Personaggi ed aneddoti protagonisti della storia di GNU-Linux

La storia di Linux è storia di una tecnologia, ma anche di uomini, idee e macchine

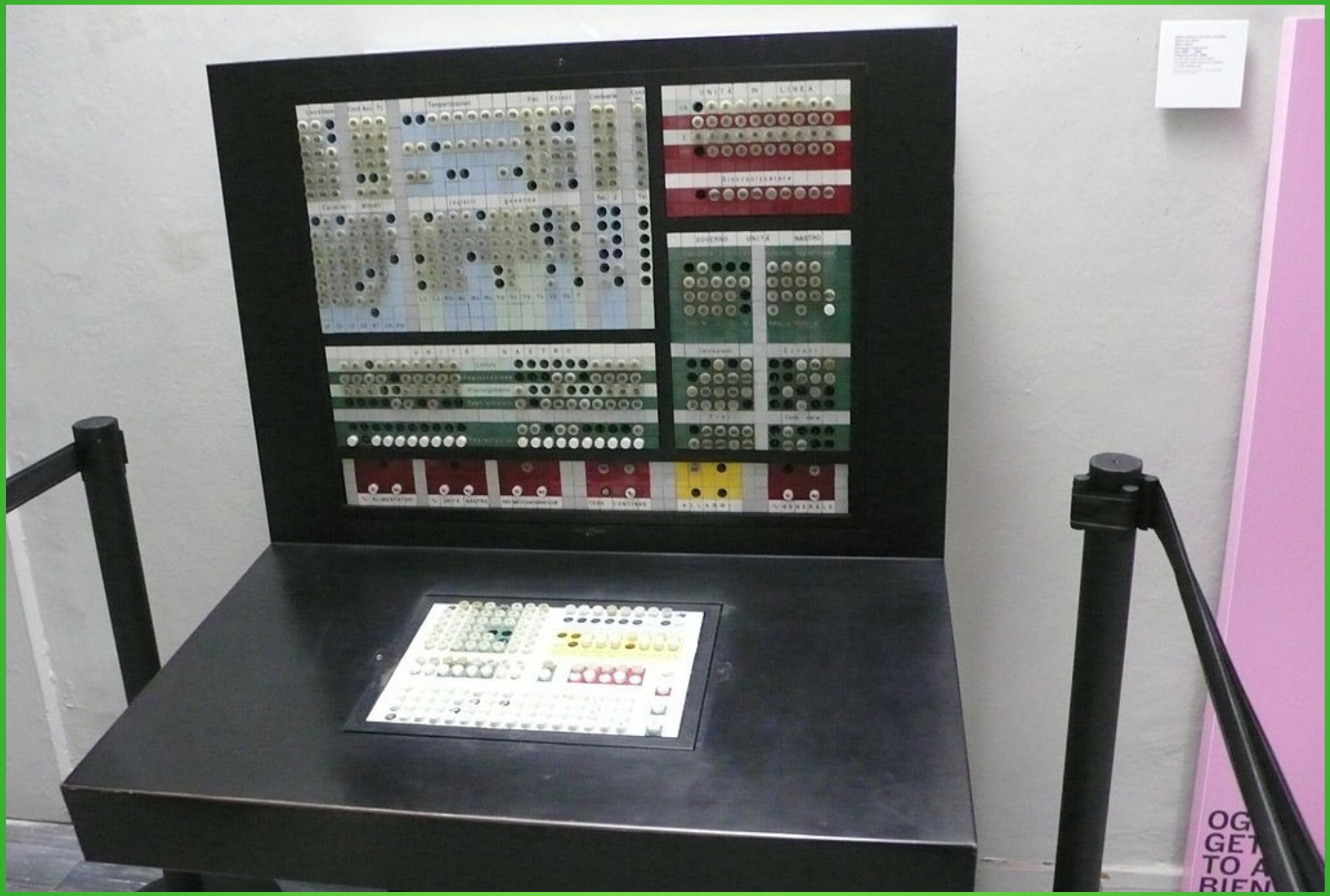
.

E' difficile decidere da dove inizi la storia di GNU-Linux. Si potrebbbe partire dal 1991, anno del post di Linus Torvalds in cui annuncia il progetto per il kernel), oppure dal 1983, anno di partenza del progetto GNU di Stallman, ma Stallman che tanto ha fatto per GNU-Linux operava al Laboratorio per l'Intelligenza Artificiale del MIT, lì respirò l'aria che circola in GNU-Linux.

- Infatti GNU-Linux è una tecnologia, ma anche un modo di produrre software e di condividerlo, un processo di sviluppo, ma anch eun modo d'intendere non solo la tecnologia, ma la società intera.
- Possiamo dire che esiste una filosofia GNU-Linux che unifica storie e personaggi diversi, a volte diversissimi
- Proverò brevemente a descrivere quali siano le caratteristiche che animano il Pinguino e che nascono insieme al computer, quando ancora il computer era un mainframe enorme e costosissimo
- Cominciamo raccontando storie e vedremo che da ogni storia nasce un carattere di GNU-Linux

ELEA 9000 (1957)

Consolle Elea 9000



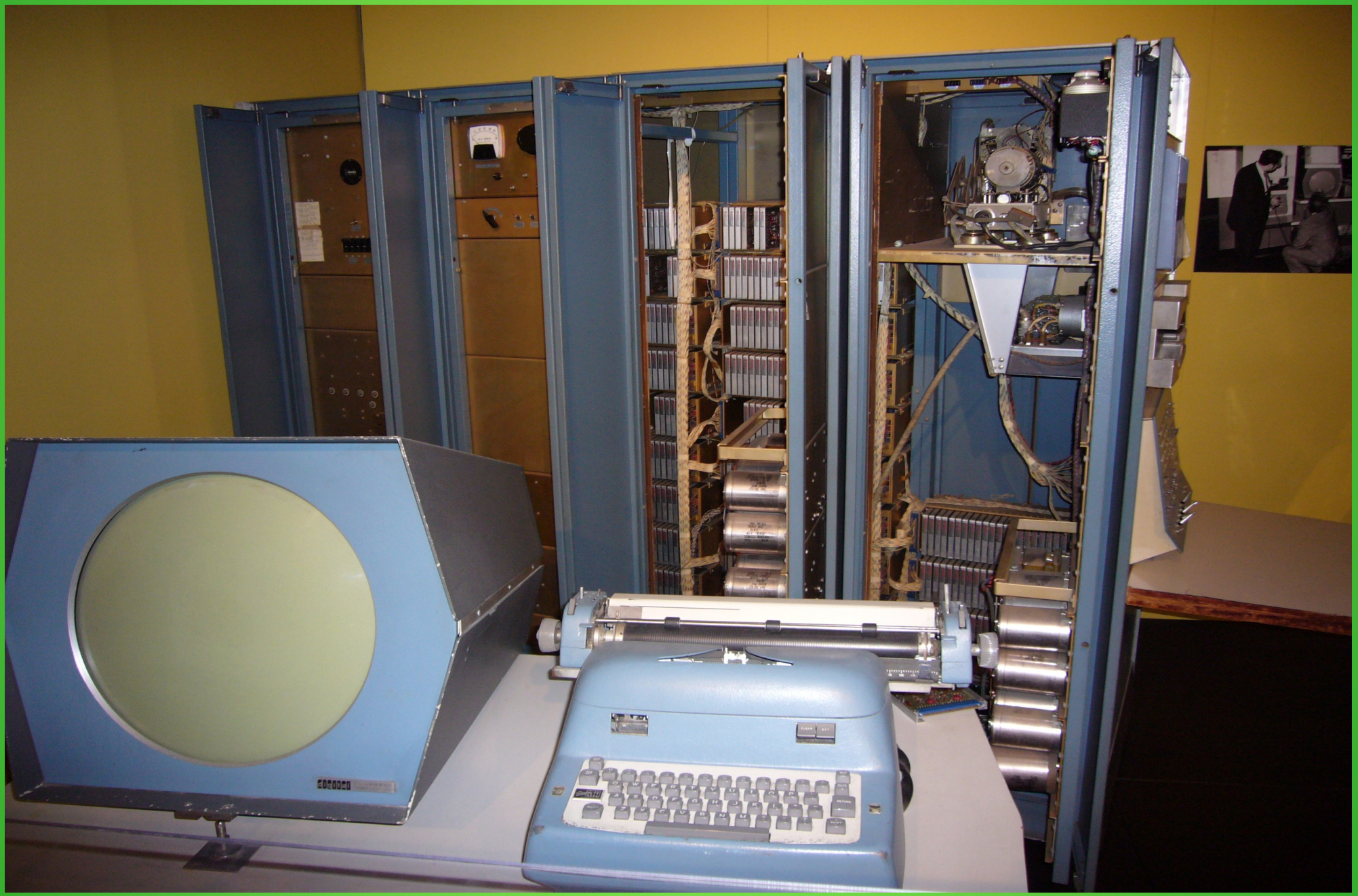
Elea 9003 al Monte dei Paschi

Potevo mostrare un qualsiasi “mostro” degli albori dell'informatica. Ma ho scelto Elea 9000 perché italiano, perché tecnologicamente avanzato per l'epoca, è stato il primo computere a transistor, ma soprattutto perché bello. I computer moderni nascono in ambito militare durante la seconda guerra mondiale. Sono manipolatori di simboli, sostituiscono le letter dei messaggi cifrati tedeschi con altre lettere, seguendo istruzioni complesse, poi vengono usati per calcolare le tabelle balistiche dei proiettili, effettuando calcoli complicati. Insomma calcolatori e elaboartori di simboli. I due utilizzi dei primi mainframe, il calcolo matematico e i database. I primi linguaggi, tra gli anni '50 ed i primi '60, il FORTRAN ed il COBOL
Costosi sul milione di dollari, delicati e complessi
ELEA 9000, il prototipo, ELEA 9003 il primo commercializzato è curatissimo nel deisgn, nell'elettronica, nel cablaggio, nell'attenzione al particolare, nella ricerca della perfezione
Le prime parti dello spirito GNU-Linux: **la ricerca del bello, del funzionale, dell'accurato, l'amore per la tecnologia come artigianato, come arte, come espressione di se, come potenziamento ed espressione di personalità.**

Vi invito a perdere una mezz'oretta per vedere questi video su yuotube e capirete

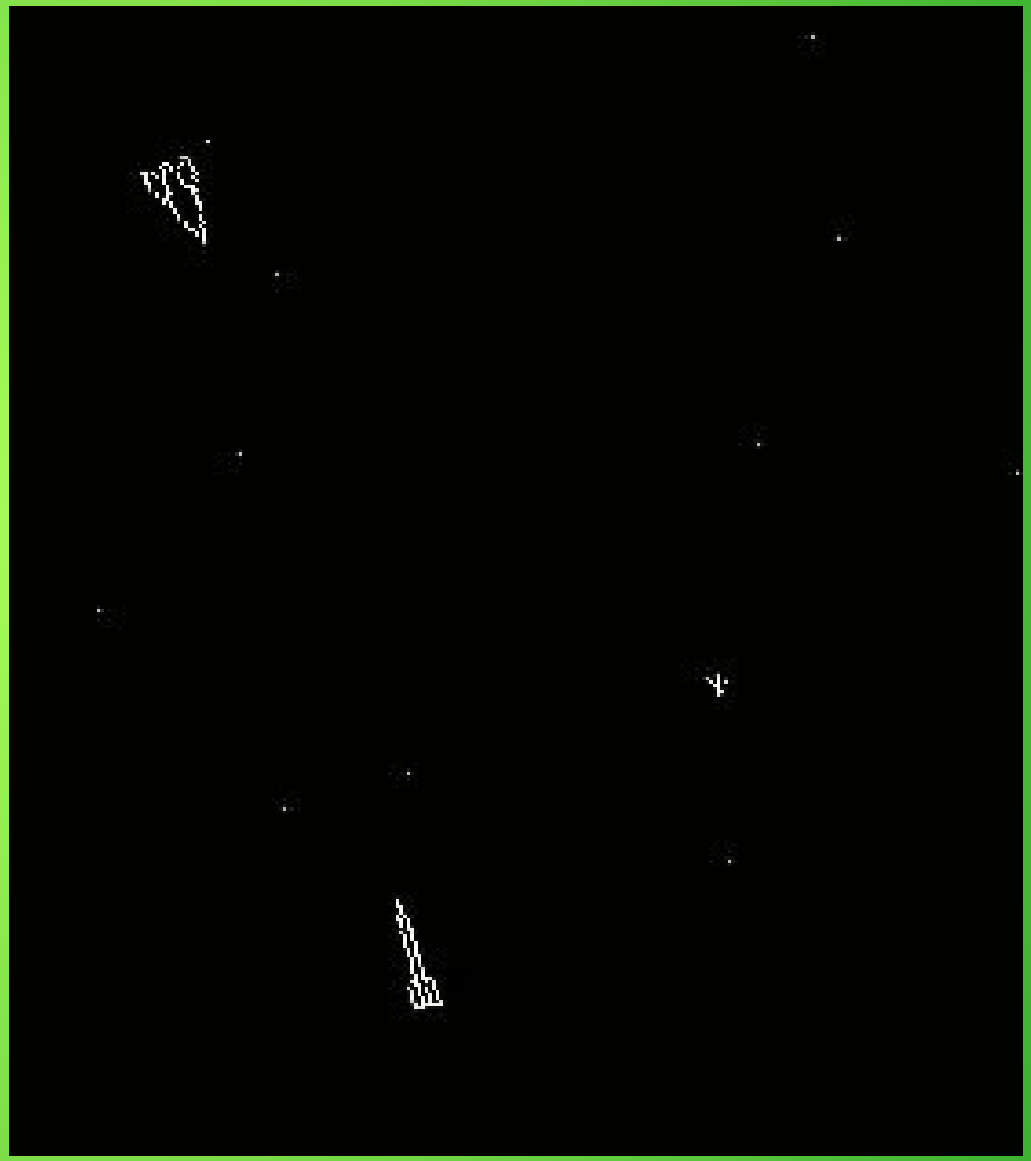
http://www.youtube.com/watch?v=zcRjDlEYGeM
http://www.youtube.com/watch?v=lUhSYR68PqQ

PDP-1 (1960)



Steve Slug Russel

Spacewar



Il primo computer “piccolo” e poco costoso, 120000 dollari, arriva al Laboratorio IA del MIT e subito si nota un tubo catodico. Steve Slug Russel mentre scriveva una applicazione matematica, scoprì casualmente una funzione che disegnava, sullo schermo del PDP-1, elaborate forme ellittiche. Grazie alle funzioni di seno e coseno della DEC, Russel scrisse una versione "base" del gioco in circa tre mesi: questa è stata, in seguito, soggetta a diverse migliorie e aggiunte da parte di amici e colleghi. Una delle prime implementazioni venne inserita da Shawn Garetz. Questo, come altri membri del MIT, era un accanito lettore di romanzi di fantascienza dello scrittore americano "Doc" Smith, dove si poteva leggere di come i "pirati spaziali" potessero usufruire di misteriose "gallerie iperspaziali" per raggiungere immediatamente diversi punti nella galassia. Decise così di fare in modo che, con la pressione di un tasto, la navetta andasse in iperspazio, cioè che scomparisse dal punto dov'era in quell'istante per riapparire in un punto casuale della schermata.
Una seconda, importante modifica è stata quella di Dan Ewards, che inserì un corpo celeste al centro dello schermo. Questo, dotato di una sua gravità, attira verso di sé l'astronave distruggendola, costringendo il giocatore a continue manovre per evitarne la collisione. Altra aggiunta, questa volta puramente estetica, è stata quella da parte di Peter Samson, che giudicava interessante il gioco ma non apprezzava la casualità della volta celeste sullo sfondo: programmò così il gioco in modo tale che riproducesse la reale posizione e luminosità delle stelle visibili da una persona posta sull'equatore. Questa versione venne scherzosamente soprannominata Expensive Planetarium (planetario costoso).
Modifiche minori includono un contatore di punteggio in ottale, un fattore di accelerazione delle navi e la possibilità di sparare più di un missile alla volta.

Si era scoperta la vera funzione ed uso del computer: giocare
Due altri elementi dello spirito GNU-Linux: **l'aspetto ludico e divertirsi in compagnia è più divertente, condividere le proprie passioni, le scoperte, le abilità è bello e proficuo**.
La tecnologia è divertente, programmare è divertente, far funzionare i propri programmi è divertente, condividere è utile e divertente

Dennis Ritchie Ken Thompson

Unix

Terminal

```
-rwxr-xr-x 1 sys     52850 Jun  8  1979 hptmunix
drwxrwxr-x 2 bin       320 Sep 22 05:33 lib
drwxrwxr-x 2 root       96 Sep 22 05:46 mdec
-rwxr-xr-x 1 root    50990 Jun  8  1979 rkunix
-rwxr-xr-x 1 root    51982 Jun  8  1979 rl2unix
-rwxr-xr-x 1 sys     51790 Jun  8  1979 rphtunix
-rwxr-xr-x 1 sys     51274 Jun  8  1979 rptmunix
drwxrwxrwx 2 root       48 Sep 22 05:50 tmp
drwxrwxr-x12 root      192 Sep 22 05:48 usr
# ls -l /usr
total 11
drwxrwxr-x 3 bin       128 Sep 22 05:45 dict
drwxrwxrwx 2 dmr        32 Sep 22 05:48 dmr
drwxrwxr-x 5 bin       416 Sep 22 05:46 games
drwxrwxr-x 3 sys       496 Sep 22 05:42 include
drwxrwxr-x10 bin       528 Sep 22 05:43 lib
drwxrwxr-x11 bin       176 Sep 22 05:45 man
drwxrwxr-x 3 bin       208 Sep 22 05:46 mdec
drwxrwxr-x 2 bin        80 Sep 22 05:46 pub
drwxrwxr-x 6 root       96 Sep 22 05:45 spool
drwxrwxr-x13 root      208 Sep 22 05:42 src
# ls -l /usr/dmr
total 0
#
```

Due grandi protagonisti. Lavorano nei laboratori di ricerca dell'ET&T, la Telecom privata americana. Da tempo si pensava di costruire un sistema operativo che garantisse:
1) Orientato all'uso tramite terminali remoti
2) Disponibilità continua analoga ai servizi elettrici e telefonici
3) Ampia possibilità di configurare il sistema, senza riorganizzare l'architettura
4) Alta affidabilità del file system
4) Supporto per la condivisione delle informazioni
5) Supporto di molte applicazioni
6) Supporto di diversi linguaggi di programmazione
7) Possibilità di aggiornare il sistema in base all'evoluzione tecnologica e alle aspirazioni degli utenti
Ambiziosissimo progetto dal nome Multics opera del MIT, dei Bell Labs e della General Electric. Ma troppo ambizioso stentava ad apparire e fu abbandonato
Dennis Ritchie e Ken Thompson non erano d'accordo e decisero di continuare,nasce UNIX (in contrapposizione a MULTICS)
Gli obiettivi sono:
1) si fa in modo che ogni programma faccia una sola cosa e bene;
2) ci si aspetta che l'output di un programma diventi l'input di un altro;
3) si sviluppa software con l'idea che esso verrà provato subito: non si esiti a condividere il programma;
4) si usano degli strumenti appositi nella programmazione e non si cerca di "reinventare la ruota". Giacchè ci sono inventano il linguaggio per costruire UNIX.il C

Un altro aspetto di GNU-Linux: **se vuoi una cosa e non c'è, fattela**
La filosofia di UNIX è spesso denominata KISS (sotto la spiegazione dell'acronimo) è GNU-Linux l'eredita in pieno.
Lo UNIX dell'At&T viene regalato alle università ad un prezzo simbolico con il codice sorgente, ma subito nascono degli UNIX proprietari, a pagamento forniti senza il codice sorgente. La storia di UNIX è lunga, affascinante, complessa e triste, per molti versi, quelli commerciali, ma la filosofia UNIX è all'origine dei GNU-Linux

KISS

- KISS, Keep It Simple Stupid. “Fallo semplice stupido”
- Questa è la filosofia di Unix: Scrivere un programma che faccia una sola cosa e la faccia bene. Scrivere programmi che lavorino insieme. Scrivere programmi che usino testo perché il testo è l'interfaccia universale. (Doug McIlroy)

Richard Stallman

Nei tardi anni settanta e primi anni ottanta, la cultura libera del laboratorio di IA del MIT che Stallman portava avanti cominciò a frammentarsi. Per evitare che il software fosse usato sui computer dei concorrenti, la maggior parte dei produttori smise di distribuire il codice sorgente e cominciò a usare Copyright e licenze restrittive per il software, per limitarne o proibirne la copia e la redistribuzione. Tale proprietà dei software già esisteva prima, e divenne chiaro che sarebbe diventata la norma.
Qualche aneddoto: nel 1977 il laboratorio per le scienze del computer (LCS) del MIT installò un sistema di accesso limitato controllato da password Stallman trovò un modo per decrittare le password e inviò agli utenti un messaggio contenente la loro password decodificata, con il suggerimento di cambiarla con una stringa vuota (che significa non immettere password), per permettere nuovamente l'accesso anonimo al sistema. Circa il 20% degli utenti seguirono il suo consiglio, sebbene gli accessi con password alla fine prevalsero. Stallman si vantò del successo nella sua campagna per molti anni a seguire.
Le password limitavano l'accesso ai dati ed alle informazioni, oltre a permettere agli amministratori di controllare, permettere e proibire l'uso delle risorse agli utenti. Questo era contrario alla logica della condivisione, della creatività personale, insomma contrario alla libertà e per di più era stupido, perché limitare la libertà delle persone impedisce loro di sviluppare appieno la loro creatività, personalità e socialità.

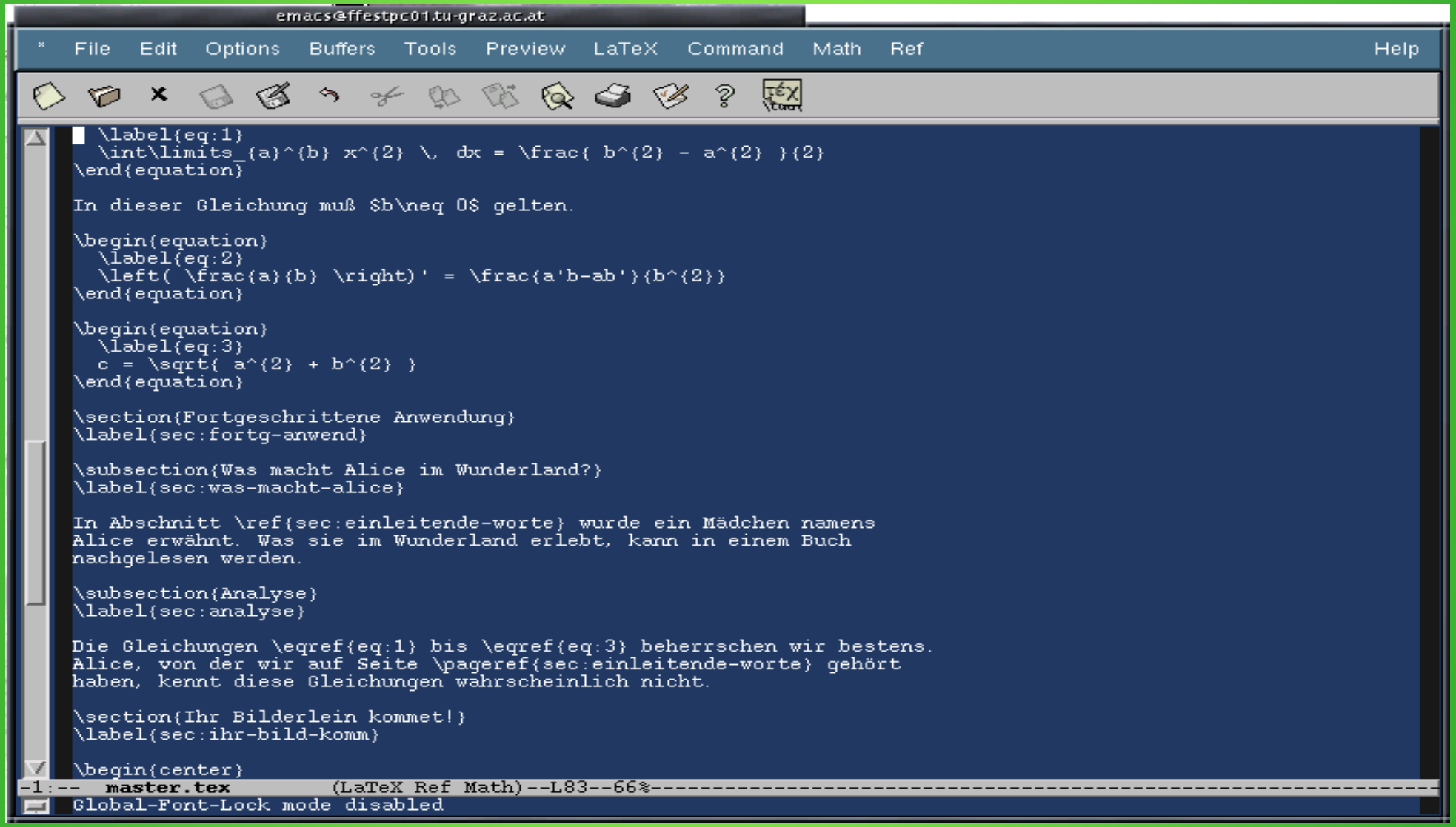
Nel 1980, a Stallman e ad alcuni altri hacker del laboratorio di intelligenza artificiale fu rifiutato il codice sorgente del software per la stampante laser Xerox 9700 (“Dover”), la prima del settore. Stallman aveva modificato il software su una vecchia stampante (la XGP, Xerographic Printer); con la sua modifica l'utente riceveva dalla stampante un messaggio elettronico che gli segnalava il completamento della stampa da lui richiesta. Tutti gli utenti in coda di stampa venivano anche avvertiti di condizioni di congestione di code di stampa, in modo che l'utente si potesse attendere un ritardo nella stampa, e la informazione necessaria per evitare altre congestioni. Vedere impedito questo servizio aggiuntivo, per l'indisponibilità del codice sorgente della stampante, non era un inconveniente trascurabile, dato che, come accadeva spesso, la stampante era unica per diversi utenti su diversi piani, e presso la stampante finivano per sostare diverse persone, a perdere tempo prezioso in attesa, o a districarsi fra le altrui stampe non ritirate. Questa esperienza convinse Stallman che le persone hanno bisogno di essere libere di modificare il software che usano.
Ancora la stupidità delle limitazioni alla libertà che impedisce a chi ha acquistato qualcosa di migliorarla di adattarla alle proprie esigenze e di distribuire i propri miglioramenti agli altri. Questo modello di sviluppo è arbitrario, violento, ma soprattutto stupido. Ci dobbiamo tenere una stampante che funziona male perché chi l'ha costruita, pur avendola venduta ed io pagata, non vuole che io sappia come funziona e dire che il miglioramento potrebbe essere utile anche al costruttore della stampante che potrebbe inserire il miglioramento nel software che fornisce alla clientela, ma il venditore preferisce considerare il cliente uno schiavo a cui proibire di capire e conoscere. Questa mancanza di rispetto per la libertà ed i diritti degli utenti spinge Stallman a lasciare il MIT e fondare il progetto GNU

GNU

Il progetto GNU è un progetto collaborativo lanciato il 27 settembre 1983 da Richard Stallman il cui scopo ultimo è la creazione di un sistema operativo composto esclusivamente da software libero chiamato GNU.

Il nome GNU è l'acronimo ricorsivo di "GNU's Not Unix".

Emacs


emacs@ffestpc01.tu-graz.ac.at
File Edit Options Buffers Tools Preview LaTeX Command Math Ref Help
\label{eq:1}
\int\limits_{a}^{b} x^{2} \, dx = \frac{ b^{2} - a^{2} }{2}
\end{equation}
In dieser Gleichung muß $b\neq 0$ gelten.
\begin{equation}
\label{eq:2}
\left(\frac{a}{b} \right)' = \frac{a'b-ab'}{b^{2}}
\end{equation}
\begin{equation}
\label{eq:3}
c = \sqrt{ a^{2} + b^{2} }
\end{equation}
\section{Fortgeschrittene Anwendung}
\label{sec:fortg-anwend}
\subsection{Was macht Alice im Wunderland?}
\label{sec:was-macht-alice}
In Abschnitt \ref{sec:einleitende-worte} wurde ein Mädchen namens
Alice erwähnt. Was sie im Wunderland erlebt, kann in einem Buch
nachgelesen werden.
\subsection{Analyse}
\label{sec:analyse}
Die Gleichungen \eqref{eq:1} bis \eqref{eq:3} beherrschen wir bestens.
Alice, von der wir auf Seite \pageref{sec:einleitende-worte} gehört
haben, kennt diese Gleichungen wahrscheinlich nicht.
\section{Ihr Bilderlein kommet!}
\label{sec:ihr-bild-komm}
\begin{center}
-1:-- master.tex (LaTeX Ref Math)--L83--66%
Global-Font-Lock mode disabled

GCC e Make

- Nessuna immagine, ma senza non c'è software
- Cosa fanno GCC e Make?

Linguaggio Macchina

```
01010001 01110101 01100101 01110011 01110100
00100000 01110110 01101111 01101100 01110100
00100000 01101001 01101100 00100000 01110000
01110011 01110100 00100000 01101100 01101111
01110011 01100011 01110010 01101001 01110110
00100000 01101001 01101110 00100000 01100010
01101110 01100001 01110010 01101001 01101111
00001101 00001010 01000001 00100000 01100010
01101111 01101110 00100000 01100011 01101111
```

Linguaggio Assembler

DDD: /media/disk-1/copia/home/fulvio/documenti/dispense/assembly-linux/allegati/programmi-assembly/op

File Edit View Program Commands Status Source Data Help

(): operazioni4.s:13

```
/*
Programma:      operazioni4.s
Autore:         Ferroni.Fulvio
Data:           28/08/2008
Descrizione:    Esempio per lo svolgimento di alcune operazioni
*/
.data
var1:    .byte 127
.text
.globl _start
_start:
   nop
   negb (var1)
   notb (var1)
   movb $15, %al
   andb $35, %al
   movb $15, %al
   orb  $35, %al
fine:
   movl $1, %eax
   int  $0x80
```

```
0x08048075 <_start+1>:   negb    0x8049090
0x0804807b <_start+7>:   notb    0x8049090
0x08048081 <_start+13>:  mov     $0xf,%al
0x08048083 <_start+15>:  and     $0x23,%al
```

```
(gdb) run

Breakpoint 1, _start () at operazioni4.s:13
Current language:  auto; currently asm
(gdb)
```

Breakpoint 1 at 0x8048075: file operazioni4.s, line 13.

DDD: Run, Interrupt, Step, Stepi, Next, Nexti, Until, Finish, Cont, Kill, Up, Down, Undo, Redo, Edit, Make

Linguaggio C

```
#include <iostream>main()
{
        char s,risp;
        int num1,num2,som;
        printf("Programma di prova sulla somma e la struttura
IF\n");

        printf("Inserire il primo numero e premere INVIO:");
        scanf("%d", &num1);
        printf("Inserire il secondo numero e premere
INVIO:");
        scanf("%d",&num2);
        som = num1 + num2 + 1;
        printf("Il risultato della somma e': %d\n",som);
        printf("Tutto bene? (s/n):");
        scanf("%c", &risp);
        if (risp = s)
                printf("***** mi hai scoperto! Ah ho imparato
il cilo IF");
        else
        {
                printf("Ehi, mi sa che non sai contare!");
}
        return(0);
}
```

Il lavoro del compilatore: tradurre

```
01010001 01110101 01100101 01110011 01110100
00100000 01110110 01101111 01101100 01110100
00100000 01101001 01101100 00100000 01110000
01110011 01110100 00100000 01101100 01101111
01110011 01100011 01110010 01101001 01110110
00100000 01101001 01101110 00100000 01100010
01101110 01100001 01110010 01101001 01101111
00001101 00001010 01000001 00100000 01100010
01101111 01101110 00100000 01100011 01101111
```

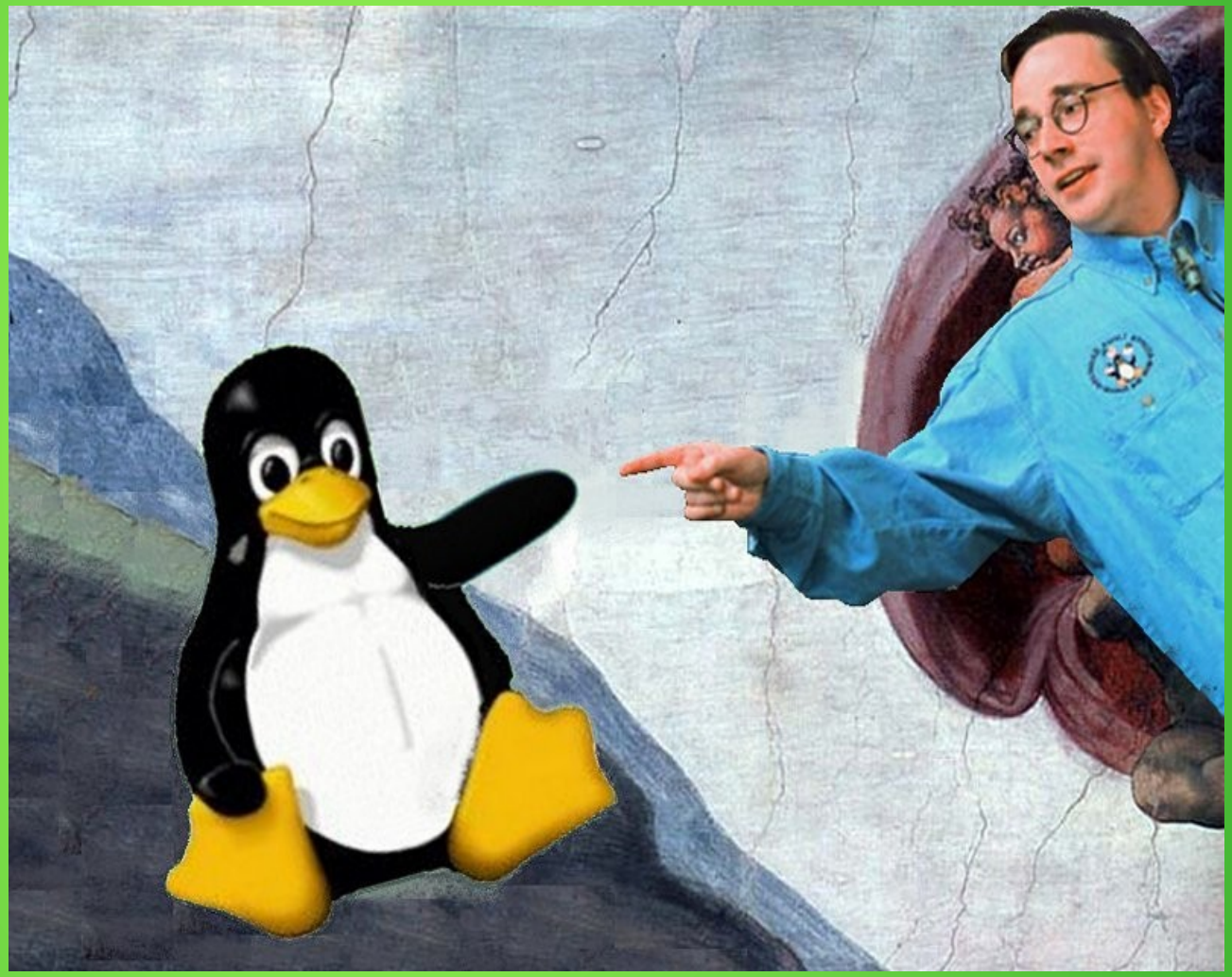
La liberta del software

- Liberta 0: Libertà di eseguire il programma come si desidera, per qualsiasi scopo.
- Libertà 1: Libertà di studiare come funziona il programma e di modificarlo in modo da adattarlo alle proprie necessità. L'accesso al codice sorgente ne è un prerequisito.
- Libertà 2: Libertà di ridistribuire copie in modo da aiutare il prossimo.
- Libertà 3: Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti da voi apportati (e le vostre versioni modificate in genere), in modo tale che tutta la comunità ne tragga beneficio. L'accesso al codice sorgente ne è un prerequisito.

Il progetto GNU e le sue quattro libertà sono i fondamenti di GNU-Linux. Un sistema operativo libero, che non significa gratuito, rispettoso dei diritti dell'utente ed aperto al contributo di tutti condiviso.
Con GNU e le licenze GPL (GNU General Public License) si realizza la filosofia che abbiamo descritto.
Il progetto fornisce tutti gli strumenti utili a a costruire un sistema simile ad UNIX, ma che non è UNIX per non incorrere nel copyright che affligge UNIX
Manca però ancora qualcosa...

Linus Torvalds

Tux or Torvalds UniX



Il post di nascita

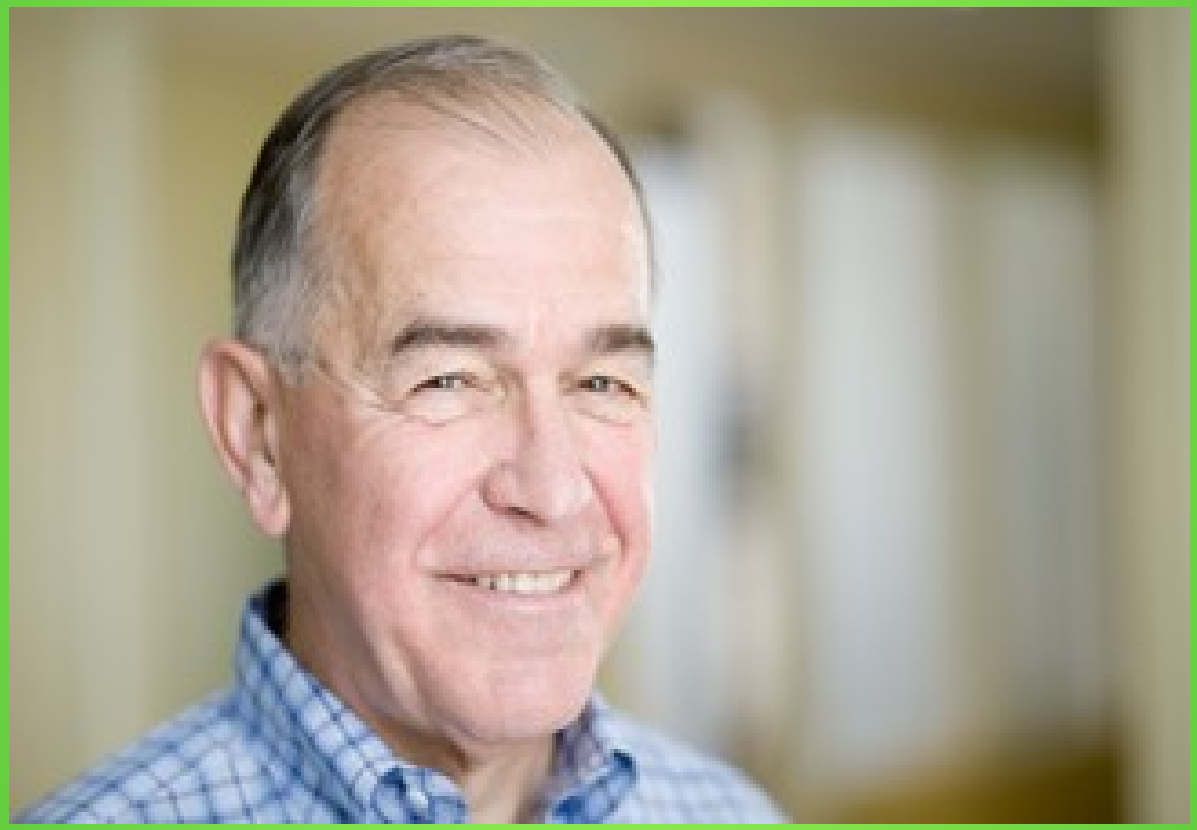
Sto programmando un sistema operativo (gratuito e solo per hobby, non vuole essere grande e professionale come GNU) per cloni di AT 386(486). È in preparazione da aprile, e sta iniziando a funzionare. Mi piacerebbe sapere cosa vi piace e non vi piace in Minix, siccome il mio Sistema Operativo gli assomiglia in parte (fra le altre cose, lo stesso layout fisico del filesystem, per ragioni pratiche).

Ho convertito la shell bash (v.1.08) e GCC (v.1.40), e sembrano funzionare. Ciò denota che otterrò qualcosa di funzionante in pochi mesi e mi piacerebbe sapere quali funzionalità vuole la maggior parte della gente. Ogni suggerimento è ben accetto, anche se non posso promettervi che lo implementerò. »

Linus Torvalds costruisce un kernel per 386 per divertimento e lo regala alla comunità, che lo amplia, lo perfezione, ne inserisce nuove caratteristiche.
Abbiamo un kernel.
Usando il kernel Linux con il software GNU, in attesa che nasca il kernel a cui sta lavorando Stallman, sta lavorando sempre meno perché impegnato nel movimento per il Free Software, insieme ai programmi GNU abbiamo un sistema aperto, libero, efficiente come UNIX, ma esente dai difetti di UNIX legati alla sua commercializzazione.

E' nato GNU-Linux

Stephen “Steve” Bourne



BASH

- Bourne again Shell. **“Un'altra shell Bourne”,** ma **“Bourne again”** ha la stessa pronuncia di **“born again”**, cioè rinata creando la definizione finale **“shell rinata”**.

Bash

```
chealer@vinci:/usr/share/doc/bash$ export LC_ALL=C
chealer@vinci:/usr/share/doc/bash$ cd ~chealer/
chealer@vinci:~$ ls
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
chealer@vinci:~$ #Why is there color when calling ls without arguments?
chealer@vinci:~$ which ls
/bin/ls
chealer@vinci:~$ $(!!)
$(which ls)
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
chealer@vinci:~$ type ls #"ls" doesn't just run /bin/ls
ls is aliased to `ls --color=auto'
chealer@vinci:~$ echo $PS1
${debian_chroot:+($debian_chroot)}\u@\h:\w\$
chealer@vinci:~$ sh
sh-3.1$ echo $PS1
\s-\v\$
sh-3.1$ echo $BASH_VERSION
3.1.17(1)-release
sh-3.1$ ls
Cloutier  Ido         Musique  logs      skolo  sources
Desktop   Mes images  boston   ncix.png  smb4k  vieux
sh-3.1$ echo $SHELLOPTS # ls isn't an alias in POSIX mode
braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor:posix
sh-3.1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
sh-3.1$ /bin/kill &> killerror # collect stdout and stderr of $ /bin/kill; in ki
llerror
sh-3.1$ wc -l !$
wc -l killerror
7 killerror
sh-3.1$ type kill # kill doesn't just run /bin/kill, even in POSIX mode.
kill is a shell builtin
sh-3.1$ !$ -n 9 $$ # OK, kill self
kill -n 9 $$ # OK, kill self
Killed
chealer@vinci:~$
```

BASH è una shell, il GNU-Linux ce ne son moltissime. La shell e' un interprete di comandi che permette all'utente di comunicare col sistema operativo attraverso una serie di funzioni predefinite, o di eseguire programmi. Semplice, conforme alla filosofia KISS, BASH permette di fare tutto quello che si vuole con il proprio computer.

Se volete un sistema Linux potete compilare un kernel linux, utilizzando GCC, poi potete compilare una shell ed i software che vi servono.
Si parte da un computer funzionante si compila ciò che serve e si copia sul nuovo computer GNU-Linux. Poi verranno interfaccia grafiche ed altro, ma per lavorare e divertirsi si ha tutto il necessario

Eric Raymond

- La cattedrale ed il Bazar

Eric Raymond è un controverso personaggio autore del testo La cattedrale ed il bazar. Il saggio descrive in sostanza due contrapposte modalità di sviluppo del software libero: Nel modello a Cattedrale il programma viene realizzato da un numero limitato di "esperti" che provvedono a scrivere il codice in quasi totale isolamento. Il progetto ha una suddivisione gerarchica molto stretta e ogni sviluppatore si preoccupa della sua piccola parte di codice. Le revisioni si susseguono con relativa lentezza e gli sviluppatori cercano di rilasciare programmi il più possibile completi e senza bug. Il programma Emacs, il GCC e molti altri programmi si basano su questo modello di sviluppo.
Nel modello a Bazaar il codice sorgente della revisione in sviluppo è disponibile liberamente, gli utenti possono interagire con gli sviluppatori e se ne hanno le capacità possono modificare e integrare il codice. Lo sviluppo è decentralizzato e non esiste una rigida suddivisione dei compiti, un programmatore di buona volontà può modificare e integrare qualsiasi parte del codice. In sostanza lo sviluppo è molto più diluito e libero, da qui il nome di modello a Bazaar. Il Kernel Linux e molti programmi utilizzano questo nuovo modello di sviluppo associativo.

La tesi centrale di Raymond è che "Dato un numero sufficiente di occhi, tutti i bug vengono a galla". Questa affermazione (che Raymond chiama "Legge di Linus") costituisce, a suo parere, il motivo centrale del successo del progetto del Kernel Linux. Prima dell'avvento di Linux si riteneva che ogni progetto di una certa complessità avesse bisogno di essere adeguatamente gestito e coordinato: in caso contrario il progetto sarebbe collassato sotto il peso di moltissime revisioni e modifiche, prodotte da più persone e quindi spesso incompatibili. Il progetto Linux riuscì a dimostrare che non solo questo non accadeva, ma che al contrario al crescere del numero di sviluppatori anche la qualità e l'affidabilità del software migliorava.

Il modello a Cattedrale è un modello tipico delle aziende commerciali. Queste normalmente non rilasciano il codice sorgente, e una nuova revisione del programma può richiedere anni; viceversa, il modello a Bazaar è un modello che si è molto diffuso nell'ambiente del software libero, poiché consente a ogni utente di ricoprire il ruolo di beta tester dei programmi. Lo stesso utente può perfino modificare il programma, se lo desidera: questo consente un rapporto stretto tra utilizzatori e programmatori, un rapporto paritario che ben si adatta alla filosofia del software libero. Consente inoltre un attento controllo del codice, aiutando a eliminare rapidamente la maggior parte dei bug; questione invece complessa per un software prodotto con la modalità a Cattedrale, dove solo un numero limitato di persone lavora sul codice.

Eric Raymond ed altri hanno innescato alcune polemiche con Stallman. In sostanza Raymond ritiene che ciò che giustifica il modello GNU-Linux sia l'efficienza del software, mentre Stallman obietta che ciò che conta non è che sia superiore al modello commerciale, ma che sia libero. Disquisizioni filosofiche , che spesso sono diventate roventi e che diversificano il modello Free Software (Stallman) da quello, più conoscito e questo è fnte di dolore per Stallman, dell'Open Source. Se vi capita di incontrare Stallman non usate questo termine (open source) o fatelo a vostro rischio e pericolo. Di seguito le cartteristiche generali delle due licenze.

Le quattro libertà offerte dal free software:

0- Libertà di usare il programma senza impedimenti;

1- Libertà di aiutare sé stesso studiando il codice disponibile e modificandolo in base alle proprie esigenze;

2- Libertà di aiutare il tuo vicino, cioè la possibilità di distribuire copie del software rielaborato, rendendolo così accessibile a tutti;

3- Libertà di pubblicare una versione modificata del software;

Principi base dell'open source:

1) Libertà di redistribuzione (sta poi al singolo decidere se farlo gratuitamente o se far pagare il prodotto);

2) Libertà di consultare il codice sorgente;

3) Necessità di approvazione per i prodotti derivati;

4) Integrità del codice sorgente dell'autore;

5) Nessuna discriminazione verso singoli o gruppi di persone;

6) Nessuna discriminazione verso i settori di applicazione;

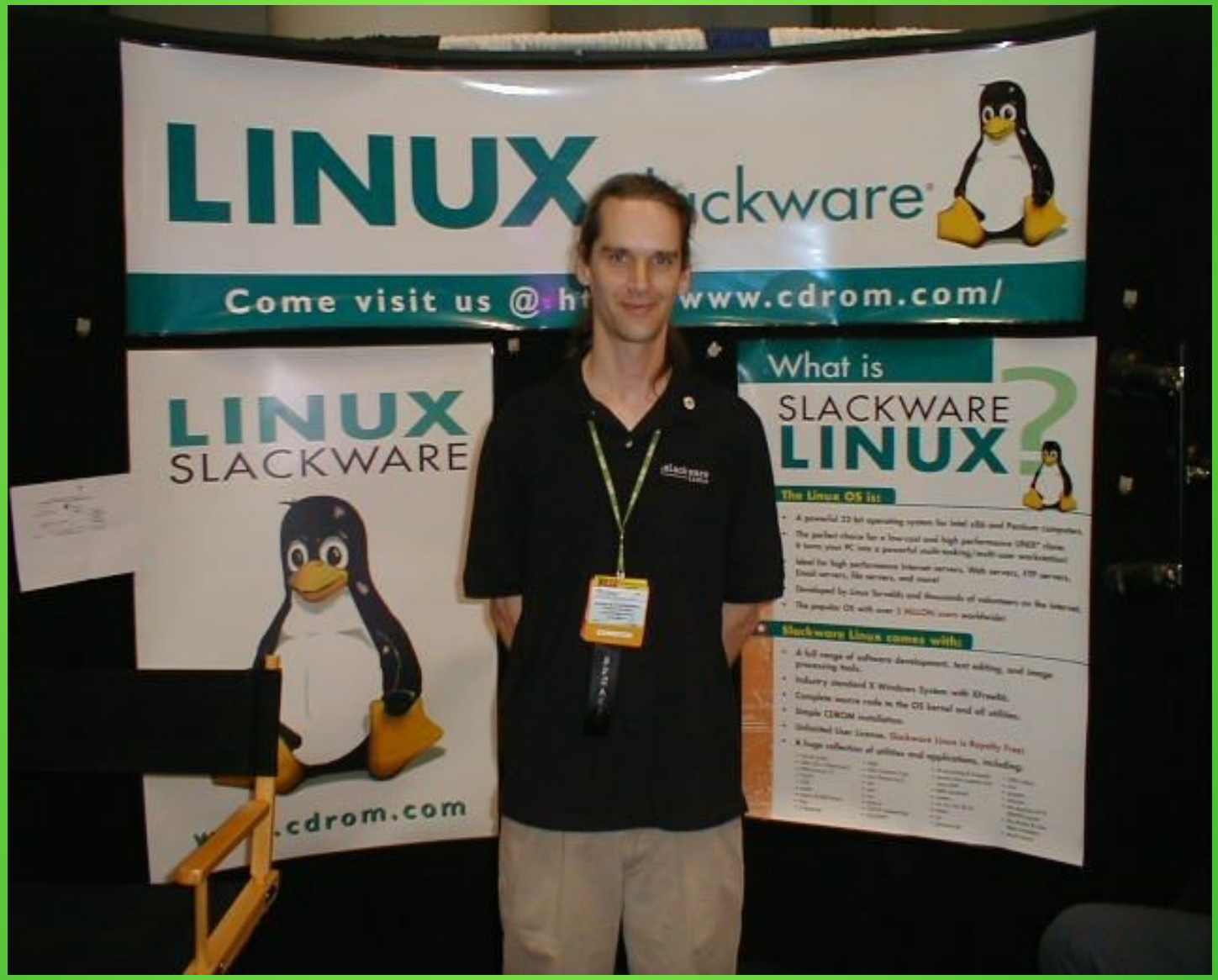
7) La licenza deve essere distribuibile;

8) La licenza non può essere specifica per un prodotto;

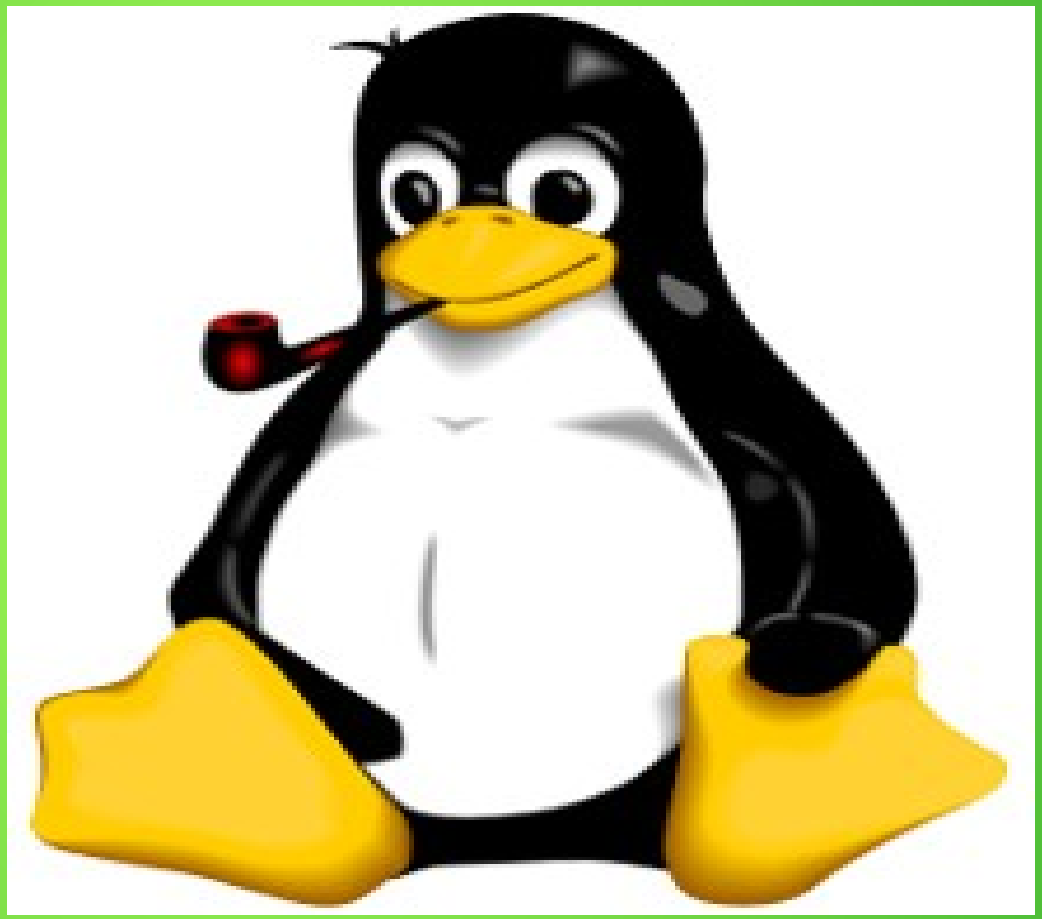
9) La licenza non può contaminare altri software;

10) La licenza deve essere tecnologicamente neutrale;

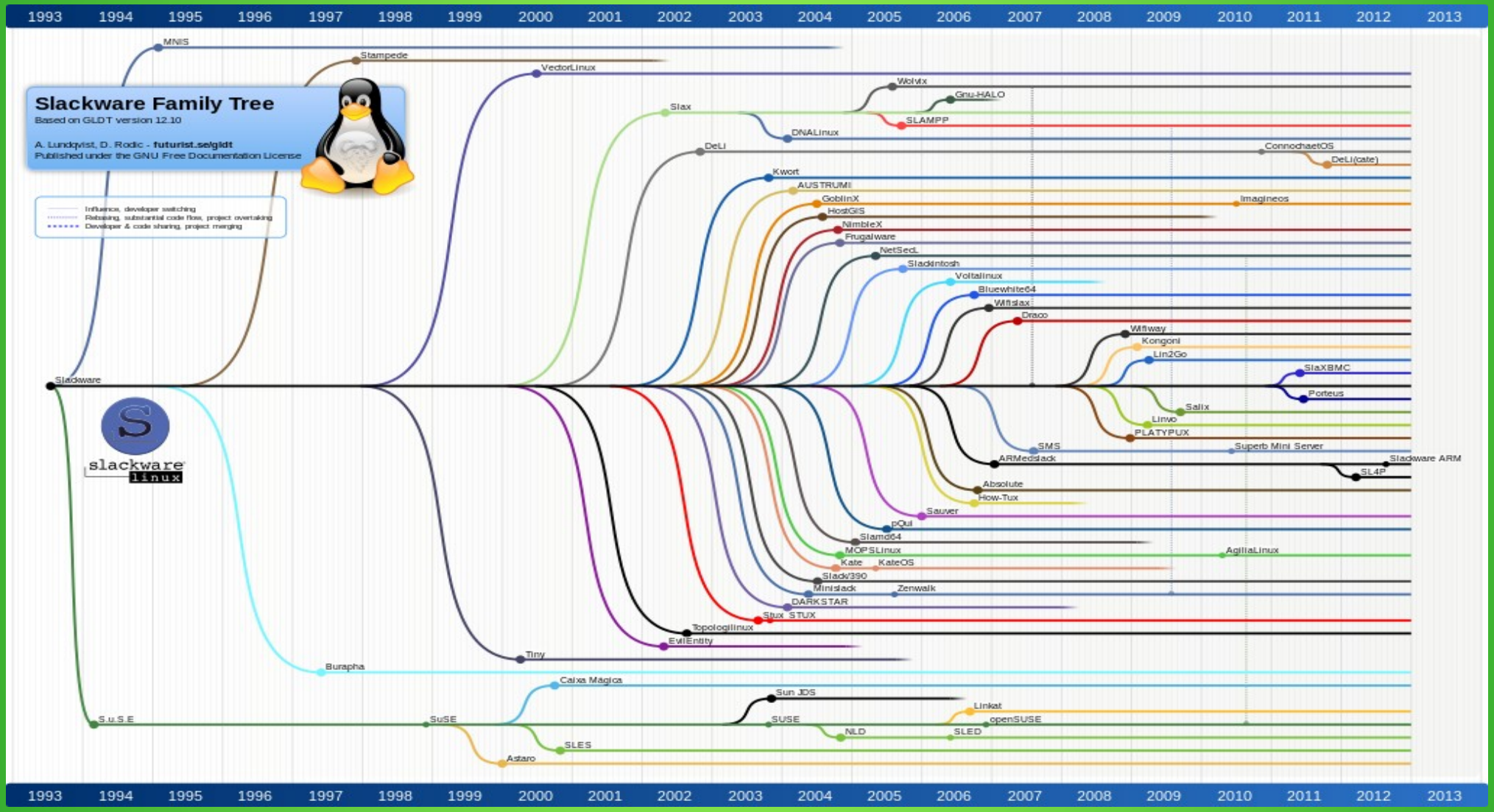
Patrick J. Volkerding (16 luglio 1993)



Slackware



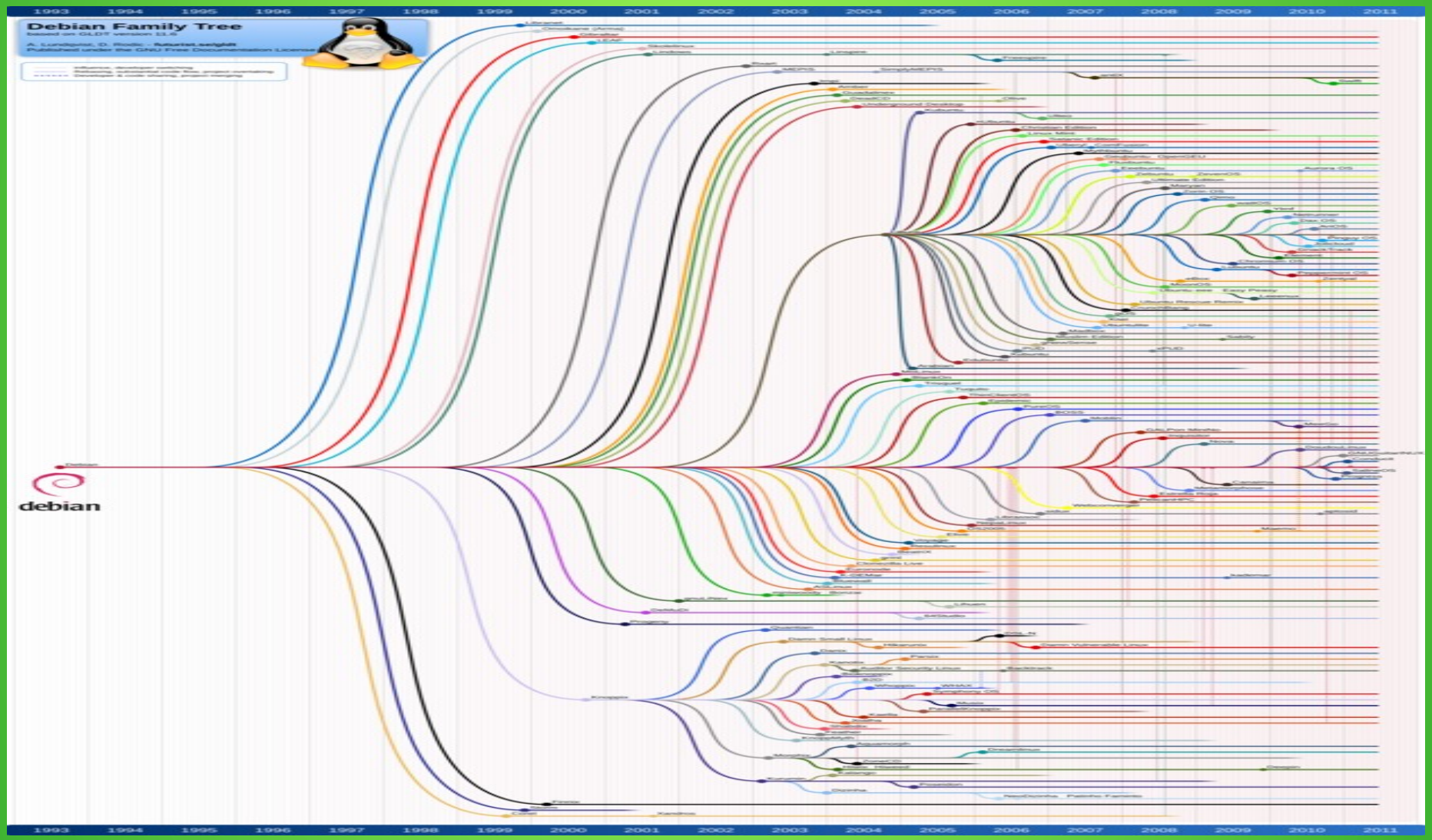
Famiglia Slackware



C'era tutto (kernel, shell, programmi), ma compilare tutto non è semplice. Così nasce l'idea di creare una distribuzione. Infatti chiunque volesse far uso di Linux doveva necessariamente essere un esperto di Unix, con la consapevolezza non solo di quali librerie ed eseguibili fossero necessari per far partire e funzionare il sistema, ma anche di alcuni importanti dettagli riguardanti la configurazione ed il posizionamento dei file nel sistema stesso.
Una distribuzione consiste in una serie di utility che, partendo da un cd-rom avviabile (prima era un floppy) aiuta l'utente a compiere i passaggi necessari per ottenere un sistema GNU-Linux funzionante. Comprende anche software per la configurazione e spesso per l'installazione dei programmi
Slackware fu la prima distribuzione ancora esistente e supportata.
Slackware è la distribuzione più vicina alla filosofia UNIX, è spartana, poche sono le utility offerte e spesso accessibili a riga di comando, è leggera ed efficiente. Da Slackware son nate moltissime diverse distribuzioni.
Il nome Slackware deriva da Slack (scritto con l'iniziale maiuscola), termine non ben definito che nel contesto della satirica Chiesa del SubGenio indicherebbe la facoltà di raggiungere i propri obiettivi senza sforzo, perseguendo una vita libera e confortevole, priva di responsabilità e duro lavoro. Nella fattispecie, Slackware consentirebbe di raggiungere lo Slack attraverso la semplicità d'uso, l'affidabilità, la durevolezza e il rifiuto di strumenti mainstream non necessari

Ian Murdock (16 agosto1993)

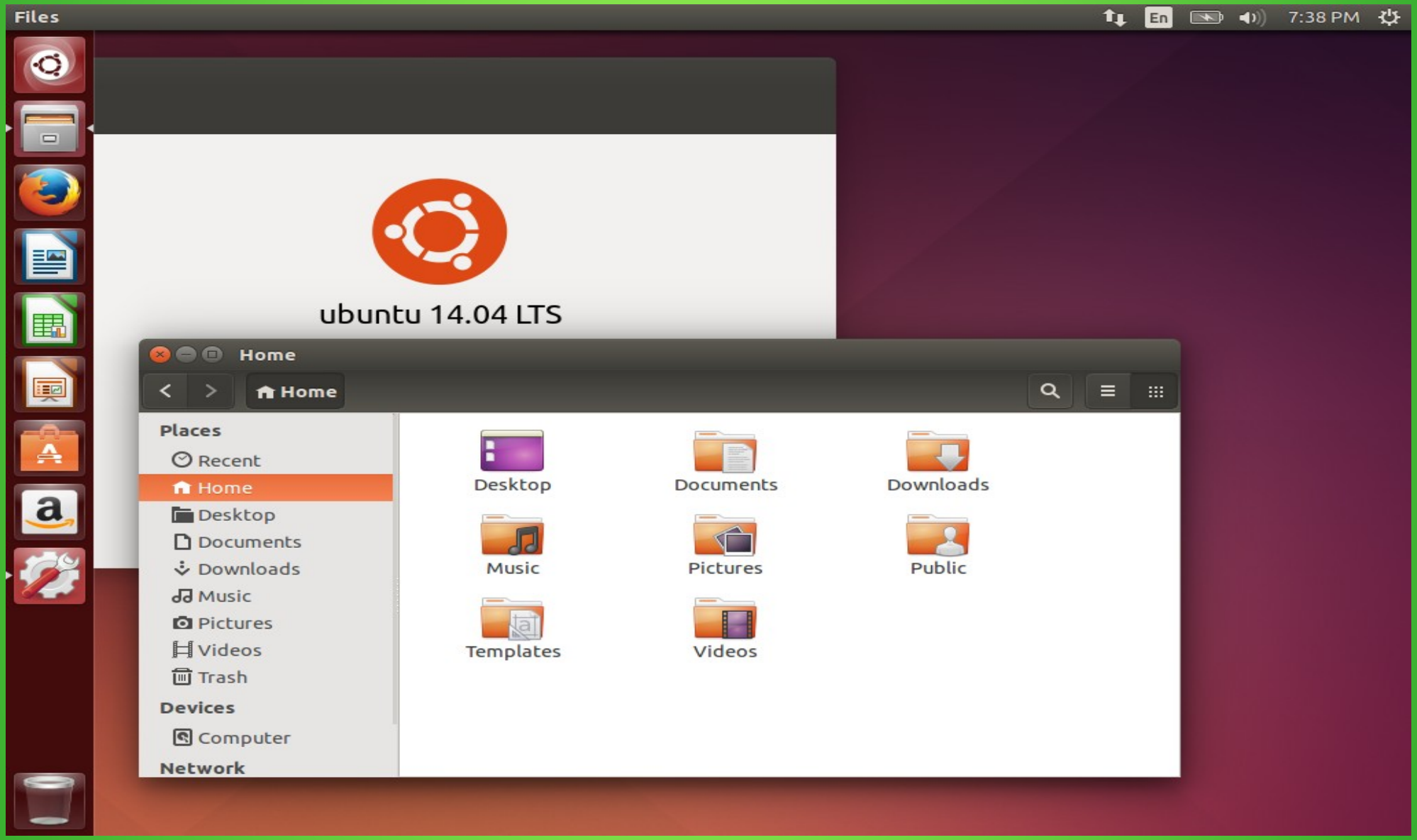
Debian Debra e Ian



Debian è la distribuzione di riferimento per il movimento Free Software. Infatti non installa software proprietario di default. Semplice, efficiente e pulita e dotata di tutto ciò che serve per semplificare la vita all'utente compreso un ottimo installatore di pacchetti, almeno quello a riga di comando APT
Installare Debian è facile divertente e si ha un completo sistema GNU-Linux.
Il nome Debian è stato coniato dal fondatore del progetto, Ian Murdock, unendo il proprio nome e le prime tre lettere di quello della sua fidanzata (ora ex-moglie) Debra.
Da Debian derivano moltissime distribuzioni di successo.
Compreso ahime l'ultima.

Mark Shuttleworth

Ubuntu


Files
En
7:38 PM
ubuntu 14.04 LTS
Home
Home
Places
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
Computer
Network
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos

Ubuntu è una delle ultime arrivate tra le distribuzioni GNU-Linux. Forse è la più installata, specie tra i nuovi utenti abituati a Windows.
La sua principale caratteristica è la focalizzazione sull'utente e la facilità di utilizzo, oltre all'ottimo supporto dell'hardware di tutti i tipi.
Installando UBUNTU si può ignorare l'esistenza della riga di comando, ed infatti molti proprio ne ignorano l'esistenza.
Il nome Ubuntu deriva da un antico vocabolo zulu (letteralmente: umanità) diffuso in varie parti dell'Africa meridionale. Il termine viene utilizzato nel detto zulu «umuntu ngumuntu ngabantu», traducibile con "io sono ciò che sono per merito di ciò che siamo tutti".
E' la più lontana dalle origini spartane (KISS) di Unix e questo non la fa apprezzare da chi ama la semplicità e l'efficienza. Il voler essere semplice da usare, il cercare di ridurre al massimo la curva di apprendimento, cosa in cui riesce benissimo, è il suo pregio ed il suo difetto.
Ma la forza di GNU-Linux è nella sua varietà e nella libertà che ci lascia di scegliere e fare ciò che ci piace.