rm file dir</code>	Rimuove una directory vuota o un file. Parametri: <ul style="list-style-type: none"> • -r elimina ricorsivamente sotto cartelle e file. • -f non fa chiedere le autorizzazioni di eliminazione.
<code>mv file1 file2 dir</code>	Rinomina file1 in file2 o sposta file1 nella directory specificata.
<code>ls [nomefile]</code>	Visualizza i files/directory contenuti nella directory corrente. Parametri: <ul style="list-style-type: none"> • -a mostra anche file nascosti (anche <code>.</code>, <code>..</code>). • -l mostra più informazioni sui files. • -h rende i dati più leggibili. • -d fa applicare il comando alla directory e non ai file. • -R mostra ricorsivamente contenuto sotto directory. <p>Se specifico un file mi dice se esiste e mi da informazioni solo di lui.</p>
<code>pwd</code>	Visualizza il percorso assoluto, da <code>/</code> fino alla directory corrente.

Controllo Comandi	Descrizione
\	Disabla interpretazione per il carattere successivo, andata a capo, permettendo di stamparlo.
"..."	Delimita un argomento e non fa interpretare nessun comando a eccezione dell'espansione di variabili (\$..).
'...'	Delimita un argomento e non fa interpretare nessun comando.
pre{s1,...}post	<p>Stringa di testo racchiusa fra separatori (spazio, tab, a capo) con coppia di graffe (non precedute da \$) e senza separatori.</p> <p>Le stringhe racchiuse dalle graffe vengono composte con il preambolo (pre) e postscritto (post), che sono opzionali. Alternative:</p> <p>Sono annidabili (quelle più esterne eseguite per prime). Vengono eseguite prima le brace expansions delle variable expansions.</p> <ul style="list-style-type: none"> • $a_1..a_2$ lettere da a_1 a a_2 nell'alfabeto. • $n_1..n_2$ numeri compresi tra n_1 e n_2.
;	Separatore di più comandi scritti sulla stessa riga di comando e eseguiti dopo la terminazione del precedente.
*	Sostituito con una qualsiasi sequenza di caratteri anche vuota.
?	Sostituito con un singolo carattere (no spazio vuoto).
[c1c2...]	<p>Sostituito con solo uno dei caratteri specificati in elenco. Alternative:</p> <ul style="list-style-type: none"> • $a_1..a_2$ lettere da a_1 a a_2 nell'alfabeto. • $n_1..n_2$ numeri compresi tra n_1 e n_2. • [:digit:] una cifra. • [:upper:] un carattere maiuscolo. • [:lower:] un carattere minuscolo. <p>Annidabili</p>

Scripting	Descrizione
<code>echo testo</code>	Visualizza a video la sequenza di caratteri passata fino al primo "INVIO". Se passo <i>testo</i> si disabilita l'interpretazione dei caratteri speciali e andate a capo.
<code>nome=valore</code>	Simboli con nome e valore, stringa modificabile, alfanumerici case-sensitive. No spazi prima o dopo "=". Sono d'ambiente o ex-novo locali. Solo la bash in cui sono create le variabili le può usare. I programmi lanciati dalla bash hanno una pseudocopia della bash.
<code>export nomevar nomevar=valore</code>	Variabile d'ambiente. Un shell figlia riceve una copia modificabile che non influenza variabile d'ambiente del padre.
<code>unset nomevariabile</code>	Elimina una variabile esistente (vuota o no). Quotare ("...") sempre variabili per evitare errori con variabili vuote o inesistenti.
<code>\${nomeVar}</code>	Fa sostituire il nome della variabile con il valore. graffe opzionali se nome variabile seguito da uno spazio.
<code>#...</code>	Commeto.
<code>#!...</code>	Se indicato nella prima riga indica quale interprete deve eseguire lo script. Se non specificato usato quello corrente.
<code>comando1 comando2</code>	pipe (pseudo-file temporaneo): collega automaticamente l'output di un comando all'input di un altro. Unidirezionale.
<code>script.sh c1...</code>	<p>Sono un insieme ordinato di caratteri separati da spazi successivi al nome del programma. Sono immutabili dopo la sostituzione dei metacaratteri (*, ?, ecc).</p> <p>Riga di comando = nomeProgramma + parametri. Nella riga di comando gli elementi sono indicizzati da 0 (nomeProgramma).</p> <ul style="list-style-type: none"> • \$# Contiene il numero di parametri passati. • \$n Accede all'n-esimo parametro a partire da indice 0. • \$* Tutti argomenti concatenati e divisi da spazi. • \$@ Vettore di argomenti quotati ("...").

Tilde Expansion	Descrizione
<code>/...</code>	Tilde espansa con il percorso assoluto della home directory dell'effective user. Valido caso con solo <code>/</code> e solo <code>/.</code>
<code>userName/...</code>	Tilde e userName espansi con il percorso assoluto della home directory dell'utente specificato. Valido caso con solo userName/ .

Privilegi	Descrizione
<code>chmod u+x script.sh</code>	Modifica permessi file mediante formato numerico: u+x terna 0-7. Ogni numero è la somma dei valori associati ai permessi di r(4), w(2), x/s(1). Ordine: proprietario, gruppo, altri utenti. Può diventare un quartetto aggiungendo per primo l'identificatore numero dei privilegi di <i>setuid</i> , <i>setgid</i> , <i>sticky bit</i> .
<code>chgrp ???</code>	Modifica il gruppo di appartenenza di un file.
<code>chown newOwner nameFile</code>	Modifica il proprietario (e anche gruppo) di un file.
<code>ls -al nomeFile.estensione</code>	Mostra permessi, anche dei file nascosti. Interpretazione: 1°carattere tipo file (- file, d directory, c collegamento seriale, b device a blocchi), 9 caratteri successivi terzine di permessi (r read, w write, x/s execute) per proprietario, gruppo, altri utenti.
<code>whoami</code>	Dice all'utente corrente le sue informazioni.
<code>sudo comando</code>	fa eseguire il comando come amministratore, può essere chiesta userPass. Solo utenti gruppo sudo (gestito dall'admin) possono usarlo.

Subshell	Descrizione
<code>bash</code>	Crea una shell figlia (interattiva non di login). Eredita dal padre: dir. corrente, copia variabili d'ambiente. Non sono ereditate le variabili locali. Creata in automatico per comandi raggruppati, script, processi in background. I comandi built-in sono eseguiti in shell corrente/padre. <ul style="list-style-type: none">• -c script.sh non interattiva.• -l -login interattiva di login.
<code>var=val comando</code>	Scrivendo le assegnazioni prima dell'esecuzione di un comando si creano delle var. d'ambiente solo per l'imminente subshell. Non saranno ereditate da successive subshell.
<code>. source script.sh</code>	Esegue lo script nella shell corrente. Utile a impostare/modificare variabili shell. Ignorata prima riga opzionale e eseguito con interprete corrente.
<code>exit</code>	Termina bash corrente, elimina l'ambiente e sale alla padre.
<code>top</code>	Mostra in tempo reale processi in esecuzione e risorse di sistema usate.
<code>ps</code>	Mostra i processi in esecuzione. Con l'opzione -all vedo più informazioni (PID, PPID, ecc).
<code>set</code>	Visualizza sia variabili locali che d'ambiente della shell corrente (anche funzioni di shell). I parametri [re]setmano dei comportamenti della shell: <ul style="list-style-type: none">• +o comando Disabilita il comando (tipo history).• -o comando Abilita il comando (tipo history).• -a Successive variabili create/modificate diverranno d'ambiente e ereditate da shell figlie. Per [ri]definire variabili locali usare export -n variabile.• +a Successive variabili create/modificate diverranno locali e non ereditabili da shell figlie. (default).

Vari	Descrizione
<code>./ comando</code>	Esegue comando presente nella directory corrente (percorso relativo). Alternativa è inserire il percorso relativo o aggiungere il percorso assoluto alla variabile PATH.
<code>strace comando</code>	Dice la sistem-call usata.
<code>clear</code>	Pulisce la CLI. Non modifica variabili create.
<code>which comando</code>	Cerca in PATH il comando e se lo trova mostra il path in cui si trova.
<code>cat nomeFile.estensione</code>	Visualizza il contenuto del file.
<code>lsmod</code>	Elenca tutti i moduli attivi.
<code>modinfo nomeModulo</code>	Dice le informazioni sul modulo specificato.
<code>sudo modprobe nomeModulo</code>	Carica il modulo specificato.
<code>history</code>	Visualizza comandi, numerati, precedentemente eseguiti, anche da shell precedentemente chiuse. Con <code>!NUMERO</code> lancio il comando corrispondente nell'elenco di history. Con <code>!STRINGA</code> lancio il comando più recente che corrisponde alla stringa.

Directory "proc" info sui processi memorizzati con una dir. per ogni processo attivo, create e cancellate continuamente.

Istruzioni di controllo di flusso:

- `for ;variabile; in ;pool; ; do ... done` Dopo il for variabile è usabile.
- `while do done`
- `if then elif then else if`

Espressioni condizionate su file o variabile: `[condizione di un file]`. Valutazione di un'espressione matematica applicata a variabili d'ambiente: `((istruzioni con espressione))`

2 Programmazione Concorrente

Inclusioni necessarie:

- `#include <sys/types.h>` Definire tipi di dato speciali usati nelle chiamate di sistema.
- `#include <unistd.h>` Include funzioni e costanti del sistema operativo Linux/Unix.

Multi-processo	Descrizione
<code>fork()</code>	Crea un processo figlio. Condivide codice successivo alla fork e possiede una copia dei dati. Restituisce il pid del figlio al padre ($PID_F > 0$) e 0 al figlio ($PID_F = 0$) o un codice d'errore ($\neq 0$).
<code>exit(0)</code>	Termina un processo restituendo lo stato indicato come parametro (0 stato successo). Se padre termina prima del figlio non si possono liberare le risorse.
<code>wait(NULL)</code>	Fa attendere al processo la terminazione del processo figlio. Restituisce il pid del processo terminato.
<code>wait(pidProcesso)</code>	Fa attendere al processo corrente uno specifico processo.
<code>waitpid(pidProcesso, &status, statoAtteso)</code>	Fa aspettare un thread specifico con uno stato specifico.
<code>exec</code>	Sostituisce codice e dati a un processo. Non crea processi figlio.

Un processo possiede: Il proprio PID (gestito dall'os), il PID del processo figlio o 0 e il pid del processo "iziale" (gestito da descrittore di file).
Struttura base programma multi-processo:

```
int main() {
    pid_t pid;

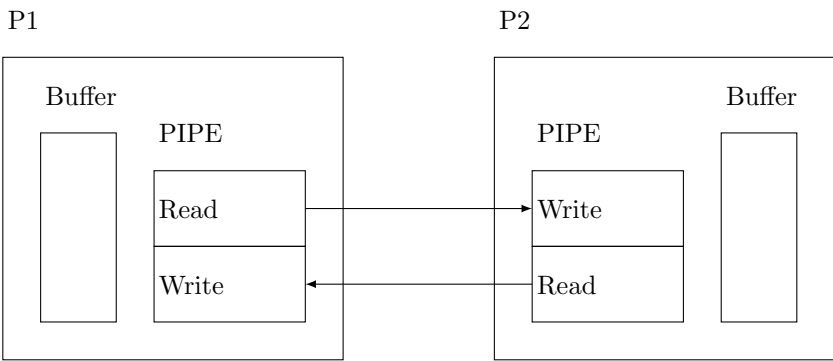
    pid = fork();

    if (pid < 0) { "gestione errore" }
    else if (pid == 0) { "processo figlio" }
    else { "processo padre" }
}
```

Ogni processo tiene referenza di un unico figlio, anche se ne può creare diversi. Quindi dopo ogni fork è buona cosa separare i flussi dei due processi.

PIPE	Descrizione
pipe(varPipe[2] : int) : int	Crea una pipe unidirezionale. Su varPipe[0] si leggerà. Su varPipe[1] si scriverà. Restituisce -1 in caso di errore, 0 se creazione avviene con successo.
close(varPipe[0 1])	Chiude una estremità della pipe.
read(varPipe[0], buffer, bufferSize)	Blocca esecuzione in attesa di dati da leggere.
strcpy(buffer, "...")	Prepara il buffer con il messaggio da inviare.
write(varPipe[1], buffer, sizeBuffer+1)	Scrive nella pipe.

Permette di comunicare fra processi correlati usando sistem-call e i descrittori di file. Il collegamento esiste fino a eliminazione esplicita o del processo.



Ogni processo ha un buffer di caratteri e una pipe (array di due celle) per scrivere e leggere con l'altro processo i dati contenuti nel buffer.
Nota: I processi sono visti come file, per questo le operazioni si chiamano come quelle dei file.

Multi-thread	Descrizione
nomeFunzioneThread(arg : void*) : void*	Funzione assegnata da eseguire a un thread. Necessita questa firma specifica per accettare e restituire qualsiasi tipo di dato. Serve eseguire un cast esplicito.
pthread_create(&varThread, &pthreadAttribut, tFunction, &args)	Crea un nuovo thread dentro al processo corrente. I parametri sono: <ol style="list-style-type: none"> 1. Variabile tipo thread. 2. Puntatore a struttura di attributi del thread. Default è NULL. 3. Funzione che sarà eseguita dal thread. 4. Puntatore a struttura contenente parametri usati dalla funzione.
pthread_join(&varThread, NULL)	Fa attendere processo la fine del thread indicato.

Struttura base programma multi-processo:

```
void *tFunction(void *args) {...}
int main() {
    pthread_t thread;
    ... args = ...;

    pthread_create(&thread, NULL, tFunction, &args);
}
```

Nota: La creazione di un processo è più lenta della creazione di un thread perchè nella prima bisogna creare un intero file descriptor, mentre nella seconda parziale.

2.1 SJF

Calcolo approssimato CPU Bust: $T_{n+1} = \alpha t_n + (1 - \alpha)T_n$.

- t_m tempo n-esimo CPU burst. Storia recente.
- T_n previsione prevista. Storia passata.
- α peso storia recente e passata.

Calcolo Media esponenziale: $T_{n+1} = \sum_{j=0}^n \alpha(1 - \alpha)^j t_{n-j} + (1 - \alpha)^{n+1} T_0$.