

UNIVERSITÀ DI PISA



**Relazione sul 2° progetto di Advanced Topic in
Computer System and Network**

Modifica e porting su FreeBSD dell'applicazione congif

Candidato:
Giovanni Pollina

Supervisore:
Prof. Luigi Rizzo

Si vuole modificare una applicazione già esistente per linux, scritta in c, aggiungendo nuove funzionalità e facendo il porting su Free BSD.

L'applicazione in oggetto è congif, un' applicazione che permette di trasformare in GIF animata una sessione di terminale in linux.

L'applicazione congif è disponibile su GitHub al seguente indirizzo:

<https://github.com/lecram/congif>

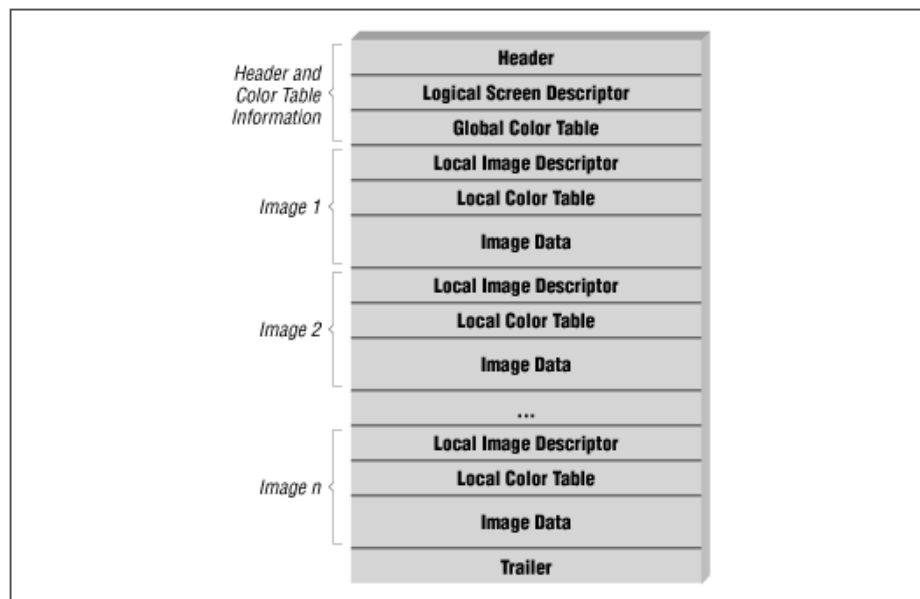
1. Standard GIF

Per prima cosa è descriviamo rapidamente come funziona lo standard GIF.

Un GIF file è formato da blocchi di dati differenti.

Ogni blocco può avere una misura fissata o variabile e contiene informazioni utili per costruire la GIF.

La parte iniziale "Header" specifica il formato del file e la versione di algoritmo GIF usato.



Per esempio:



Dove 47="G", 49="I", 46="F" e 38="8", 39="9", 61="a", avremo quindi GIF89a in esadecimale che indica l'inizio di una GIF animata.

Congif utilizza proprio questa struttura per comporre il file GIF in output.

2. Funzionamento Congif

L'applicazione per poter funzionare si serve di un programma disponibile in linux di nome script:

script [options] [*file*]

Questo comando lancia un processo in background che registra tutti i comandi inseriti nella shell compresi di istanti i cui vengono inseriti.

Lanciando il semplice comando nella shell linux:

```
$ script -t 2> foo.t foo.d
```

Otterremo in output due file:

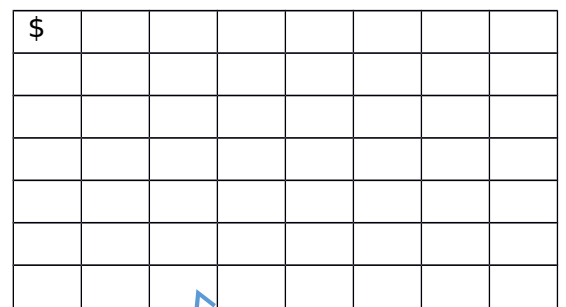
- **foo.t** che contiene due campi separati da uno spazio: il primo campo indica quanto tempo è passato prima del precedente output. Il secondo campo indica quanti caratteri sono stati immessi dallo stout in questo istante.
- **foo.d** contiene i caratteri in output

Queste informazioni sono usate da congif per riprodurre una gif che emula la sessione di terminale in modo realistico.

Per poter funzionare congif all'inizio si crea una struttura che emula il terminale, avremo quindi una matrice di elementi che corrisponde al terminale virtuale.

Ogni cella del terminale emulato viene gestito tramite una struttura che ne memorizza il:

- code: Carattere inserito
- attribute: stringa di bit che rappresenta le caratteristiche del carattere: dimensione, Bold, sottolineato ecc..
- Pair: stringa di bit dove i 4 più rappresentativi indicano il colore del carattere, i 4 meno significativi il background del carattere.



\$							

```
typedef struct Cell { //basic block of the terminal
    uint16_t code;
    uint8_t attr;
    uint8_t pair;
} Cell;
```

Una volta creato il terminale virtuale il programma effettua il rendering del terminale virtuale, solo quando il timing supera una certa soglia rd:

```
d += (MIN(t, options.maxdelay) * 100.0 / options.divisor);  
rd = (uint16_t) (d + 0.5);
```

Dove:

-**options.maxdelay** è il massimo delay tra un evento di input e un altro e si usa per evitare lunghe pause in riproduzione.

-**options.divisor** è lo speedup in riproduzione, è chiamato divisore perché divide il delay tra un frame e un altro.

Ad ogni iterazione il programma preleva il timing dal file foo.t e lo riporta nella variabile t.

Inoltre se ad ogni iterazione se la variabile rd maggiore o uguale alla variabile statica MIN_DELAY allora il programma renderizza l'immagine del terminale virtuale.

Options.maxdelay e options.divisor sono settabili dall'utente.

Questo approccio rende l'applicazione poco user-friendly dal punto di vista dell'utente e inoltre il rendering dei frame e la riproduzione dei frame non è costante ma è variabile nel tempo.

3. Modifiche all'applicazione

3.1 Possibilità di settare il sampling rate(espresso in FPS) nella creazione della GIF

Come prima modifica, si rende possibile la scelta dell' intervallo di renderizzazione in termini di FPS(frame per secondo).

1. Per ogni intervallo prelevato dal file di timing foo.t si controlla in base al frame rate scelto se è necessario inserire almeno un frame.
2. Nel caso in cui è necessario inserire almeno un frame si procede alla renderizzazione del terminale il quale si trova nello stesso stato per tutta la fase di renderizzazione, questo provoca una serie di frame identici in riproduzione.

Pro:

Maggiore usabilità, applicazione più user-friendly.

Cons:

Maggior spazio occupato dalla nuova GIF.

Comunque quest'approccio non grava dal punto di vista dello spazio occupato in memoria perché l'applicazione prima di renderizzare un frame controlla se il nuovo frame è variato rispetto al precedente, se varia si preoccupa di aggiungere solo la parte di immagine variata, che verrà sovrapposta al frame precedente (in riproduzione).

Se non varia aggiunge un frame di dimensione uguale a un pixel.

Il comando da dare per settare il rate di campionamento è:

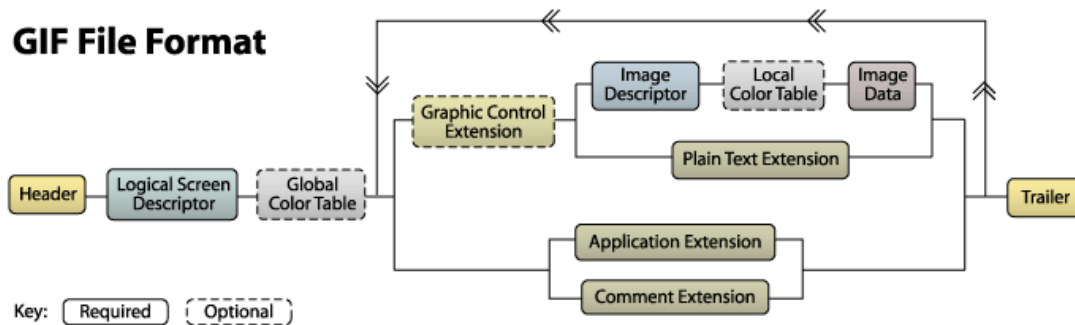
-s sampling Fix the sampling of the gif in terms of fps

3.2 Possibilità di settare la velocità di riproduzione (espressa in FPS)

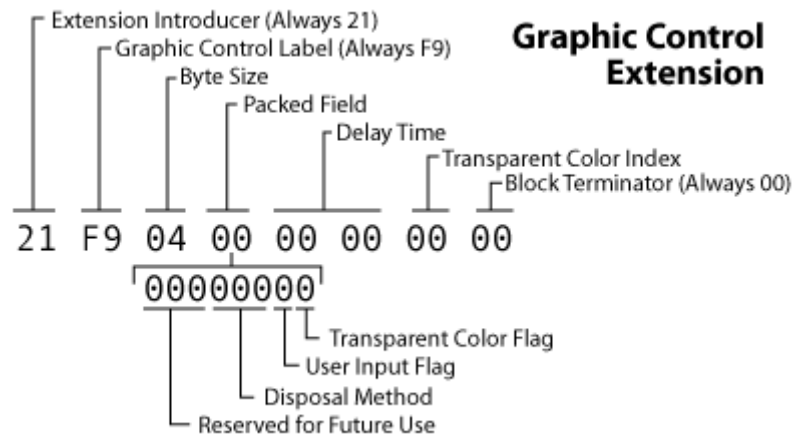
Un'altra feature aggiunta è quella di poter scegliere la velocità di riproduzione in termini di FPS.

Tramite questa opzione non si fa altro che regolare il delay-time tra un frame e un altro in riproduzione.

Per settare il delay si deve scrivere nella tabella Graphic Control Extension che compare prima di ogni frame:



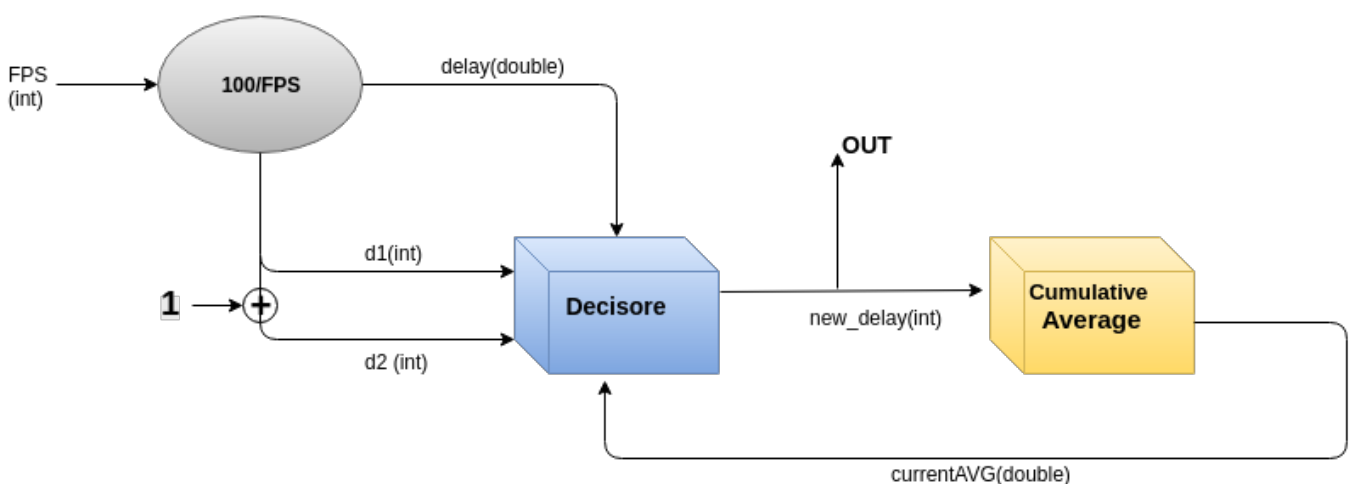
questo campo è opzionale e serve per fornire funzionalità grafiche aggiuntive in riproduzione della GIF, la Graphic Control Extension Table apparirà così:



Quindi settando i 2 byte nella tabella si può scegliere il delay nel passare da un frame all'altro.

Dato che non è possibile settare con precisione il Frame rate in riproduzione, perché il formato GIF accetta solo numeri interi espressi in centesimi di secondo per il delay tra un frame e un altro, useremo un approccio basato sulla media cumulativa, ossia useremo un delay_time variabile tra un frame e un altro ma in media avremo il frame-rate desiderato.

Per far ciò utilizzeremo questo approccio:



Tramite la formula $100/\text{FPS}$ stabiliamo il delay time esatto da inserire tra un frame e un altro per avere i FPS in riproduzione.
Purtroppo questo dato è un double e non è usabile nello standard GIF.

Quindi stabiliamo due possibili valori da inserire come delay tra un frame e l'altro nella GIF.

I due possibili valori sono $(\text{int})(100/\text{FPS})$ e $(\text{int})((100/\text{FPS})+1)$:

Tramite un esatta sequenza di questi due valori otterremo in media il delay time in double voluto, e quindi anche l'esatto Frame-rate voluto.

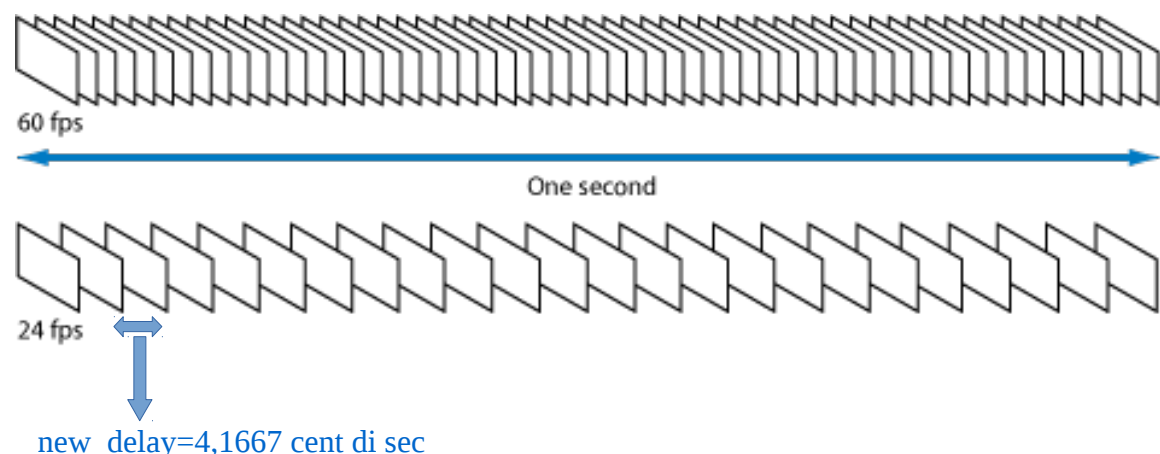
Il **Decisore** non fa altro che decidere se inserire un valore o l'altro, per esempio se vogliamo 60 FPS in riproduzione i due possibili valori saranno $(\text{int})(100/60)=1$ e $(\text{int})((100/60)+1)=2$ quindi il decisore non farà altro che scegliere tra 1 e 2, il valore scelto è inserito come delay (espresso in centesimi di sec.) nella GIF.

Il **cumulative average** effettua semplicemente la media cumulativa e comanda il decisore che sceglierà il numero inferiore se la media cumulativa supera il delay_time voluto, il numero superiore altrimenti.

Per esempio per un FPS uguale a 60 la sequenza generata dal decisore sarà esattamente:

1-2-2-1-2-2-1-2-2-1-2-2-1-2-2-2.....

E la media oscillerà tra 1,6500.. e 1,6785.. raggiungendo l'80% delle volte il valore cercato 1,66667.

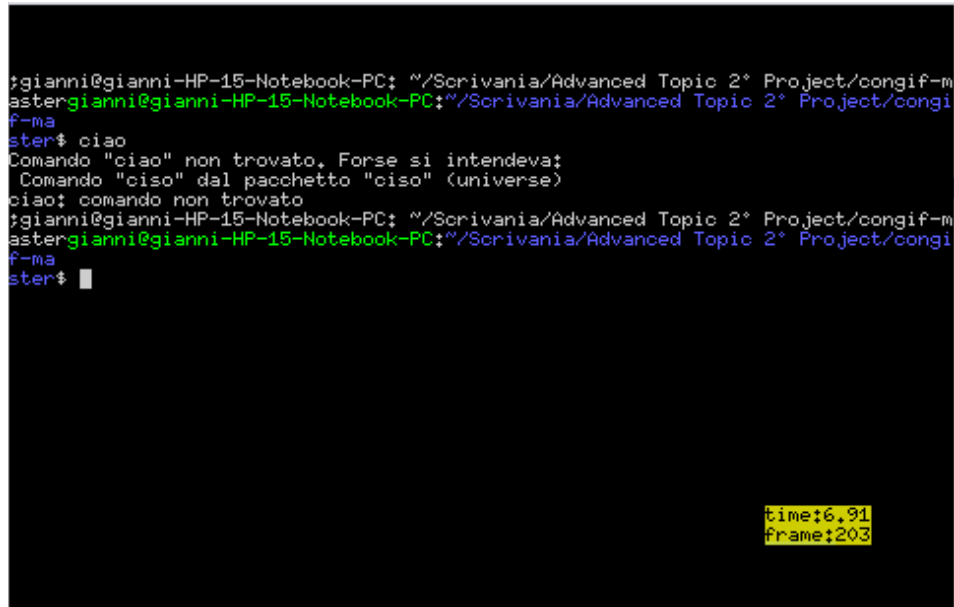


Il comando da dare per questa opzione è:

-r reproduction Fix the speed of replay in terms of fps
speed

3.3 Campo info nella GIF creata

Questa opzione dà la possibilità di inserire un box in cui vengono fornite informazioni sulla GIF creata come un timer che indica l'istante di riproduzione o un contatore che indica in numero corrente di frame

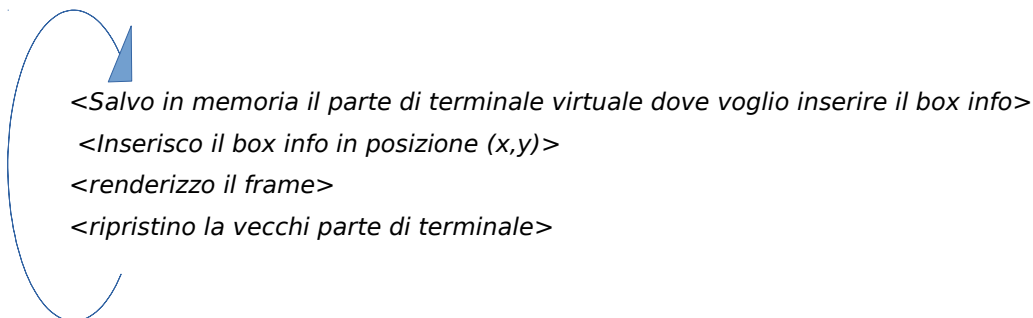


```
gianni@gianni-HP-15-Notebook-PC: ~/Scrivania/Advanced Topic 2° Project/congif-m
astergianni@gianni-HP-15-Notebook-PC:~/Scrivania/Advanced Topic 2° Project/congi
f-ma
ster$ ciao
Comando "ciao" non trovato. Forse si intendeva:
Comando "ciso" dal pacchetto "ciso" (universe)
ciao: comando non trovato
gianni@gianni-HP-15-Notebook-PC: ~/Scrivania/Advanced Topic 2° Project/congif-m
astergianni@gianni-HP-15-Notebook-PC:~/Scrivania/Advanced Topic 2° Project/congi
f-ma
ster$
```

time:6.91
frame:203

Questo box con informazioni è inserito nel terminale virtuale prima del rendering per poi essere sostituito con la parte originale, in questo modo non si intacca assolutamente il corretto funzionamento del programma originale.

In pseudo codice avrò il seguente schema:



Inoltre tramite appositi comandi si può impostare la posizione in cui visualizzare il box info e quanti informazioni voler visualizzare:

- i y|n Put info on the gif, about # of frame and timing
- x x_coord Position of info box in the x-axis
- y y_coord Position of info box in the y-axis
- n number_mode
 - 1: Show only the time counter
 - 2: Show only the frame counter
 - 3: Show the time counter and frame counter (by Default)

4.1 Porting su Free BSD

Dato l'utilizzo di librerie standard, da parte di congif, che sono anche disponibili su FBSD, non è necessario creare un patch file.

Anche il comando script per la creazione del dello script su file foo.t e foo.d è presente in Free BSD, infatti comparve per la prima volta proprio su 3.0BSD.

Per fare il porting è dunque necessario creare i file di porting, nello specifico servono il Makefiles, il distinfo e il pkg-descr.

Per prima cosa creiamo il makefile, che avrà la seguente struttura:

```
# $FreeBSD$

PORTNAME=      congif
PORTVERSION=   1.0
CATEGORIES=    graphics

MAINTAINER=    gianni.pollina@gmail.com
COMMENT=       Converts a terminal session file into
              animated gif

WRKSRCS=       ${WRKDIR}/${PORTNAME}-1.0

USES= tar:bzip2 gmake

PLIST_FILES=   bin/${PORTNAME}

.include <bsd.port.mk>

# $FreeBSD$
```

Tramite il tag USES specifichiamo che i sorgenti sono compressi con estensione .tar.bz2 e che è necessario bzip2 per la decompressione.

Comprimiamo i sorgenti con un file compresso di nome: lecam-congif-1.0_GH0.tar.bz2.

Tale file verrà posizionato in /usr/ports/distfiles su Free BSD, perché è lì che il programma di porting andrà a cercare i sorgenti una volta lanciato il makefile.

Dovrà essere creata una apposita cartella di nome `congif` nel percorso `/usr/ports/graphics/congif` e posizionarvi il `makefile` con il `distinfo` e il `pkg-descr`.

Il `distinfo` si crea con il comando `make makesum` che effettua un hash sul file compresso `/usr/ports/distfiles/giannix90-congif-1.0_GH0.tar.bz2` e ne genera il `distinfo` che serve per controllare l'integrità dei sorgenti.

Una volta iniziato il meccanismo di porting tramite il comando `make`, il programma estrarrà i sorgenti nella cartella `work` in `/usr/ports/graphics/congif/work/congif-1.0`, a questo punto verranno installati i sorgenti.