Sorgenti, Binari e Pacchetti

(Compilazione e pacchettizzazione per GNU/Linux)

Augusto Scatolini (webmaster@comunecampagnano.it) (a.scatolini@linux4campagnano.net)
Miniguida n. 134
Ver. 1.0 Luglio 2011

```
25
26  int
27  main
28      (
29      int     argc,
30      char*   argv[]
31      )
32  {
33      ParseTextOptions(argc, argv);
34
35      if (!THXMDIServer::WillBeMDIServer(THXApp::GetAppSignature(), argc, argv))
36          {
37          return 0;
38          }
39
40      THXApp* app = new THXApp(&argc, argv);
41      assert( app != NULL );
42
43      app->Run();
44      return 0;
45  }
```

esempio di codice sorgente

CODICE SORGENTE

Il punto di forza, rivoluzionario, del Software Libero, tipico di GNU/Linux, è la libertà di vedere, gestire, studiare, modificare e compilare il codice sorgente dei vari programmi. Questo è possibile grazie alla licenza GNU/GPL

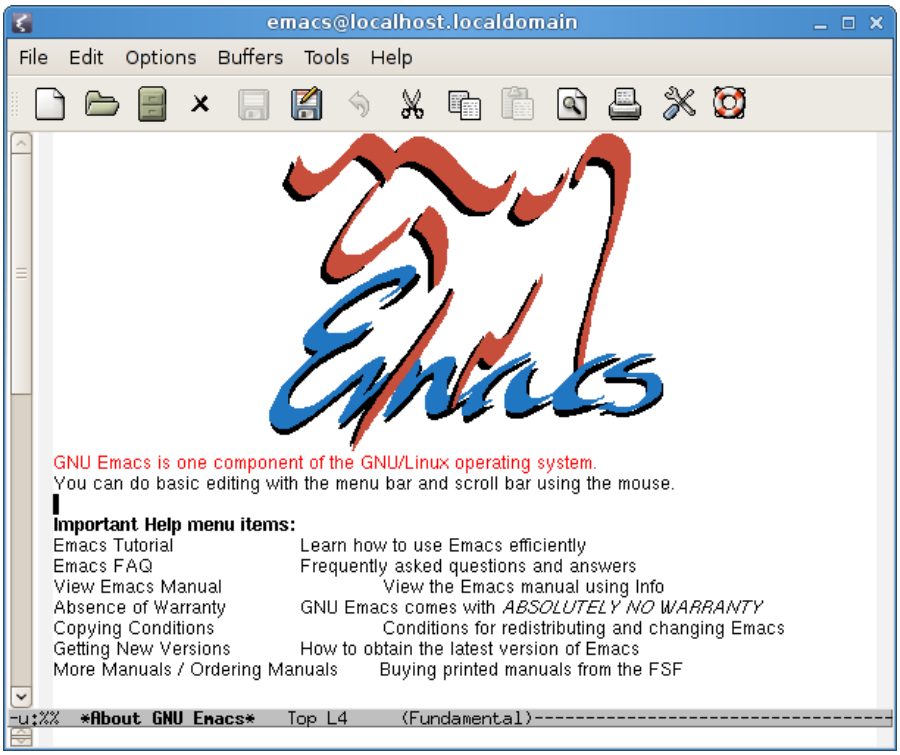
Esclusi i programmi il cui codice viene interpretato “al volo” da un apposito programma con funzioni appunto di interprete come avviene per l'HTML, l'ASP o il PHP, quando un programmatore inizia a scrivere un programma, praticamente, prende un editor di testo ASCII e inizia a scrivere righe e righe di codice, a volte migliaia di righe di codice.

Questi codici possono essere scritti in vari linguaggi di programmazione come C o **C++** o altri ancora come **Java**, **Python**, **Ruby** o **Perl** e successivamente vengono tradotti, permanentemente, da un apposito programma chiamato compilatore.

Il processore di un Computer capisce solo codice macchina (detto a basso livello o binario), il programmatore invece capisce, dopo lunghi studi, solo linguaggi non binari – detti ad alto livello – come C++ appunto.

Ecco perché è necessario scrivere il codice (in un linguaggio comprensibile al programmatore) e poi renderlo binario tramite il processo della compilazione.

Gli editor di testo più utilizzati per scrivere codice sotto GNU/Linux sono **Emacs** (scritto direttamente dal fondatore del Software Libero **Richard Stallman**, **Pico** e **VI** (ora VIM).



Il compilatore di riferimento sotto GNU/Linux è GCC (GNU Compiler Collection) anche questo scritto da Stallman.

Esempio di codice scritto in C++

```
  for (;;) {
    // l'utente preme un tasto finchè non ne preme uno di quelli precedentemente indicati
    do {
      move = getch();
      } while(move!=UP && move!=DOWN && move!=LEFT && move!=RIGHT && move!=EXIT && move!
=DRAW &&move!=CANCEL && move!=CLEAR && move!=SAVE);
  // in base al tasto premuto viene eseguita un'istruzione diversa
  switch (move) {
    case UP: mx=-1; my=0; break;
    case DOWN: mx=1; my=0; break;
    case LEFT: mx=0; my=-1; break;
    case RIGHT: mx=0; my=1; break;
    case DRAW:
      mx=0; my=0;
      if(draw_switch) draw_switch = false;
      else draw_switch = true;
      cancel_switch = false;
      break;
    case CANCEL:
```

In realtà il processo di compilazione è un insieme di processi in sequenza:

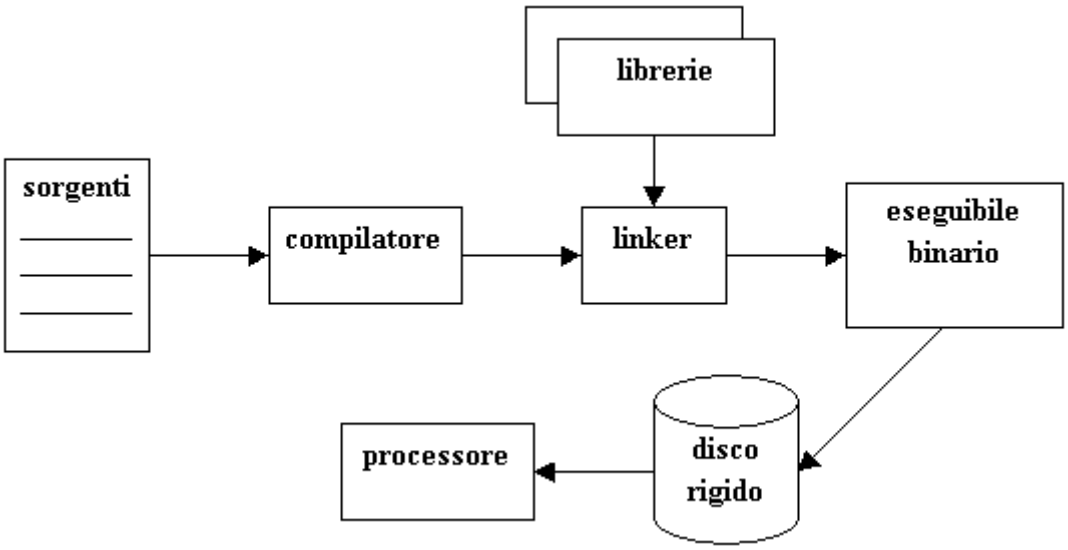
- **pre-processor** – prende in ingresso il file sorgente e esegue delle direttive come copiare il contenuto del file, espandere le definizioni, e altro
- **compiler** – traduce il codice C++ in Assembler
- **optimizer** – esegue le ottimizzazioni per rendere il programma più veloce
- **linker/loader** – unisce più codici sorgenti in un unico eseguibile

Tralasciando tutte le specifiche tecniche delle varie fasi che compongono il processo di complilazione, si può dire che una volta scritto il programma si passa alla fase di **debugging** (per correggere tutti gli eventuali errori detti **bug**) e infine si passa alla compilazione definitiva.

Nota. *Un buon programmatore come Stallman, osservando lo stile con il quale è stato scritto un codice riesce a capire la decade nella quale è stato scritto.*

PROGRAMMI BINARI

Quindi dal **CODICE SORGENTE** tramite la **COMPILAZIONE** si ottiene il **BINARIO ESEGUIBILE**



In realtà un'applicazione complessa come **LibreOffice** (per esempio) è composta da molti file binari, molte librerie e diverse **DIPENDENZE** (ovvero altri programmi o librerie propedeutiche e necessarie)

Tutti questi file, poi, devono essere posiszionati ognuno in una specifica posizione del **File System** ovvero in specifiche **Directory** (o cartelle per gli utenti Windows).

Ecco perché per coloro che decidono di utilizzare i binari piuttosto che compilarsi autonomamente i sorgenti sono stati predisposti dei **PACCHETTI** che contengono tutti i file binari e tutte le librerie e che quando vengono lanciati (i pacchetti) vanno ad installare (copiare) tutti i singoli file nelle rispettive directory.

Quando è opportuno compilarsi autonomamente i codici sorgenti?

- Un motivo potrebbe essere quello di voler modificare il codice sorgente (avendone le capacità) adattandolo alle proprie particolari esigenze e quindi personalizzarsi l'applicazione
- Un altro motivo potrebbe essere quello di adattare l'eseguibile, del sorgente non modificato, al processore e all' architettura del proprio Computer per trarne dei vantaggi in termini di velocità e performance.
- Un altro motivo potrebbe essere la mancanza di pacchetti per la propria DISTRIBUZIONE.

Nota. *Conosco personalmente utilizzatori di GNU/Linux che non installerebbero mai un binario compilato da terzi.Questi installano solo binari compilati in proprio.*

PACCHETTI

Un programma di gestione dei pacchetti è una collezione di strumenti che automatizzano il processo di installazione, aggiornamento, configurazione e rimozione dei pacchetti software di un computer. Il termine è più comunemente utilizzato in relazione a sistemi Unix-like (tipo Unix), particolarmente GNU/Linux, dato che questi sistemi poggiano molto più pesantemente su di esso, con migliaia di pacchetti in una sola normale installazione. In tali sistemi, il software è distribuito in pacchetti, generalmente incapsulati in un singolo file. I pacchetti spesso includono anche altre importanti informazioni, come il nome completo, versione, e fornitore del software, informazioni sul checksum, ed una lista di altri pacchetti, conosciuti come dipendenze, che sono necessarie al software per funzionare correttamente. I sistemi di gestione dei pacchetti sono incaricati del compito di organizzare tutti i pacchetti installati in un sistema e mantenere la loro usabilità. Questi sistemi raggiungono questo scopo usando varie combinazioni delle seguenti tecniche: Verifica del checksum dei file per evitare differenze tra le versioni locali ed ufficiali di un pacchetto; Semplici strumenti per l'installazione, l'aggiornamento, e la rimozione; Gestione delle dipendenze per la distribuzione del software funzionante da un pacchetto; Controllo degli aggiornamenti per fornire le ultime versioni dei software, che spesso includono riparazioni di difetti ed aggiornamenti di sicurezza; Raggruppamento di pacchetti a seconda della funzione per aiutare l'utente ad eliminare la confusione durante l'installazione ed il mantenimento.

http://it.wikipedia.org/wiki/Sistema_di_gestione_dei_pacchetti

In termini di distribuzioni GNU/Linux, ovvero Sistemi Operativi con annesso un certo numero di applicativi per le più svariate finalità, si possono condiderare la Debian la Red Hat.
Ognuna di queste distribuzioni ha poi reso possibile (grazie alla licenza GNU/GPL) una serie pressochè infinita di distribuzioni cosiddette derivate.
La differenza tra queste due grosse famiglie consiste appunto nella gestione dei pacchetti che sono chiamati DEB e RPM, rispettivamente.

deb è il nome del formato, nonché estensione, dei pacchetti utilizzati dalla distribuzione Debian e dalle sue derivate, come Ubuntu, Kubuntu ecc.

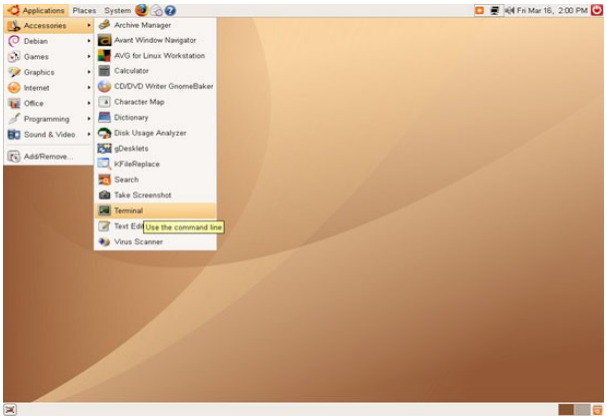
Rpm è un sistema di gestione dei pacchetti, utilizzato per installare, verificare, aggiornare e disinstallare i pacchetti in alcune distribuzioni del sistema operativo GNU/Linux; il nome deriva da Red Hat Package Manager, e vede tra i suoi principali utilizzatori Red Hat, Fedora, Mandriva, Suse e loro derivate.

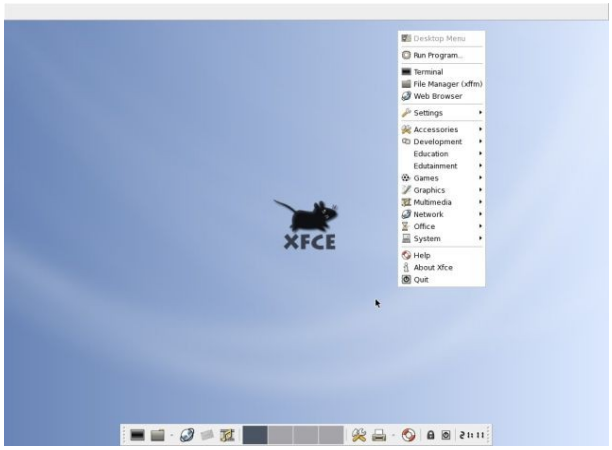
YUM (Yellow dog Updater, Modified) è un sistema di gestione dei pacchetti open source a riga di comando per i sistemi operativi GNU/Linux compatibili col formato RPM.

METAPACCHETTI

I metapacchetti, tipicamente DEB, contengono le istruzioni per scaricare e installare ambienti grafici più o meno pesanti.

I metapacchetti più comuni installano ambienti come GNOME, KDE, XFCE, LXDE, ecc.





Per l'utente (anche) medio di sistemi GNU/Linux è importante sapere che può installare da sorgenti (previa compilazione), da binari eseguibili o da pacchetti (soluzione più diffusa).
Ma proprio perchè la installazione da pacchetti è la più diffusa bisogna sapere cos'è un pacchetto e quale tipo di pacchetto si deve installare (di quale famiglia).

Nota. *Esiste un programma ALIEN, da usare con cautela, capace di convertire un pacchetto RPM in DEB e viceversa.*

FINE

altre miniguide su

http://www.comunecampagnano.it/gnu/miniguide.htm
oppure direttamente su http://miniguide.tk

sito consigliato: http://www.linux4campagnano.net