

## **Unix. Una cultura di condivisione**

*Alessandro Rubini*

Progetto GNU Pavia

Email: rubini@gnu.org

### *Abstract*

*L'autore approfondisce alcune caratteristiche tecniche dei sistemi operativi di tipo Unix mostrando come la sua elevata modularità sia una delle ragioni principali della sua efficienza. Viene poi esposta brevemente la storia del progetto GNU e di Linux.*

### **Premessa**

Innanzitutto, cos'è Unix? Unix è un progetto nato nel 1969, che nasce dall'esperienza che alcuni programmatori si erano fatti sul sistema Multics. Multics ("Multiprogrammed Computer System") era un sistema che doveva essere "tutto per tutti", e si è rivelato un fallimento; conteneva però al suo interno alcune buone idee che sono state riutilizzate in seguito nel sistema Unics (Uniprogrammed ComputerSystem). Tale sistema, in seguito ribattezzato Unix, è stato scritto per gioco da tre programmatori di AT&T che volevano utilizzare un vecchio calcolatore abbandonato in un angolo, una macchina con 16kilobyte di memoria.

Le stesse persone hanno scritto anche il linguaggio C, uno dei linguaggi oggi più usati. Il linguaggio C è stato inventato apposta per scrivere Unix: il sistema Unix è scritto in C e anche il compilatore C è scritto in C (come d'altronde il compilatore Java è scritto in Java: questo tipo di cose che si appoggiano su se stesse piacciono molto agli informatici).

A causa delle sue lontane origini, Unix viene a volte definito come tecnologia arcaica, come se le cose vecchie fossero per forza sbagliate. Personalmente mi fido di più di strumenti che si sono affinati e perfezionati nel tempo piuttosto che strumenti scritti da zero che non hanno avuto una verifica sperimentale di affidabilità. Val la pena di ricordare che anche le auto più moderne si basano sul motore a scoppio, una tecnologia di cento anni fa.

## **1. Il concetto di file**

Il sistema Unix è nato per essere semplice. Questo in parte è una conseguenza della sua origine ludica: un sistema che deve funzionare su un vecchio elaboratore e che sia divertente da scrivere e usare deve per forza essere semplice. Ma la sua semplicità dipende anche dal fatto che le idee migliori sono sempre quelle più semplici. La "semplicità" di Unix sta principalmente nella sua elevata modularità: il sistema operativo contiene centinaia di piccoli mattoncini che svolgono operazioni elementari sui dati e tali mattoncini possono essere composti a discrezione dell'utente per svolgere i compiti più disparati. Questa modularità di base del progetto di Unix permette una flessibilità estrema nell'utilizzo del sistema.

Alla base della modularità di Unix sta il concetto di file: un file è un canale di comunicazione dati, praticamente un "tubo" attraverso cui scorrono i dati. Il caso più noto, quello del file su disco, è solo un caso particolare di un concetto più astratto: il file su disco può essere usato come sorgente o destinazione del flusso di dati, ma il programma vede comunque quasi sempre i dati come un flusso.

Nei sistemi Unix quasi ogni risorsa viene messa a disposizione come un file: la tastiera è un file di ingresso, la stampante è un file di uscita, un dischetto è come un file su disco, con l'unica particolarità che la sua lunghezza è costante (1.44MB, per esempio).

Quando viene eseguito un programma, il sistema gli predispone il collegamento con tre file, chiamati "stdin", "stdout", "stderr". Durante l'esecuzione il programma legge i dati di ingresso da "stdin", scrive i risultati su "stdout" e manda eventuali errori su "stderr", senza bisogno di sapere a cosa corrispondono "fisicamente" i tre file che vengono visti come "tubi" dove scorrono i dati.

Di solito, quando un utente digita i comandi al terminale, tutti i tre "tubi" sono collegati al terminale (tastiera e video). È possibile però concatenare l'elaborazione di più programmi, in modo che i dati prodotti da uno siano visti come dati di ingresso dal programma successivo. Questa situazione di solito viene chiamata "pipe", cioè "tubo", e gli errori di tutti i programmi della catena di solito rimangono diretti verso il terminale. È possibile anche collegare ingressi e uscite dei programmi verso una connessione di rete, senza che i programmi stessi se ne accorgano.

Naturalmente ai canali di comunicazione, ai "file", si possono anche collegare dei file su disco. La struttura dati che ospita i "file" convenzionali si chiama filesystem, e in ambiente Unix il filesystem ospita anche oggetti che non corrispondono ai file cosiddetti "regolari", quelli che contengono dati. Il filesystem ospita anche i punti di accesso alle periferiche, alle connessioni di rete e a strutture di comunicazione di tipo "FIFO", in pratica una "pipe" simile a quella descritta precedentemente. L'utilizzo di tutte queste risorse è uniforme, in quanto le applicazioni (o gli utenti) possono accedere a qualunque entità ospitata nel filesystem con gli stessi meccanismi: le funzioni `open()`, `read()`, `write()`, `close()`.

Poiché i programmi hanno a disposizione molteplici modalità di comunicazione con il mondo esterno e con altri programmi, i singoli programmi messi a disposizione dal sistema sono solitamente molto semplici e predisposti per l'utilizzo di standard-input e standard-output. Questo permette all'utente di avere la più ampia libertà nell'utilizzo di questi programmi, componendoli secondo le sue esigenze. Nonostante sia effettivamente più facile usare il mouse per agire su menu predefiniti per scegliere tra opzioni predisposte dagli autori dell'applicazione, quello che occorre per esprimere nuove idee e risolvere nuovi problemi è una sintassi. Nel caso di Unix troviamo la sintassi della linea di comando, ormai sconosciuta alla maggior parte degli utenti di calcolatore. La disponibilità della linea di comando offre una potenzialità enorme all'utente, senza per questo pregiudicare l'utilizzo di ambienti grafici e interfacce a menu.

In figura ho riportato un esempio di linea di comando:

```
sort data | a2ps | bold | rsh prozio "cat > /dev/lp0"
```

Questo comando dice: metti in ordine il contenuto del file "data" e manda il risultato al comando "a2ps" (ascii to postscript: converte il testo puro in linguaggio postscript, un linguaggio di definizione della pagina usato dalle stampanti di qualità e altri sistemi). Il risultato di questa conversione viene mandato al programma "bold", un programma di due righe che ho scritto per utilizzare i caratteri courier-bold invece che courier. L'uscita del comando "bold" spedisce attraverso la rete al calcolatore chiamato "prozio" e lì indirizzalo verso la stampante. Un altro programma che ho scritto utilizza la linea di comando per cercare il numero di telefono di una persona dalla mia rubrica personale e stamparlo sulla scheda audio, così basta appoggiare la cornetta del telefono sull'altoparlante per fare il numero.

Nonostante la sintassi di una linea di comando richieda una fase di apprendimento da parte dell'utente, nessuno strumento grafico permette questa flessibilità. Unix, in questo senso, è una "cultura di condivisione", perché ogni piccolo pezzetto della struttura del sistema è progettato per la massima integrazione con il resto del sistema e, tramite la rete, con il resto del mondo.

## **2. La rete Internet**

Proprio su un modello di questo tipo, incentrato sulla modularità e sulla facilità di condivisione dei dati, è stata progettata la rete Internet. La rete è stata "scritta" su macchine Unix adottando un modello di comunicazione client-server: in una comunicazione via rete una delle due macchine svolge il ruolo di chi offre un servizio mentre l'altra macchina opera da cliente. Dicevo che Internet è stata scritta perché in pratica la rete è definita da un insieme di protocolli di comunicazione. Questi protocolli sono liberamente accessibili a chiunque, per cui chiunque può creare una nuova implementazione di un servizio, che risulterà compatibile con le altre implementazioni.

I progettisti della rete hanno sempre avuto la massima cura nella differenziazione dei ruoli tra protocollo e implementazione, che ha un parallelo nella separazione tra meccanismi offerti e politiche d'uso associate a questi meccanismi: in genere in ambiente Unix e in ambiente Internet i protocolli di comunicazione non fanno alcuna assunzione su quale sia il loro utilizzo. Per questo si può spedire posta elettronica da una vasta gamma di programmi, grafici e testuali, come pure da sistemi completamente automatici, come quelli di gestione delle liste di posta: la politica d'uso rimane scorrelata dal meccanismo di spedizione dei messaggi.

I protocolli che definiscono Internet sono ratificati dall'IETF (Internet Engineering Task Force), un organismo internazionale composto da ricercatori. I documenti dell'IETF si chiamano RFC (Request For Comments) e definiscono ogni singolo aspetto dei meccanismi di comunicazione (ma non le politiche associate). La comunicazione tra calcolatori viene gestita, tramite questi protocolli, attraverso diversi livelli logici. Siccome i meccanismi di consegna dei dati operano solo ai livelli più bassi, è possibile estendere le funzionalità della rete senza modificare i macchinari (i "router") che si occupano della consegna dei dati tra i calcolatori remoti.

È interessante notare come le tecnologie di rete non richiedono grosse potenze di calcolo, proprio perché tutto è nato per la massima semplicità e modularità, unito al fatto che la rete è nata negli anni 70-80, quando la potenza di calcolo a disposizione era molto minore di quella attuale.

Scrivere un programma server non è affatto una cosa complicata: come si diceva un server è semplicemente un programma che legge da standard-input e scrive su standard-output, spesso senza nemmeno sapere che è collegato ad una rete. In figura sono rappresentati un semplice server "ad uso scolastico" e altri due programmi, un server e un client per la trasmissione dei log di sistema attraverso la rete (a differenza del primo, questi due sono programmi che uso realmente per l'amministrazione della mia rete).

Anche la "tecnologia web", spesso spacciata come "tecnologia del terzo millennio", è di una semplicità estrema: in una delle mie attività, per esempio, ho realizzato un server web che gira su un 386-SX con 2MB di memoria e 2 di disco, una macchina che già da alcuni anni è considerata obsoleta. Un altro esempio interessante è lrp.org (Linux Router Project), software per utilizzare vecchi elaboratori di scarto come router, i macchinari che collegano le varie tratte della rete Internet.

### 3. L'ambiente grafico

Ovviamente anche Unix è dotato di un ambiente di lavoro grafico (chiamato X-Windows o semplicemente X) nonostante la maggior parte dei programmatori Unix prediligano la linea di comando per i motivi esposti prima. Come nel caso della rete Internet, l'ambiente grafico Unix è estremamente modulare e flessibile, oltre ad essere perfettamente integrato con la rete stessa. Tutto il sistema grafico è, per definizione, trasparente rispetto alla rete e neutrale rispetto alla architettura di sistema. "Trasparente rispetto alla rete" vuol dire che le applicazioni possono girare su una macchina e usare il display di un'altra macchina; "neutrale rispetto all'architettura" vuol dire che tutto il sistema non dipende in alcun modo dal tipo di calcolatore e di processore che si sta usando.

Perciò, se per avere un ambiente grafico Windows devo comprare un PC e per avere un ambiente grafico Mac devo comprare un Mac, X funziona su una varietà incredibile di macchine diverse. Per esempio, per disegnare i lucidi delle mie applicazioni uso un programma che gira su processori Sparc ma lo faccio visualizzare sul monitor del mio PC o del portatile.

#### 4. Unix oggi

Unix, Internet, X, sono cose che sono state scritte negli anni '80 da programmatori entusiasti del loro lavoro; tutti avevano libero accesso al codice sorgente (il programma nella forma in cui è stato scritto dal programmatore e quindi nella forma in cui può essere modificato e migliorato) e ciò permetteva un continuo arricchimento del patrimonio informatico. Con gli anni però le cose sono cambiate e molte aziende hanno iniziato a realizzare versioni proprietarie dei programmi, di cui non è accessibile il codice sorgente. Il mondo Unix quindi si è ramificato in decine di versioni del sistema, tutte in qualche modo simili ma tutte in qualche modo diverse. C'è quindi un sistema distribuito da Sun che è leggermente incompatibile con quello distribuito da IBM, entrambi sono leggermente incompatibili con il sistema HP, eccetera.

Ma come è potuto succedere che un sistema si ramificasse in tanti sistemi diversi? Il motivo principale è a causa della licenza d'uso adottata per il codice di Unix. Unix è stato sviluppato in parte da AT&T (quindi sotto licenza proprietaria) e in parte dalle università americane, in particolare Berkeley (quindi sotto licenza libera); la ramificazione è potuta avvenire perché AT&T dopo aver fatto circolare liberamente il codice ha venduto i diritti a varie aziende, ma anche le parti "libere" permettevano di venire incorporate in prodotti non-liberi.

La licenza d'uso dei programmi delle università permette di prendere senza restituire; è permesso cioè derivare un nuovo programma proprietario da un programma libero.

Perciò, anche se gli Unix sono tutti simili, ci sono delle forti incompatibilità interne perché ogni azienda ha sviluppato il sistema in una direzione diversa dalle altre e quello che è stato aggiunto non è stato restituito alla comunità di programmatori. Il risultato è un gran guazzabuglio di versioni diverse dello stesso sistema che in pratica ha danneggiato la diffusione del sistema stesso.

## 5. GNU

Non chiamavamo il nostro software "software libero", poiché questa espressione ancora non esisteva, ma si trattava proprio di questo.

La situazione cambiò drasticamente all'inizio degli anni '80. [...] I moderni elaboratori di quell'epoca, come il VAX o il 68020, avevano il proprio sistema operativo, ma nessuno di questi era libero: si doveva firmare un accordo di non-diffusione persino per ottenerne una copia eseguibile.

Questo significava che il primo passo per usare un computer era promettere di negare aiuto al proprio vicino. Una comunità cooperante era vietata. La regola creata dai proprietari di software proprietario era: «se condividi il software col tuo vicino sei un pirata. Se vuoi modifiche, pregaci di farle».

L'idea che la concezione sociale di software proprietario - cioè il sistema che impone che il software non possa essere condiviso o modificato - sia antisociale, contraria all'etica, semplicemente sbagliata, può apparire sorprendente a qualche lettore. Ma che altro possiamo dire di un sistema che si basa sul dividere utenti e lasciarli senza aiuto?

Allora cercai un modo in cui un programmatore potesse fare qualcosa di buono. Mi chiesi dunque: c'erano un programma o dei programmi che io potessi scrivere, per rendere nuovamente possibile l'esistenza di una comunità?

La risposta era semplice: innanzitutto serviva un sistema operativo. [...]

Essendo un programmatore di sistemi, possedevo le competenze adeguate per questo lavoro. [...] Scelsi di rendere il sistema compatibile con Unix, in modo che fosse portabile, e che gli utenti Unix potessero passare facilmente ad esso.

Nel 1983, Richard Stallman, amante del software libero e disgustato dalla proliferazione di sistemi operativi proprietari, ha lanciato il progetto GNU (GNU's Not Unix: GNU non è Unix). I fini del progetto GNU sono la realizzazione di un sistema operativo completo che sia anche libero. Un programma è libero quando tutti possono avere accesso al codice sorgente, modificarlo, anche rivenderlo.

GNU è simile a Unix a causa dell'elevata modularità di Unix, che permette di sostituire le parti proprietarie del sistema una per volta. GNU non è Unix perché è un sistema libero e perché risponde a standard qualitativi più elevati.

Nel lucido sono riportate le parole di Stallman stesso che riporta come è nato il progetto GNU.

## 6. Linux

E finalmente arriviamo a Linux. Linux è un progetto del '91, sviluppato da Linus Torvalds dopo aver lavorato su Minix. Minix è uno dei tanti Unix disponibili sul mercato, scritto da un professore universitario come strumento didattico: il codice sorgente completo del nucleo di Minix è disponibile per essere studiato e capito. Ma non è permesso modificarlo e ridistribuirlo.

Linus aveva apportato dei miglioramenti alla sua versione di Minix, ma non era autorizzato a distribuirle. Per questo motivo decise di fare il suo kernel Minix, che doveva essere migliore di Minix e completamente libero per tutti gli utenti. Nel lucido ho riportato un brano scritto da Linus nell'ambito di uno scambio epistolare con Tanenbaum (scambio avvenuto in rete e disponibile in rete).

È interessante notare come Linus stesso sottolinei che Linux, il kernel, è una minima parte del sistema completo; un sistema operativo infatti contiene anche molte applicazioni, il compilatore, il debugger, l'editor di testi, l'ambiente grafico. Quasi tutte queste componenti erano già disponibili come software libero, in gran parte grazie al lavoro del progetto GNU; quello che mancava era solo il kernel. Nonostante Linus abbia iniziato da solo, ha presto trovato molti altri programmatori che lo hanno aiutato in questa grande avventura che in pochi anni ha portato al Linux che conosciamo oggi.

Il termine "Linux" quindi si riferisce propriamente solo al nucleo del sistema, il kernel. Il nucleo è quello che si occupa della gestione del calcolatore: processore, memoria, connessioni in rete, periferiche di ingresso e uscita. Il ruolo del nucleo è quello di rendere disponibili tutte le risorse fisiche della macchina tramite un'interfaccia ben definita, le "chiamate di sistema", attraverso cui le applicazioni possono usare la macchina da calcolo sottostante.

Quando si installa "Linux" su un calcolatore in pratica si installa un sistema GNU che usa Linux come kernel. La differenza è sostanziale e va ricordata, anche se spesso si usa comunque il termine "Linux" per indicare tutto il sistema operativo. Per realizzare tale sistema operativo hanno lavorato migliaia di persone, ciascuna mettendo il proprio piccolo contributo. Nello stesso kernel Linux c'è il lavoro di centinaia di persone, spesso contributi molto piccoli ma importanti all'interno del progetto nel suo complesso.