15 Linguaggi di programmazione

Probabilmente un utente principiante non avrà la necessità immediata di usare dei linguaggi di programmazione, ma visto che Linux è un sistema operativo su cui si trovano innumerevoli compilatori, dedicheremo questa lezione a fare una lista delle possibilità che vengono offerte.
Con in più, quasi come bonus, alcune nozioni sull'utilizzo di un editor di testi di grandissima qualità quale Emacs.

I programmi che noi usiamo in un qualsiasi sistema operativo sono essenzialmente di tre tipi: compilati, interpretati o misti. In ogni caso il programmatore deve scrivere del "codice sorgente", cioé un insieme di istruzioni che dicono al computer cosa deve fare quando esegue il programma stesso (operazioni aritmetiche, lettura di input, presa di decisioni a seconda del valore di qualche variabile, etc).
Quello che cambia è cosa succede una volta scritto il codice: se usiamo un linguaggio compilato, allora dobbiamo dare il codice sorgente in pasto (cioè in input) ad un programma chiamato appunto compilatore. Questo programma ci restituirà un file in linguaggio macchina (o comunque un linguaggio di basso livello), che potrà essere eseguito immediatamente, scrivendone il nome su una shell, oppure aggiungendo un'icona al menù delle applicazioni, e andandoci a fare click con il mouse. Un linguaggio compilato è per esempio il C.
Un programma interpretato, invece, deve essere dato in input, ogni volta che lo vogliamo eseguire, ad un altro programma chiamato interprete. Di norma, poiché un interprete ogni volta deve tradurre tutte le istruzioni di un programma, il risultato è notevolmente più lento che un programma compilato. E' tuttavia vero che, una volta riscritto l'interprete per tutti i tipi di computer e per tutti i sistemi operativi, qualsiasi programma da esso interpretabile sarà utilizzabile da chiunque (mentre i programmi compilati, per quanto detto nel primo modulo di lezioni, sono indissolubilmente legati ad una combinazione tipo di processore/sistema operativo). Un linguaggio puramente interpretato è per esempio il Perl.
Infine, un buon compromesso tra velocità e portabilità è dato dai linguaggi misti: il codice sorgente viene dato ad una sorta di compilatore, che lo traduce in un linguaggio intermedio, di livello alto abbastanza perché sia facile implementare un interprete su ogni piattaforma. A questo punto, ogni volta che si vuole eseguire il tutto, è necessario dare il codice semi compilato in input ad un interprete, che però avrà vita piuttosto semplice, essendo le istruzioni da interpretare sicuramente più vicine al computer che al linguaggio di programmazione. Le prestazioni dunque saranno anch'esse una via di mezzo, come accade per il più famoso linguaggio di questo tipo: JAVA.

Alessandro Urpi - Manuali.Net © 2003 Tutti i diritti riservati

Per fare un esempio forse più comprensibile, è come se una persona ad esempio giapponese, amante della cucina mondiale, dovesse consegnare una lista di alcuni prodotti che vuole recapitatigli periodicamente ad un italiano. A questo punto, supponendo che entrambi i personaggi parlino solo la lingua della loro nazione, il compratore può far tradurre la lista in italiano da un interprete giapponese, mandarla, ed essere (ragionevolmente) sicuro che verrà capita dal venditore. Se però nel futuro cambierà nazione in cui si approvigiona, scegliendo ad esempio la Francia, dovrà farsi tradurre di nuovo la lista, e così via. E' un po' il caso della compilazione.
Può invece mandare la lista in giapponese, facendola tradurre in loco. In questo modo può inviare in tutto il mondo sempre la stessa lista, delegando la traduzione a degli interpreti. Tuttavia, poiché non è certo che la lista sia sempre la stessa, ogni volta questa dovrà venire tradotta, anche se è uguale alla volta precedente, facendo andare il compratore incontro a costi ingenti. E' il prezzo che si paga per usare interpretazione.
Infine, potrebbe farsi tradurre la lista, per esempio, in spagnolo. Prima di tutto perché é una delle lingue più parlate al mondo, e quindi qualora si rivolgesse a mercati latini non pagherebbe niente, e poi perché è facilmente comprensibile da chi parla una lingua neolatina (ma anche dagli anglofoni, rispetto al giapponese!). Probabilmente i venditori che si vedranno recapitata una tale lista, potranno avvalersi di consigli di amici, e soluzioni economiche di questo tipo.

Tornando alla scrittura di programmi, il primo passo da fare è la scrittura del codice sorgente. Ovviamente in una lezione non posso entrare nel merito di nessun linguaggio specifico, ma quasi tutti i linguaggi non sono altro che una lista di comandi che il computer dovrà eseguire. I comandi base sono gli assegnamenti di valori a variabili (che possono essere visti come oggetti adatti a contenere dati), il comando condizionale (se è vero qualcosa, allora fai questo, altrimenti fai quest'altro) e il ciclo (fai questo finché non si verifica una certa condizione). Ogni linguaggio poi offre tutta una serie di cose in più per rendere più facile la scrittura di programmi complessi, ma tutto (o quasi) si riduce a scorciatoie che si basano sui costrutti appena presentati.
Il sorgente è un testo normale, che va scritto in ASCII, cioè senza abbellimenti dati da caratteri, paragrafi, etc. I files di testo sono i files che di solito si aprono con il Blocco Note in ambiente Windows.
In Linux abbiamo tantissimi programmi, chiamati editor, che ci permettono di editare files di testo. Nella distribuzione da noi installata sono presenti gedit (molto simile al blocco note di Windows) ed il suo parente stretto kedit. Il primo si trova nel menù delle applicazioni di GNOME nella categoria "Accessori", sotto la voce "Editor di testi". Il secondo invece è uno strumento di KDE (ma notate che si può usare senza alcun problema da GNOME), che si può invocare da una shell (sicuramente è anche in qualche parte del menù KDE). Questo programma permette appunto di scrivere, salvare e riaprire files di testo, ma è abbastanza scomodo per grandi programmi (ma anche per programmi che superano le cento linee di codice).
Per questo, sempre nel menù delle applicazioni, nella sezione "Programmazione", è presente Emacs, che è un programma famosissimo fra gli utenti Unix/Linux.
E' un programma piuttosto vecchio, ma proprio questo lo fa estremamente stabile, testato e sicuro. D'altra parte un editor di testi non deve avere delle caratteristiche che con il tempo diventano superate! Inoltre Emacs è un programma che permette di fare un'infinità di cose (credo che le persone al mondo che lo sanno usare in tutto e per tutto siano davvero poche), e facilita la vita in molti casi.
Giusto a titolo di conoscenza personale, è un programma che può essere usato per gestire la posta, per navigare in Internet in modalità testuale, e per eseguire altri programmi!
A noi tuttavia interessa la possibilità di scrivere testi, per sfruttare la quale non bisogna fare niente di particolare.
Una volta aperto, ci verrà mostrato un messaggio di benvenuto, dopo il quale è possibile aprire un file (eventualmente creandolo) con l'icona che mostra il foglio bianco, oppure dal menù "File" ->

Alessandro Urpi - Manuali.Net © 2003 Tutti i diritti riservati

"Open file". Notate che non appare nessuna finestra da cui scegliere il file, ma bisogna digitarne il nome nella riga più in basso della finestra stessa di emacs. Come piccolà comodità, possiamo usare il tasto [Tab] come se fossimo su una shell.
Dopo aver digitato un nome (per esempio prova.java), emacs tenta di capire che tipo di programma stiamo scrivendo, in modo da offrirci i suoi servizi. Notate che, se avete scritto prova.java, alla destra del nome del file (appena sopra la riga in cui avete digitato il nome) è apparsa la dicitura: "Java Abbrev", perché il programma ha capito che tipo di file è. Tra le mille comodità offerte ci sono la colorazione delle parole chiave, in modo da capire meglio il codice che stiamo scrivendo (provate per esempio a digitare "import" oppure "int", che sono parole chiavi di Java, per vederle colorate all'istante), e l'indentazione automatica (un file si dice indentato se segue delle regole per l'andata a capo e la tabulazione dettate dalla struttura del programma, in modo da capirne velocemente la struttura).
Emacs permette la gestione contemporanea di più files, in modo automatico: aprendo un nuovo file chiamato, per esempio, "prova.pl", oltre ad accorgersi che vogliamo editare del codice Perl, la finestra da cui partivamo viene nascosta, ma non perduta. Nel menù "Buffers" c'è una lista di tutti i file attivi, compreso quello contenente tutti i messaggi che emacs visualizza.
I comandi mostrati nei menù non sono che un'infinitesima parte di tutti quelli offerti dal programma: provate a richiedere l'aiuto in linea (in inglese), per avere un'idea della complessità del programma!
Tuttavia, tutto quello che può essere necessario ad un utente inesperto vi è presente, comprese le funzionalità di copiatura/taglio/incollatura, di "undo" per annullare l'ultima operazione fatta, e di "redo" per rifarla (come curiosità: in Linux, ogni volta che si seleziona del testo con il mouse, questo viene automaticamente copiato, e può essere incollato con un semplice click del tasto centrale del mouse sul punto in cui vogliamo l'incollatura).

A questo punto, è bene chiudere questa lezione con una rapida carrellata dei compilatori/interpreti di linguaggi di programmazione offerti GRATUITAMENTE da Linux: la maggior parte di essi sono di ottimo livello, e permottono la creazione di programmi estremamente efficienti, e non hanno niente da invidiare a prodotti venduti a carissimo prezzo. Nessuno di essi è installato se si è scelta un'installazione di tipo Desktop personale: apprenderemo nel prossimo modulo di lezioni ad installarli in modo molto semplice.

Un primo linguaggio di programmazione, molto semplice ma veramente utile, è offerto dalla stessa shell: con esso è possibile scrivere dei file cosiddetti "batch", che vengono interpretati ed eseguiti dalla shell stessa. E' molto utile quando si devono compiere ripetutamente operazioni molto simili tra di loro, ma anche quando bisogna fare qualcosa per ogni file contenuto in una directory (per esempio rinominarlo), e così via. E' detto il linguaggio integrato (o built-in in inglese) nella shell.

Il kernel di Linux è scritto in C: come potrebbe dunque mancare il compilatore di questo linguaggio tanto amato/odiato? Il gcc (Gnu C Compiler) è molto efficiente e stabile, e genera codice piuttosto ottimizzato. Con esso vengono offerti i compilatori di C++ (chiamato g++) e di Fortran (f77).

Vista la sua popolarità, anche Java è facile da trovare. Tuttavià ci sono in questo caso dei problemi di licenza: essendo un prodotto SUN non rilasciato con licenza GNU, ed essendo il compilatore/macchina virtuale della SUN l'unico davvero completo, non è possibile ancora scrivere del codice "free" in Java. Tuttavia diversi gruppi di programmatori stanno lavorando ad una versione Gnu della macchina virtuale, e si prevede di averne delle versioni complete a breve (ci sono diverse versioni incomplete, di cui la più famosa è forse kaffe).

Tra i linguaggi puramente interpretati ce ne sono due usatissimi per la creazione di siti web

Alessandro Urpi - Manuali.Net © 2003 Tutti i diritti riservati

dinamici, e nati in ambiente Linux: Perl e PHP. Entrambi hanno degli ottimi interpreti in tale ambiente, e sono facilmente usabili (pur di averne una minima conoscenza).

Infine, poiché .NET di Microsoft sta prendendo piede, è bene precisare che la piattaforma esiste anche in Linux con due diversi progetti: .GNU e mono della Ximian (che tra l'altro offre una versione molto accattivante di GNOME). In entrambi i progetti esiste un compilatore di C#, anche se ovviamente ancora non è del tutto stabile (come non lo è quello Microsoft per ambiente Windows).

Ci sono ovviamente altre centinaia di linguaggi di programmazione, creati per soddisfare le più diverse esigenze ed i più disparati programmatori. Sappiate che la probabilità che esista un compilatore per uno di essi in Linux è molto alta, essendo Unix/Linux un sistema operativo molto usato da chi fa ricerca, che quindi spesso deve implementare qualche programma per fare simulazioni, o per produrre dei dati.

Alessandro Urpi - Manuali.Net © 2003 Tutti i diritti riservati