# Tcl pad, un editor di testo estendibile, incorporabile, con interfacce di text-processing

Antonio Bonifati

20 Luglio 2000

## Indice

# 1 Introduzione e caratteristiche principali

*Tcl pad* (abbreviato in *tpad* e pronunciato come *tea pad*) è un editor di testo, facile da usare, portabile, estendibile con nuove funzionalità, incorporabile in altre applicazioni e munito di front-end grafici sulle più comuni interfacce di text processing.

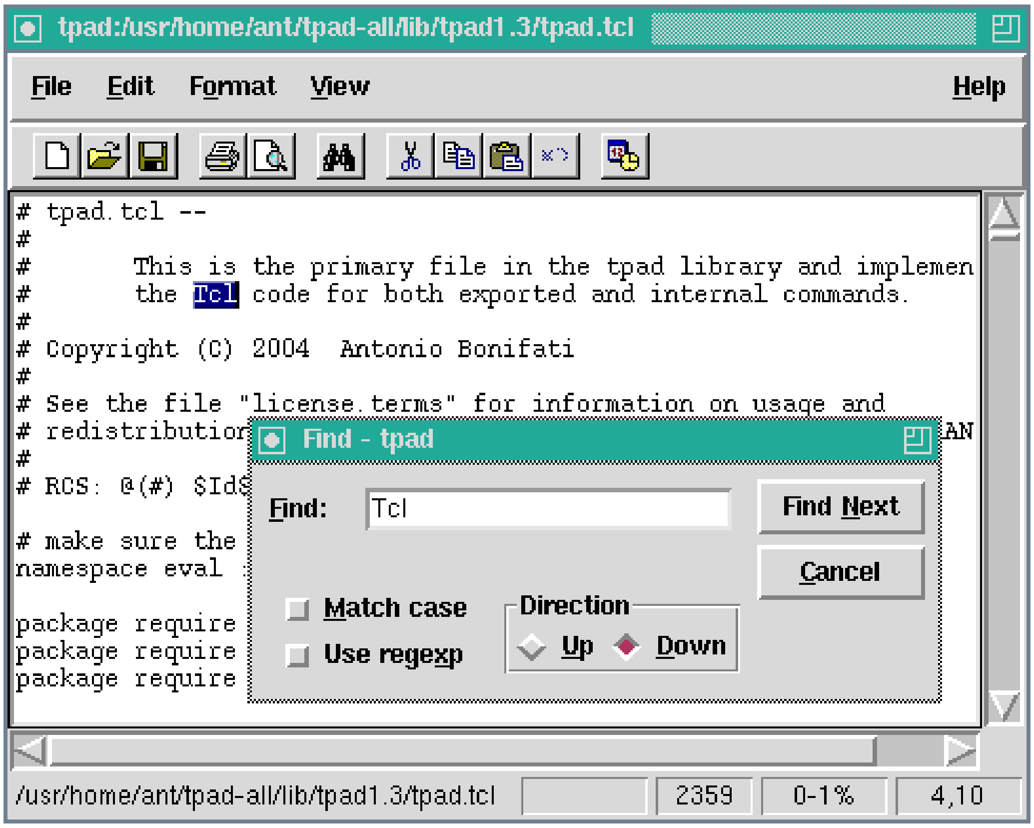


Figura 1: Una schermata dell'interfaccia di default di tpad

Tpad è stato scritto interamente nel linguaggio Tcl/Tk ed è distribuito in formato di script sorgenti in formato portabile ASCII a 7-bit. In aggiunta alla versione portabile, sono anche disponibili versioni pacchettizzate per i più comuni sistemi operativi, che automatizzano le operazioni di installazione, disinstallazione e configurazione iniziale.

A differenza della maggior parte degli editor che sono per lo più programmi a sé stanti, tpad è invece una libreria di funzioni di editing. Funzionalità e interfacce che le richiamano sono state tenute separate per massimizzare la riusabilità del codice. Tramite una API molto semplice basata su Tcl nel ruolo di linguaggio "colla" (vedere sezioni 3 e 5), il codice di tpad può essere riutilizzato per costruire rapidamente editor integrati in altri programmi, come ad esempio debugger orientati allo schermo, ambienti di sviluppo integrati e programmi di posta elettronica.

Tpad può essere usato come editor esterno, operante in una finestra separata da quella dell'applicazione che lo usa (finestra che può essere sia modale che non), oppure può essere facilmente integrato nella applicazione stessa, operando nella sua stessa finestra principale. Per scambiare il testo che deve essere editato tra tpad e l'applicazione, è possibile scegliere tra l'uso di file temporanei e l'editing diretto di una variabile di memoria dell'applicazione stessa (stringa lunga di Tcl).

Tutti i menu, le combinazioni di tasti attive, le funzioni di risposta alle interazioni dell'utente con l'interfaccia grafica *(callback functions)*, il modo in cui gli elementi dell'interfaccia grafica (detti *widget*, abbreviazione di *window object*) sono combinati tra loro possono essere ridefinite, qualora necessario, editando un unico file di codice Tcl di dimensioni contenute. Nella versione corrente questo file consta di meno di 400 linee, avendo incluso nel conteggio anche le numerose linee vuote e di commento presenti.

Diverse istanze di tpad possono essere attive contemporaneamente, in modo indipendente tra loro. All'atto della creazione una istanza può essere configurata per agire come editor oppure come solo visualizzatore di testo. Una istanza di tipo viewer ha una interfaccia simile ad una di tipo editor, ma non sono presenti tutte quelle funzionalità che comportano la modifica del contenuto del testo.

Nella definizione dell'interfaccia grafica di default e nella scelta delle funzioni di editing fondamentali, ho deciso di rendere tpad molto simile al ben noto Blocco Note di Microsoft Windows XP (detto Notepad nell'edizione in lingua inglese). I menu sono organizzati alla stessa maniera, le combinazioni di tasti sono le stesse, e la toolbar è una copia addirittura a livello di pixel di Wordpad, non essendo presente nel Notepad originale (vedere Figure 1 e 2). Per questa somiglianza esteriore, tpad può essere definito brevemente come un clone "avanzato" di Notepad.

tpad è un progetto open-source soggetto alla Licenza Generale Pubblica GNU [1] che viene gestito tramite il portale collaborativo per lo sviluppo software Sourceforge.net. È inoltre entrato a far parte della collezione delle port del sistema operativo FreeBSD [2] e viene distribuito come pacchetto contrib RPM dalla distribuzione Linux Mandriva (precedentemente nota come Mandrake).

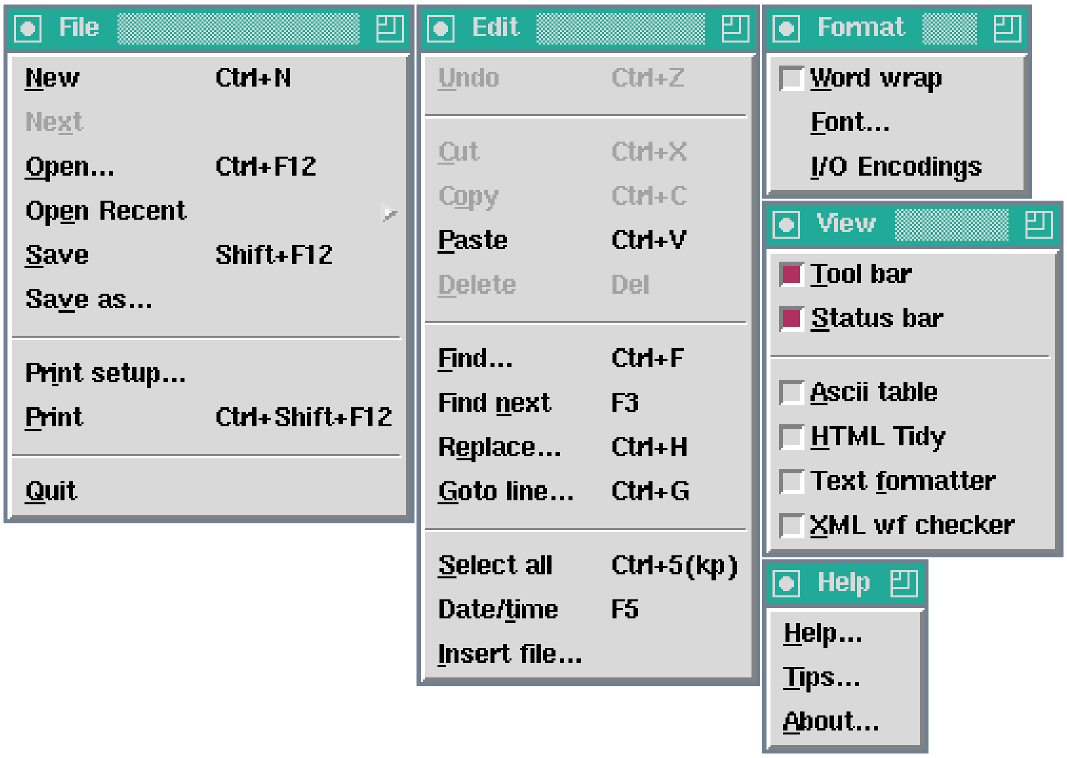


Figura 2: I menu staccabili contengono tutte le funzioni implementate

## 2 Internazionalizzazione e localizzazione

Tpad è una applicazione pienamente I18N/L10N-compliant. Tutti i menu e i messaggi nei dialoghi possono essere facilmente tradotti a runtime e il programma è in grado di ricevere in input ed elaborare del testo in tutte le principali lingue del mondo.

Inizialmente ho scritto i cataloghi dei messaggi nelle sole due lingue che conosco, l'Inglese e l'Italiano. Poi con l'aiuto di altri collaboratori madrelingua, è stata aggiunta la traduzione in Francese, Spagnolo, Tedesco ed Ebraico. Sto cercando attualmente altri traduttori. Per essere davvero internazionale, tpad dovrebbe supportare almeno tutte le maggiori lingue del mondo, incluse quelle ideografiche come Cinese, Giapponese e Coreano e naturalmente anche il Russo.

Per agevolare la diffusione internazionale del programma, la lingua base di tpad è l'Inglese. In questa lingua ho scritto tutta la documentazione allegata al programma (che si divide in documentazione per gli utenti e documentazione per i programmatori), i numerosi commenti che fanno parte del codice e le pagine web del sito ufficiale. Questi documenti non saranno mai oggetto di traduzione, in quanto, oltre al notevole lavoro iniziale che sarebbe richiesto, diventerebbe poi troppo oneroso mantenere aggiornate tutte le traduzioni ad ogni modifica. L'Inglese è anche la lingua che uso per comunicare, prevalentemente via email, con i traduttori.

Attualmente mantengo sincronizzati tramite il protocollo di aggiornamento remoto rsync [3], tre mirror completi del sito web di tpad, disposti fisicamente in Italia, negli USA e in India.

I contenuti principali del sito [4] sono:

- breve descrizione del progetto e screenshot del programma
- informazioni sulle versioni disponibili
- collegamenti per scaricare il software e dettagli su come installarlo
- indicazioni per i contributor
- patch per eventuali bug scoperti dopo il rilascio della versione stabile

Per ottenere l'ultima versione di sviluppo (ovvero instabile) del programma è disponibile un servizio di CVS anonimo messo a disposizione da Sourceforge.net.

# 3 Il linguaggio Tcl/Tk

Tcl [5] (Tool Command Language, pronunciato come *tickle*) è un linguaggio di comandi ad alto livello, non proprietario, interpretato, facilmente estendibile, dinamico, portabile e incorporabile. Appartiene alla categoria dei linguaggi di scripting [6] non tipati piuttosto che a quelli di sistema e ha subito influenze da altri linguaggi quali C, Lisp e C-shell.

Una sintassi molto semplice e uniforme, basata solo su comandi (detti anche procedure) aventi parametri con nome, rende Tcl molto facile da usare e imparare. Le strutture di controllo, le definizioni di nuove procedure sono esse stesse dei comandi e non vi sono operatori a livello sintattico. Per la valutazione delle espressioni si usa infatti un apposito comando *expr* in grado di interpretarle come parametri. Ad un comando possono essere forniti altri comandi come argomenti, sotto forma di stringhe, che il comando tipicamente passerà all'interprete Tcl per ottenerne l'esecuzione.

A scopo illustrativo, segue un semplice esempio di applicazione Tcl/Tk:

```
button .b -text Hello! -font {Times 16} -command {puts hello}
pack .b
```

Questa è una banale applicazione Hello World! grafica, che consta di un unico tasto la cui pressione provoca la stampa della stringa `hello` sul terminale. Il creatore di Tcl, John K. Ousterhout, mostra [6] come lo stesso esempio implementato in Java richiederebbe 7 linee di codice in due metodi differenti (senza contare le altre 21 linee di codice necessarie per creare un *applet*) e ben 25 linee di codice in tre procedure utilizzando C++ con le Microsoft Foundation Classes (avendo escluso ancora una volta la corposa infrastruttura di codice necessaria per qualsiasi applicazione C++/MFC).

Tcl incorpora già molte delle funzionalità utili in un linguaggio di scripting moderno e avanzato. La lista seguente è sicuramente incompleta, ma rende l'idea delle sue potenzialità:

- comandi per operare sulle stringhe, le liste e gli array associativi
- supporto per espressioni regolari avanzate
- namespace dinamici per creare contesti separati per comandi e variabili
- package per facilitare il riuso, la modularizzazione e la distribuzione del codice
- dynamic loading [1]
- tracing delle variabili e dei comandi
- supporto per la programmazione event-driven, socket e thread [2]
- error trapping
- comandi portabili per la gestione del file-system e delle porte seriali
- funzionalità di internazionalizzazione e supporto Unicode [7]

I comandi predefiniti supportano solo tre tipi di strutture dati (stringhe, array associativi e liste) e un solo tipo di dati, le stringhe, che sono sufficienti per molte applicazioni. Tuttavia, mediante opportune estensioni, molte delle quali già disponibili e pronte all'uso, si possono aggiungere altri tipi e strutture dati in maniera efficiente e senza complicare la sintassi di base di Tcl. Ad esempio l'estensione BLT mette a disposizione il tipo dati vector (vettori di valori floating point), importanti per molte applicazioni di calcolo.

[1] Permette di caricare estensioni su richiesta a run-time, riducendo i tempi di startup delle applicazioni.

[2] Tcl è thread-safe; nessun comando Tcl crea conflitti quando vi sono più thread concorrenti in esecuzione.

Tcl puro è ideale per scrivere velocemente piccole applicazioni o prototipi. Se combinato con il toolkit Tk [8] (pronunciato come *tik*) e altre estensioni se necessario, può essere usato con successo anche per sviluppare applicazioni di media/grande dimensione e dotate di interfaccia grafica. Le ultime versioni di Tcl hanno tutte le funzionalità richieste per creare applicazioni aziendali mission-critical. Tramite estensioni è supportato l'accesso ai più popolari sistemi di gestione di basi di dati, come Sybase, Oracle e MySQL.

Tk è una estensione di Tcl che rende molto semplice creare sistemi interattivi perchè mette già a disposizione i componenti riusabili necessari; come visto in piccolo nel semplice esempio del programma "hello", questi vengono poi assemblati e integrati per costruire l'interfaccia grafica tramite l'uso del linguaggio Tcl. Le GUI create con Tk sono portabili tra Unix, Macintosh, Windows e assumono l'aspetto e i comportamenti dell'interfaccia di default del sistema su cui girano, che è quella a cui l'utente è maggiormente abituato.

Da uno script Tcl è possibile eseguire altri programmi esterni, redirigendone l'input e/o l'output, come si fa negli script di shell Unix. Esistono due shell tipo Unix contenenti l'interprete Tcl: tclsh e wish. Tclsh è una shell che utilizza come comandi interni quelli di Tcl e come esterni quelli messi a disposizione dal sistema operativo ospite. Wish sta per *WIndowing SHell* e, rispetto a tclsh, è dotata di finestra grafica e aggiunge i comandi Tk. Con wish gli amministratori di sistema possono crearsi velocemente script dotati di interfacce grafiche sui comandi Unix, che consentono di automatizzare e semplificare le operazioni ripetitive [9].

Per la distribuzione di applicazioni non open-source in forma solo compilata, è possibile ricorrere ad un compilatore Tcl [10] che genera bytecode portabile (codificato in formato testuale ASCII a 7 bit). Questo non è però al momento ancora disponibile per la piattaforma Macintosh.

Tcl è anche un linguaggio di scripting universale per Internet. Può essere usato come linguaggio di script lato client embedded in HTML, similmente a JavaScript o come plugin esterno del browser, per creare appletti (detti tclet) analoghi agli applet Java. Come Java, consente di eseguire in tutta sicurezza gli script scaricati, grazie ad un sofisticato meccanismo di controllo e assegnazione dei privilegi detto Safe-Tcl. Esiste poi Tclhttpd, che è una implementazione in Tcl puro e con buone performance di un server web basato sul protocollo HTTP e dotato di un framework per la generazione di pagine con contenuto dinamico. Poichè Tcl è un linguaggio facilmente incorporabile in altre applicazioni, anche Tclhttpd può essere incorporato, ad esempio allo scopo di aggiungere rapidamente un'interfaccia web alle vecchie applicazioni legacy.

L'interprete Tcl è strutturato come una libreria di funzioni che può essere incorporata in altri linguaggi, sia di sistema (C, C++, Java) sia di scripting (Python). L'interprete è disponibile con licenza open-source sulle tre piattaforme più diffuse (Unix, Macintosh, Windows).

Supponiamo, per esempio, di voler sviluppare un'applicazione di foglio elettronico in uno dei suddetti linguaggi di sistema: anziché inventare un altro linguaggio per le formule e la definizione delle macro funzioni, si può usare Tcl come linguaggio incorporato. Così facendo non è necessario definire un nuovo linguaggio e scrivere il corrispondente parser e interprete in quanto questi vengono già forniti con poco sforzo includendo la libreria Tcl. Tutte le funzionalità generiche di un linguaggio di comandi, come le strutture di controllo, le variabili, le procedure sono già fornite da Tcl: occorre solo estendere i comandi di base Tcl con i comandi addizionali necessari alla particolare applicazione che si sta scrivendo. Questi nuovi comandi possono essere scritti in C, C++, Java o in Tcl stesso e facilmente incorporati grazie alla semplice interfaccia fornita dal parser Tcl. L'aggiunta di comandi specifici di un'applicazione arricchisce il linguaggio Tcl senza variarne minimamente la sintassi; non c'è alcuna differenza tra i comandi predefiniti di Tcl e quelli aggiunti dall'applicazione.

È possibile creare estensioni (ovvero set di nuovi comandi) che lavorano con diverse versioni di Tcl senza necessità di ricompilazione. Alcune estensioni Tcl general-purpose e liberamente disponibili:

**incr Tcl** Estensione che aggiunge complete funzionalità object-oriented a Tcl, ispirata al linguaggio C++.

**Expect** Estensione che consente di automatizzare il dialogo con i programmi interattivi in Unix.

**TclX** Detto anche Extended Tcl, set di estensioni orientato principalmente alla programmazione di sistema di Unix.

**Tcllib** Libreria di codice di utilità scritto in Tcl puro. Attualmente consta di circa 80 moduli. Ogni modulo contiene un certo numero di comandi correlati.

**Snack** Toolkit per la manipolazione e l'analisi dei suoni, per creare applicazioni audio multi-piattaforma.

**BLT** Toolkit che aggiunge nuovi tipi di widget più complessi, tra cui grafici 2D, controlli ad albero e contenitori a schede.

Attualmente tpad utilizza solo TclX e Tcllib, per alcune funzioni avanzate, tra cui il locking dei file e la creazione dei file temporanei in modo portabile. Tpad può essere utilizzato anche se queste estensioni non sono installate.

Naturalmente in questo caso le funzionalità che richiedono queste librerie aggiuntive non saranno disponibili.

# 4 I plugin di tpad

Tpad può essere facilmente esteso tramite l'aggiunta di plugin specifici, che permettono di invocare utility esterne per l'elaborazione del testo senza lasciare l'editor.

La scrittura di un nuovo plugin non richiede una conoscenza dettagliata del resto del codice di tpad, ma solo l'adesione ad una serie di semplici convenzioni necessarie affinché il plugin possa essere invocato dall'interno di tpad ed operare correttamente la lettura ed eventuale modifica del testo dell'editor.

Ogni plugin è "specifico" nel senso che "conosce" l'interfaccia a riga di comando, il modo di operare e il formato dei messaggi di errore emessi dall'applicazione esterna che esegue.

I plugin forniscono all'utente una interfaccia grafica facile da usare con le seguenti funzionalità (ove applicabili):

- impostazione delle opzioni del programma da richiamare in modo consistente, prevenendo ogni tipo di errore di invocazione
- raccolta di eventuali messaggi di errore forniti dal programma esterno, per consentirne la comoda consultazione da parte dell'utente
- opzione per stabilire se il filtro esterno deve essere applicato all'intero testo o solo ad una sua porzione selezionata
- posizionamento veloce del cursore di inserzione dell'editor alla riga e alla colonna del testo dove si è verificato un errore

I plugin vengono caricati dinamicamente la prima volta che vengono visualizzati, tramite la funzionalità di auto-loading del codice di Tcl. In questo modo può essere sviluppato anche un numero elevato di plugin, senza che i tempi di startup dell'applicazione si allunghino.

Ho a lungo pensato anche ad un tipo di plugin generico, che possa essere configurato tramite GUI, per generare nuovi plugin in grado di richiamare qualsiasi tipo di filtro o applicazione esterna. Tuttavia mi sono reso conto che per generare plugin specifici con tutte le caratteristiche volute, il plugin generico dovrebbe essere molto complicato sia da implementare sia da configurare e di certo non potrà mai avere la flessibilità consentita dalla programmazione diretta in Tcl di un nuovo plugin. Perciò qualora decidessi di includere in futuro un plugin generico che permetta anche agli utenti non programmatori di creare dei semplici plugin, questo non sarà sostitutivo dei plugin specifici già implementati.

## 4.1 Plugin Tabella ASCII

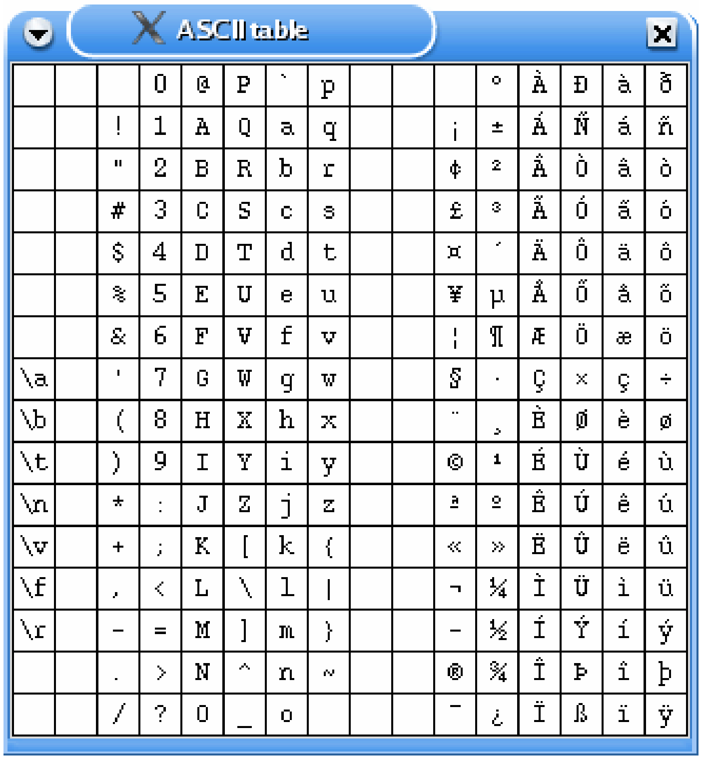


Figura 3: Il plugin Tabella ASCII

Questo plugin (Figura 3) visualizza una griglia quadrata contenente 256 caselle (16x16): nelle caselle sono rappresentati i caratteri del set ASCII standard (codici da 0 a 127) e quelli dell'estensione nota come Latin-1 (codici da 160 a 255). Di solito i font non comprendono i glifi corrispondenti ai caratteri di controllo (i primi 32 del set ASCII standard) e quelli non appartenenti all'estensione Latin-1. In questo caso le caselle corrispondenti sono vuote, eccetto per alcuni caratteri di controllo che vengono rappresentati con le ben note sequenze di escape usate in C o Tcl. Cliccando con un tasto del mouse su un carattere, questo viene inserito nel testo, alla posizione corrente del cursore.

La tabella ASCII risulta utile per inserire nel testo caratteri latini di cui manca il tasto apposito sulla tastiera.

## 4.2 Plugin Correttore e validatore di HTML

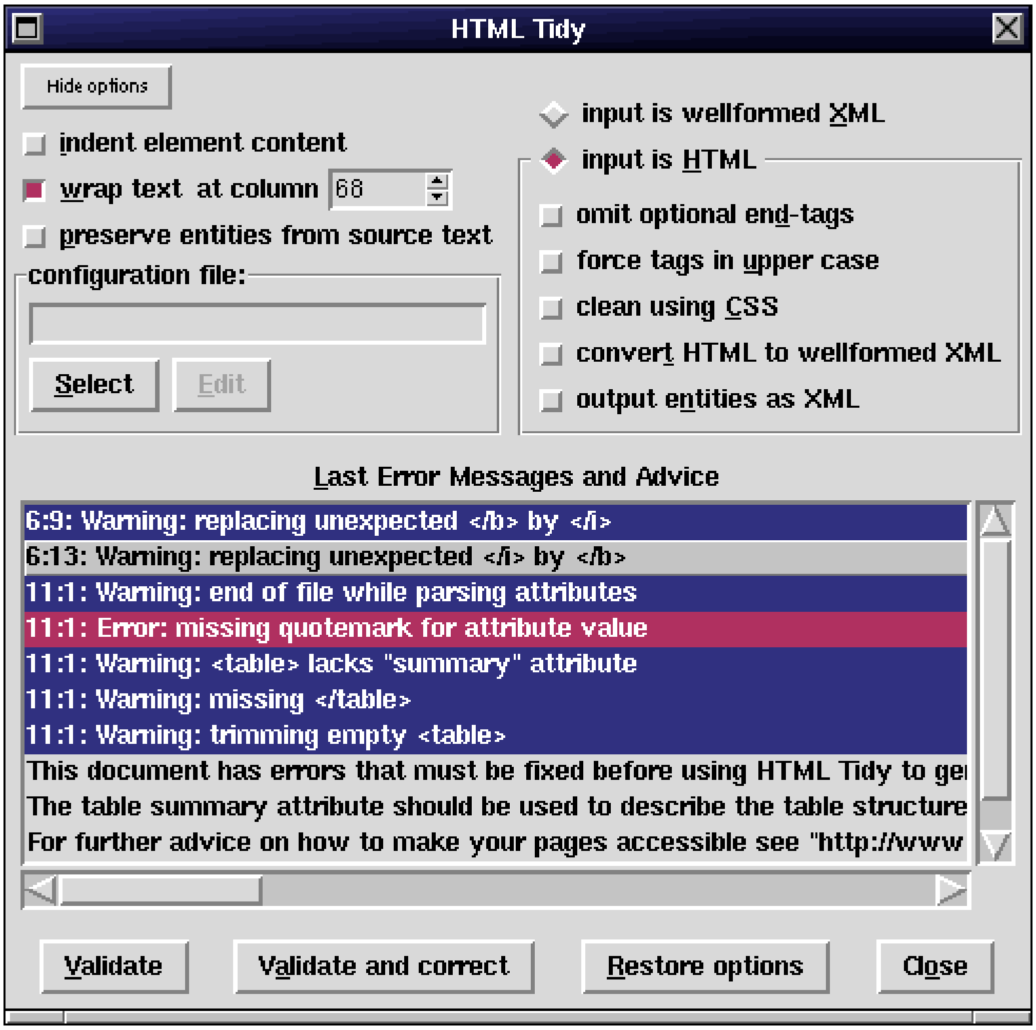


Figura 4: Il plugin di HTML Tidy

HTML Tidy [11] è un validatore, correttore e formattatore di codice HTML, scritto da Dave Ragget del W3C.

Tidy è in grado di aggiustare automaticamente molti errori presenti nel markup e, nei casi di ambiguità, emette messaggi di errore esplicativi che aiutano il webmaster a correggere manualmente il problema. Tidy migliora molto il pessimo markup generato da alcuni editor HTML e strumenti di conversione. Ha perfino una modalità apposita per ristrutturare l'orribile markup generato da Microsoft Word 2000! Inoltre fornisce una serie di consigli per rendere le pagine più accessibili a persone disabili.

Tidy è un programma molto ricco e può essere configurato con circa una cinquantina di opzioni. Allo scopo di rendere l'interfaccia grafica semplice ho selezionato le sole opzioni più comunemente usate (Figura 4). Eventuali direttive più sofisticate possono comunque essere specificate, utilizzando un file di configurazione testuale esterno, che può essere modificato lanciando un'altra istanza di tpad.

## 4.3 Plugin Controllore di XML Ben-Formato

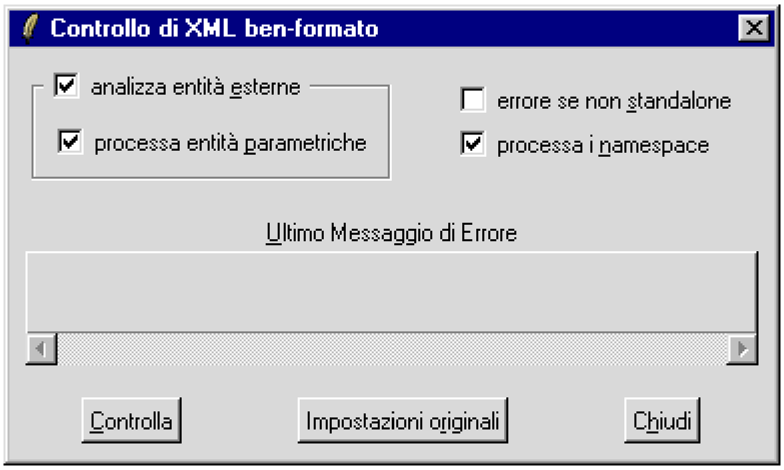


Figura 5: Il plugin xmlwf

Xmlwf è un plugin che determina se un documento XML è ben formato (Figura 5). Xmlwf è basato sulla libreria Expat, che implementa in linguaggio C un parser per XML versione 1.0, pienamente conforme alle specifiche W3C.

Xmlwf è un processore non validante: se il documento ha una DTD, non controlla se la sintassi della DTD è corretta e se il documento è conforme alla DTD stessa. Tuttavia supporta il parsing delle entità esterne.

## 4.4 Plugin Formattatore di Testo

Fmt è il classico filtro formattatore di testo di Unix. Ne esistono fondamentalmente due varianti, che hanno interfaccia e opzioni diverse: quella BSD (Figura 6), disponibile per default in FreeBSD e quella GNU (Figura 7), tipica di Linux: tpad le supporta entrambe.

La funzionalità fondamentale, comune sia alla versione BSD che GNU, consiste nel riorganizzare il testo, riempiendo quanto più possibile le linee di uno stesso paragrafo, senza superare un certo numero prefissato di caratteri

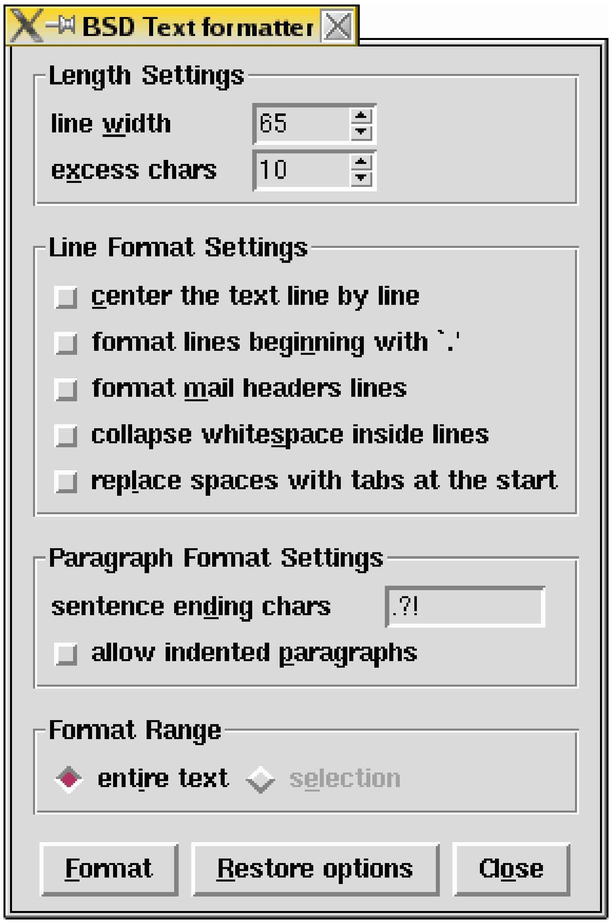


Figura 6: Il plugin BSD fmt

per linea. Per fare ciò le linee vengono unite o spezzate solo in corrispondenza degli spazi bianchi; vale a dire che le parole non vengono mai unite o sillabate.

# 5 Struttura del codice

Il codice di tpad segue le convenzioni dettate dalla Guida di Stile Tcl [12], che è stata elaborata dal gruppo di lavoro su Tcl/Tk dei Laboratori Sun. Le convenzioni di stile usate non sono importanti di per sè quanto lo è il fatto di essere consistenti con esse. L'uso di uno stile uniforme infatti facilita il riuso, la lettura e la manutenzione del codice da parte di un altro programmatore.

Nonostante il nome, la Guida di Stile Tcl non si limita a prescrivere delle regole di stile di codifica. Vi si trovano anche regole pratiche e suggerimenti

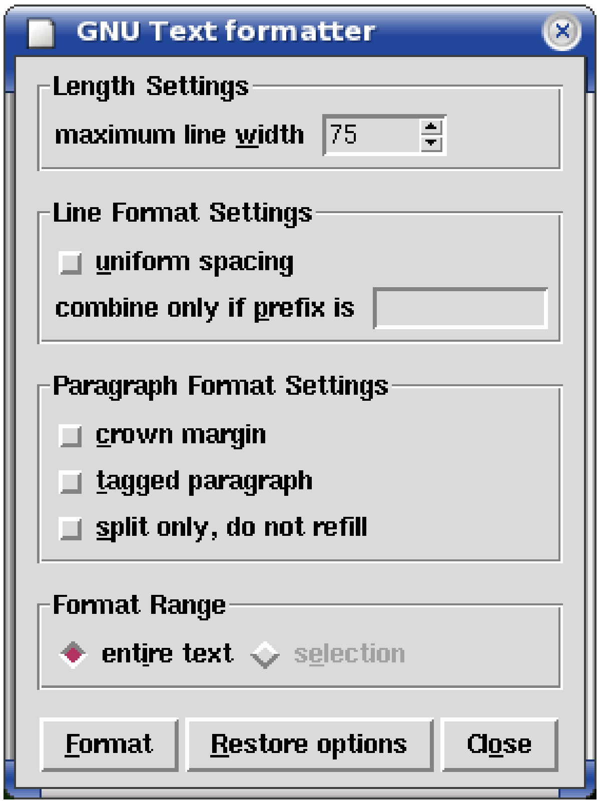


Figura 7: Il plugin GNU fmt

che riguardano alcuni aspetti importanti per la produzione di software di qualità:

- documentazione e commenti
- scelta appropriata dei nomi degli identificatori
- portabilità
- automatizzazione e riproducibilità dei test
- strutturazione del codice in entità facilmente riusabili e ridistribuibili
- prevenzione degli errori di programmazione

La Figura 8 rappresenta il nesting degli spazi dei nomi quando sono attive $n$ istanze di tpad.

Ciascuna istanza è identificata tramite il percorso Tk della finestra grafica che la contiene. I percorsi in Tk hanno sintassi analoga a quelli dei file o delle

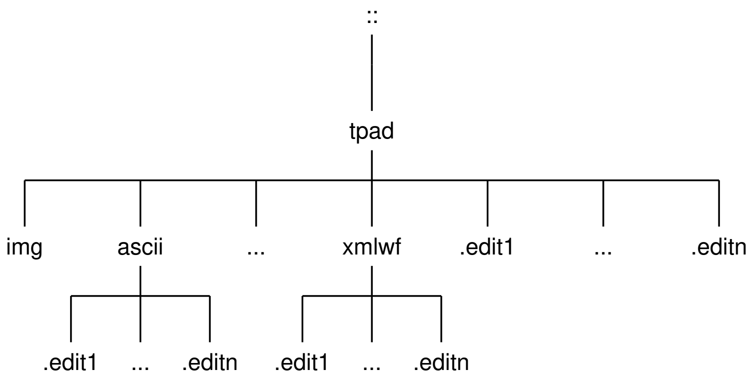


Figura 8: La struttura ad albero dei namespace creati da tpad

directory in un filesystem Unix. Le uniche differenze sono che al posto del carattere slash (/) si usa il punto (.) e non si può avere un punto alla fine di un percorso, eccezion fatta per il percorso radice (.), che rappresenta la finestra principale dell'applicazione. Le $n$ istanze sono state chiamate con i nomi *.edit1 ... .editn* solo per semplicità; in realtà ciascuna istanza può essere creata associandogli un percorso identificatore qualsiasi, purché esista il widget padre.

Tutti i comandi e le variabili di tpad risiedono all'interno del namespace *tpad*; nessuna procedura o variabile di tpad risiede nello spazio nomi globale (*::*). Questa è l'organizzazione raccomandata dalla Tcl Style Guide [12] per evitare il problema della collisione dei nomi, che è il motivo principale per cui sono stati introdotti i namespace: un comando non dovrebbe mai creare o modificare altri comandi o variabili nello spazio globale a meno che non gli venga chiesto esplicitamente tramite la sua interfaccia dei parametri.

I comandi esportati dal namespace *::tpad* costituiscono l'interfaccia pubblica della libreria, e si possono suddividere in tre categorie:

- comandi di inizializzazione, per creare l'istanza e per visualizzarla
- comandi corrispondenti a tutte le voci del menu dell'interfaccia di default (Figura 2, pag. 4)
- comandi addizionali, non corrispondenti alle voci del menu

I comandi addizionali sono comandi di utilità che, al pari dei comandi di menu, possono essere invocati dall'esterno in qualsiasi ordine e con qualsiasi argomento, senza causare errori imprevisti oppure perdita di consistenza dello stato di una istanza.

Ad esempio un comando addizionale è *goToLine* che permette di riposizionare il cursore di inserzione ad una certa riga o colonna passata come parametro e riaggiustare la vista nella finestra dell'editor in modo che il cursore sia visibile nella sua nuova posizione. Questo non è il comando che corrisponde alla voce del menu *Edit/Goto line*, che si chiama invece *goTo*. Quest'ultimo visualizza la finestra di dialogo modale *Goto line* che internamente usa il precedente comando *goToLine*, associandolo alla pressione del tasto *Go*, il tasto che appunto causa il riposizionamento del cursore.

Altri comandi addizionali ritornano informazioni di stato dell'editor, come per esempio il comando *lines*, che ritorna il numero di linee del testo corrente.

Per contro i comandi il cui nome inizia con una lettera maiuscola non fanno parte dell'interfaccia della libreria e sono da considerarsi privati; essi non sono ufficialmente supportati e potrebbero cambiare oppure scomparire senza notifica. Per questo motivo il loro riuso dovrebbe essere evitato.

Quasi tutti i comandi operano su una specifica istanza dell'editor e richiedono come primo parametro il pathname Tk che la identifica.

Il namespace *::tpad::img* (Figura 8) non contiene comandi, ma solo nomi di immagini, che funzionano da identificatori nei buffer tenuti da Tk. Le uniche immagini usate fino ad adesso da tpad sono quelle della toolbar e le icone predefinite di Tk (usate nelle finestre di dialogo). Gli ID delle immagini sono tenuti in un namespace separato per amor di ordine all'interno della libreria e per evitare collisioni con altri nomi di immagini eventualmente caricate da un'applicazione che usa la libreria tpad.

Inoltre per ogni istanza dell'editor esiste un namespace innestato all'interno di *::tpad* avente lo stesso nome dell'identificatore dell'istanza. Nell'esempio di Figura 8, questi namespace sono *::tpad::.edit1* ... *::tpad::.editn*. Ciascuno di questi namespace contiene tutte le variabili di stato dell'istanza come ad esempio il nome del file corrente che si sta editando, il tipo di istanza (se editor o viewer), l'ultimo pattern utilizzato per la ricerca (allo scopo di poterla ripetere), ecc.

Infine ogni plugin vive nel proprio namespace, che contiene sia i comandi del plugin (esportati e privati) sia le eventuali variabili di stato relative al plugin, suddivise all'interno di namespace con nome coincidente con il pathname dell'istanza, secondo la stessa convenzione usata nel namespace principale *::tpad* (Figura 8). Questo assicura che non vi siano mai collisioni dei nomi tra i plugin. Ogni plugin può accedere anche alle variabili dell'istanza dell'editor o ai comandi pubblici o privati della API di tpad, usando i loro nomi pienamente qualificati.

Tpad è un programma completamente configurabile: al momento le direttive di configurazioni manipolabili sono all'incirca 80. La configurazione è memorizzata in un unico e semplice file Tcl, che viene caricato nel momento in cui si crea la prima istanza. In questo modo non è stato necessario definire un nuovo formato e creare un parser per il file di configurazione. Quando il programma viene installato viene creato un file di configurazione globale, valido per default per tutti gli utenti di un sistema. Gli amministratori di sistema possono cambiare il file di configurazione globale se desiderano imporre diversi valori di default per alcune opzioni. Secondo un meccanismo comune in Unix, ma applicabile anche in altri sistemi multi-utente, gli utenti possono copiare il config file globale in un file nascosto all'interno della loro home directory qualora desiderino partire con valori di default differenti da quelli stabiliti dall'amministratore.

Le direttive di configurazione per tpad si dividono in due categorie:

- variabili o costanti di configurazione che si applicano a tutte le istanze di tpad, create nel namespace *::tpad*
- valori iniziali delle variabili relative ad ogni singola istanza di tpad; queste pure sono definite nel namespace *::tpad* ma vengono poi copiate nei namespace che conservano lo stato di ogni singola istanza quando l'istanza viene creata, ad es. nello spazio nomi *::tpad::.edit1* per l'istanza .edit1.

Nello stesso modo sono memorizzate e copiate le direttive di configurazione relative a ciascun plugin, a partire dal namespace che rappresenta il plugin, ad es. *::tpad::GNUfmt*.

Dalle considerazioni fatte, si capisce che la API offerta da tpad, basata principalmente sui namespace dinamici di Tcl, ha una struttura orientata agli oggetti, pur non essendo basata su nessuna estensione object-oriented di Tcl. Le istanze di tpad sono simili agli oggetti e il pathname Tk che li identifica è come il nome unico di una variabile oggetto. I namespace dinamici di Tcl sono usati per contenere diversi oggetti e gli operatori (comandi) che agiscono su di essi, ed essendo degli insiemi di oggetti simili con operazioni, corrispondono al concetto di classe. Il comando di inizializzazione che crea l'istanza è l'analogo di un costruttore. Gli altri comandi pubblicamente esportati dal namespace *::tpad* si possono definire metodi pubblici e quelli non esportati, metodi privati. I comandi esportati che non hanno un parametro che identifica l'istanza di tpad su cui agire sono come i metodi statici di una classe. Il pathname Tk che identifica l'istanza di tpad su cui operare equivale alla variabile *this* che si riferisce all'oggetto corrente. Le variabili o costanti di configurazione relative a tutte le istanze di tpad o di un suo plugin sono l'analogo delle variabili o costanti statiche di una classe.

# 6 Controllo di tpad dall'interno di altre applicazioni

Un notevole beneficio si ha quando un'applicazione, sia scritta in Tcl puro sia in qualche altro linguaggio di sistema che può incorporare Tcl, implementa, oltre che la sua GUI, anche le sue principali funzionalità tramite estensione del linguaggio Tcl, ovvero tramite una API Tcl. In tal caso il linguaggio Tcl stesso potrà essere usato anche per il controllo dell'applicazione da parte di un'altro processo che gira sullo stesso display.

Questo avviene tramite il comando *send*, che riceve come argomenti un nome che identifica univocamente un'applicazione e un comando Tcl qualsiasi, con i suoi argomenti, da eseguire nell'applicazione identificata. Si tratta di una forma di comunicazione simile alle chiamate di procedura remote, ma limitata ad applicazioni che hanno la finestra grafica principale sullo stesso display.

L'invocazione può essere fatta in modo sincrono o asincrono. Nel modo sincrono, che è quello di default, l'applicazione che manda il comando ad un'altra si mette in attesa che il comando inviato sia stato eseguito. In tal caso *send* ritornerà il risultato del comando (od eventuali informazioni di errore) all'applicazione invocante. Nel modo asincrono invece il comando *send* ritorna immediatamente senza alcuna attesa da parte del chiamante e le informazioni di ritorno dal comando chiamato vengono scartate.

Gli usi del comando send sono innumerevoli. Ad esempio i demo Tk includono un'applicazione detta tcolor che è un semplice color editor. Nella Figura 9 vediano un esempio di come *send* possa essere usato per la riconfigurazione a run-time delle GUI definite tramite Tcl: l'utente avvia tcolor, sceglie un nuovo colore e poi manda ad un'altra applicazione, nell'esempio a tpad, un semplice comando Tcl per impostare lo sfondo del testo con il colore scelto.

La Figura 10 mostra un altro esempio di comunicazione tra applicazioni. Per la simulazione ho utilizzato *rmt*, una semplice applicazione demo di Tk che consente, tramite una comoda GUI, di scegliere un'applicazione destinataria e inviarli dei comandi. I comandi visibili nella Figura 10, mostrano come un ipotetico debugger possa comunicare con tview, la versione di tpad che opera come solo visualizzatore di testo. Il debugger dapprima comanda a tview di aprire il file sorgente e poi gli chiede di evidenziare la linea corrente di codice che dovrà essere eseguita durante un'esecuzione passo passo.

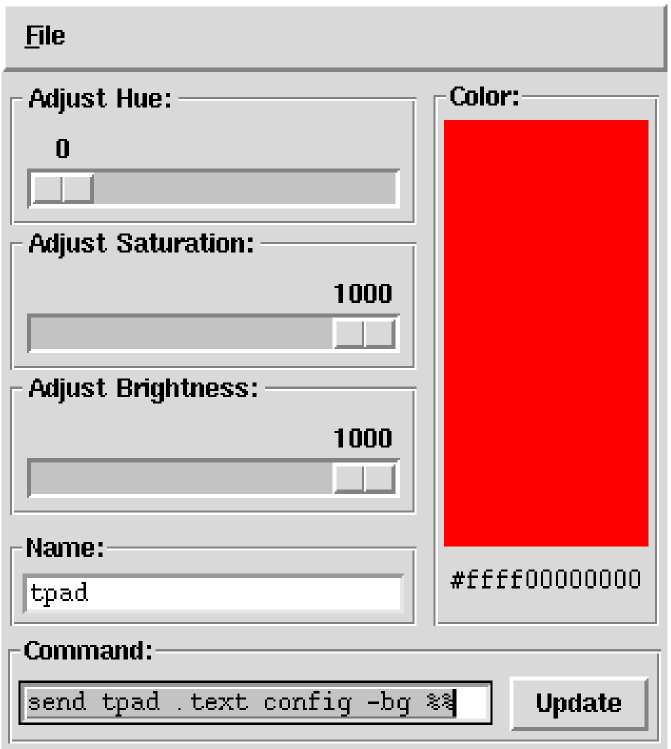


Figura 9: L'applicazione tcolor può essere usata per cambiare i colori di un'altra

# 7 Utilizzo di tpad in un'architettura client-server grafica

Il sistema grafico network-trasparente X Window, chiamato anche X11 o semplicemente X, disponibile per un'ampia gamma di sistemi operativi e architetture hardware, rende possibile l'utilizzo delle sue applicazioni grafiche attraverso la rete. Questo vale anche per le applicazioni Tcl/Tk in Unix.

Con il termine *stazione grafica* o *display X* si intende l'insieme di una tastiera, uno o più monitor e un mouse utilizzati insieme per l'interazione. In genere una workstation ha un solo display (con un solo monitor) e quindi può essere usata da un solo utente alla volta, ma X supporta anche situazioni più complesse, con più di un display e con più di un monitor appartenente a ciascun singolo display grafico.

Il sistema X è suddiviso in vari pacchetti contenenti non solo programmi eseguibili, ma anche font, icone, file di configurazione, librerie condivise con relativi file header e documentazione per utenti e programmatori.

Un servente X (X server) si occupa della gestione di un unico display e

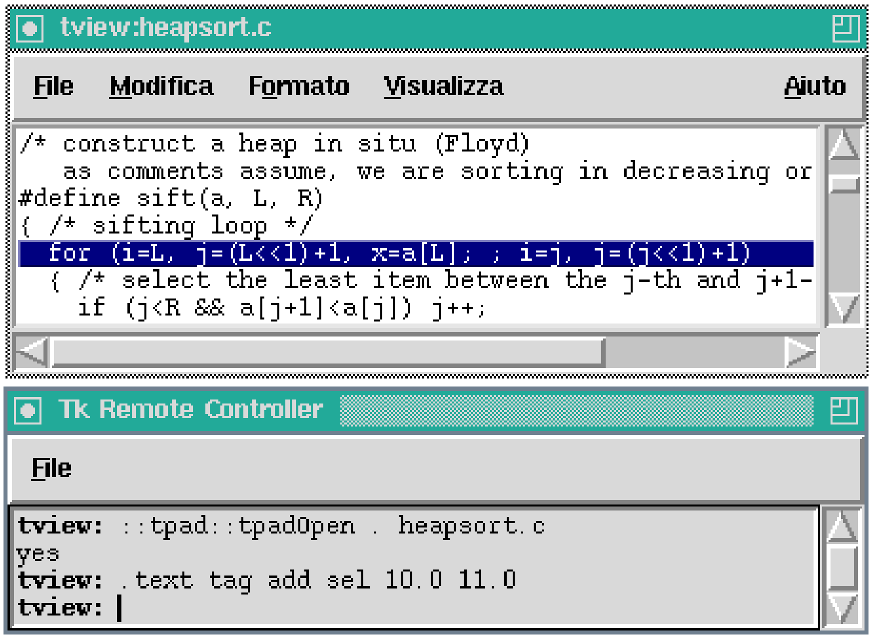


Figura 10: Simulazione dell'uso di tpad da parte di un debugger

contiene i driver specifici per far funzionare l'hardware utilizzato. Nel caso un calcolatore abbia più di un display, si devono avviare più server X, uno per ciascun display che si vuole controllare.

Ciascun server X mette a disposizione le primitive per il disegno e gestione degli eventi tramite un opportuno protocollo, permettendo solo in questo modo l'accesso al display ai programmi applicativi.

Quest'ultimi sono detti clienti X (X client) e possono girare sullo stesso calcolatore a cui il display controllato dal server X è fisicamente connesso o su calcolatori differenti connessi tramite reti TCPIP o DECnet. I client X si connettono al server X locale o remoto e per gestire l'interazione con l'utente utilizzano le funzioni delle librerie client X (Xlib) che implementano il protocollo X nascondendone la complessità.

Persino il gestore di finestre (window manager) non è altro che un client X speciale e come tutti i client X è un programma separato dal server X. Infatti ne esistono svariati tipi intercambiabili. Tutti implementano comunque le funzionalità di base che consistono nel permettere all'utente di passare da un'applicazione X ad un'altra e di aprire, chiudere, ridimensionare, spostare e ridurre ad icona le finestre delle applicazioni.

Ho sperimentato l'utilizzo di tpad attraverso la rete, facendo uso di un client X Linux e una macchina Windows XP nel ruolo di server X.

In Linux al minimo occorre installare: tpad, l'interprete Tcl/Tk e le librerie di XFree86 [13], una versione open-source di X. Non è necessario installare un server X, a meno che non si voglia utilizzare una eventuale console grafica direttamente sul server.

In Windows occorre installare un server X e le relative librerie client Xlib, che sono necessarie anche per i server X. Ho scelto XWin, la port di XFree86 per questo sistema che si basa sul layer di compatibilità Unix Cygwin.

La macchina Windows va poi configurata per permettere l'accesso controllato del computer con Linux al suo display con uno dei metodi di autenticazione messi a disposizione da X Window, che variano a seconda del livello di sicurezza richiesto. I dettagli si possono trovare nella documentazione ufficiale in lingua Inglese di tpad.

Dato che il computer Linux agiva anche da server web, questa configurazione è risultata molto comoda per editare i file HTML a livello di sorgente direttamente sul server web, senza necessità di continui trasferimenti tramite FTP. Tpad risulta adatto per questo scopo in quanto è molto simile a Notepad, quindi è già familiare a tutti gli utenti Windows, e inoltre è dotato di correttore HTML integrato (sezione 4.2).

# 8 Sviluppi futuri

*Un programma non è mai finito, ma*
*da qualche parte bisogna pur fermarsi.*

— Peter Norton

Dal punto di vista dell'utente le prossime versioni vedranno implementate molte nuove funzionalità di editing e nuovi plugin. Il file "TODO" reperibile nel sito ufficiale [4] presenta una lista dettagliata degli sviluppi futuri.

Dal punto di vista del programmatore, la API di tpad sarà riscritta utilizzando l'estensione object-oriented Itcl. La nuova API rimpiazzerà l'attuale sua mimica basata solo namespace dinamici e Itcl diventerà una dipendenza obbligatoria di tpad e di tutte le applicazioni che lo incorporeranno. La attuale API è un po' inconsistente dal punto di vista sintattico con il resto del linguaggio soprattutto perchè non ha una sintassi di comandi con parametri con nome, bensì anonimi. Ho in progetto di uniformare completamente la sintassi con quella degli altri comandi standard di Tcl.

Attualmente la libreria tpad permette la creazione di editor capaci di editare solo testo semplice. In futuro verrà estesa in modo che possa essere usata da supporto per la creazione di sistemi di word-processing WYSIYWG.

Al contrario di quello che si potrebbe pensare a prima vista, questa evoluzione non richiederà molti sforzi e sarà in gran parte indolore: non sarà infatti necessario complicare la API di tpad, che rimarrà semplice ed elegante com'è attualmente, grazie al fatto che il widget `text` di Tk su cui tpad è basato e la libreria sono del tutto disaccoppiati. Inoltre il widget `text` già permette di visualizzare diverse porzioni di testo con differenti font e colori, di incorporare delle immagini nel testo e persino creare sistemi di navigazione ipertestuali.

La libreria pertanto dovrà subire solo poche modifiche interne per supportare queste nuove funzionalità. Per esempio le funzioni di lettura e salvataggio dei testi dovranno essere modificare per consentire la lettura e la scrittura di file con formato di testo formattato, ma la loro interfaccia rimarrà retrocompatibile.

Tutto questo vedrà la luce nella versione 2.0 di tpad.

## Riferimenti bibliografici


[1] Free Software Foundation. *GNU General Public License, Version 2, 1991.* http://www.fsf.org/copyleft/gpl.html

[2] *The FreeBSD Project.* http://www.FreeBSD.org

[3] A. Tridgell, P. Mackerras. *The rsync algorithm*, Australian National University of Canberra, TR-CS-96-05, 1996.

[4] *Tpad web site.* http://tclpad.sourceforge.net

[5] J. K. Ousterhout. *Tcl: An Embeddable Command Language*, Proc. USENIX Winter Conference, pp. 133-146, 1990.

[6] J. K. Ousterhout. *Scripting: Higher Level Programming for the 21st Century*, IEEE Computer magazine, March 1998.

[7] The Unicode Consortium. *The Unicode Standard Worldwide Character Encoding, Version 2.0*, Addison-Wesley, 1996.

[8] J. K. Ousterhout. *An X11 Toolkit Based on the Tcl Language*, Proc. USENIX Winter Conference, pp. 105-115, 1991.

[9] S. J. Peralta. *Scripting Graphical Commands with Tcl/Tk Mini-HOWTO.* http://www.tldp.org/HOWTO/Scripting-GUI-TclTk

[10] *TclPro Compiler.* http://www.tcl.tk/software/tclpro/compiler.html

[11] D. Ragget. *Clean up your Web pages with HTML TIDY.* http://www.w3.org/People/Raggett/tidy/

[12] R. Johnson. *Tcl Style Guide*, Sun Microsystems, 1997.

[13] *XFree86 web site.* http://www.xfree86.org