Nell'attesa e, perché no, nella speranza di ricevere programmi dei lettori non più chilometrici, ma adatti alla pubblicazione in questa rubrica, questo mese il software Amiga è composto da tre (mie) piccole utility file oriented, da tenere nella propria directory comandi

Cut & Paste file

di Andrea de Prisco

Avete presente un bel paio di forbici e un vasetto di colla? I programmi presentati questo mese fanno proprio lo stesso servizio: taglia e incolla. Ovviamente non carta, stoffa o... capelli, ma file. Il primo dei due, Cut, prende un file ed effettua su questo ben due tagli, orizzontali o verticali prelevando quindi solo la parte centrale. I tagli orizzontali sono facilmente «immaginabili»: infatti un file di testo è formato da una sequenza di righe (sequenze di caratteri terminate da un NewLine), quindi tagliare vuol dire selezionare la porzione di testo a partire da una determinata riga fino ad un'altra. In senso verticale i tagli possono essere effettuati solo su file di forma tabellare, come quelli ottenuti da uno spreadsheet o da un data base: in questo secondo caso la selezione avverrà per colonne, ad esempio dalla 34 -esima alla 128-esima.

La sintassi di questo comando è quanto mai semplice:

```
cut v|h sorgente destinazione start stop
```

```
/*************************************
*                                    *
*                                    *
*              L I N E S             *
*                                    *
*                                    *
*       ------------------------     *
*       (c) 1989  ADP  SOFTWARE      *
*       ------------------------     *
*                                    *
*                                    *
*************************************/


#include "exec/types.h"
#include "libraries/dosextens.h"
#include "stdio.h"

extern struct FileHandle *Open();

main(argc,argv)
int argc;
char **argv;
{
int i,nchr,k;
struct FileHandle *infh;
char buffin[BUFSIZ];

if (argc!=2) Exit(0);

infh  = Open(argv[1],MODE_OLDFILE);

if (infh == 0) Exit(0);
k=1;
printf("\n  1:");

while ((nchr=Read(infh,buffin,BUFSIZ))!=0)
   {
   for (i=0; i<nchr; i++)
      {
      printf("%c",buffin[i]);
      if (buffin[i] == 0x0a)
                printf("%3d:",++k);
      }
   }
Close(infh);
}
```

Il primo parametro «v» oppure «h» (significato del «v|h») indica appunto se desideriamo un taglio verticale o orizzontale. Segue il nome del file da tagliare e un nome da dare al nuovo file «tagliato». Infine start e stop sono due interi che delimitano la porzione da conservare. Ovviamente sorgente e destinazione possono anche essere lo stesso file, ma attenzione che in questo modo se commettete un errore non potrete più tornare indietro.

Facciamo un esempio. Immaginiamo di avere un file di testo - formato da 143 linee e decidiamo di togliere le prime e le ultime venti di queste. Il file si chiama, tanto per cambiare, pippo e al nuovo file daremo nome «pippo 2». Il comando sarà:

```
cut h pippo pippo 2 21 123
```

Ovvero effettuiamo un taglio orizzontale (h) sul file pippo ottenendo il file pippo 2 che contiene le linee dalla 21 alla 123 del file d'origine. Ovviamente tanto il parametro «start» quanto il parametro «stop» possono coincidere con la prima o l'ultima linea (colonna) del file sorgente, in modo da effettuare un taglio unilaterale.

Per facilitare l'operazione con file di testo dove non si conoscono a priori le posizioni (numeriche) delle linee, ho approntato una «utility dell'utility», di nome Lines, che permette appunto la visualizzazione numerata delle linee del file sorgente prima di dare di piglio alle forbici (programma Cut). Anche questa, come gli altri due programmi, è scritta in linguaggio C ed è stata compilata col Lattice C 4.0 «liscio» ovvero un banale LC -L dei file sorgenti. Per quanto riguarda invece i tagli verticali, come già detto, è necessario che il file sorgente abbia una forma tabellare, ovvero formato da righe di uguale lunghezza (anche queste terminate da un NewLine) o perlomeno di lunghezza tutte maggiori del taglio da effettuare. La sintassi è la stessa del caso precedente, con la sola banale differenza del primo parametro «v» in lugo di «h».

Usiamo la colla

Il secondo programma, come detto, serve per «incollare». Attenzione: non in senso longitudinale (questa funzione è già offerta dal comando di sistema Join) ma in senso «trasversale». Nel senso che, prese due tabelle anche non di pari lunghezza, il file ottenuto sarà formato dalla prima tabella «affiancata»

```
/*************************************
*                                    *
*                                    *
*          C U T - F I L E           *
*                                    *
*                                    *
*       ------------------------     *
*       (c) 1989  ADP  SOFTWARE      *
*       ------------------------     *
*                                    *
*************************************/


#include "exec/types.h"
#include "exec/memory.h"
#include "libraries/dosextens.h"
#include "stdio.h"

void                        Abort(char *, int);
void                        swap(int *,int *);
long                        len;
extern struct FileHandle   *Open();
char                       *sorgente;
struct FileHandle          *infh,*outfh;

VOID main(argc,argv)
int argc;
char **argv;
{

int                     i,j=0,k=0,nchr,start,stop;
char                    buffin[BUFSIZ],buffout[BUFSIZ];
struct FileInfoBlock   *info;
struct FileLock        *lock;

if (argc!=6 || argv[1][0] == '?' ||(argv[1][0] != 'h' && argv[1][0] != 'v'))
Abort("Uso: Cut v|h Source Dest start stop\n\n** (c) 1989  ADP SOFTWARE **",0);

info = (struct FileInfoBlock *)AllocMem(
                sizeof(struct FileInfoBlock),MEMF_CLEAR);

lock  = (struct FileLock *)Lock(argv[2],ACCESS_READ);
Examine(lock,info);

len = info->fib_Size;

FreeMem(info,sizeof(struct FileInfoBlock));

UnLock(lock);


infh  = Open(argv[2],MODE_OLDFILE);
if (infh == 0) Abort("Source File Not Found",0);

sorgente = (char *)AllocMem(len,MEMF_CLEAR);
if (sorgente == 0) Abort("AmigaDOS Memory Fault... Sorry.",1);

while ((nchr=Read(infh,buffin,BUFSIZ))!=0)
        for (i=0; i<nchr; i++) sorgente[j++] = buffin[i];
Close(infh);

outfh = Open(argv[3],MODE_NEWFILE);
if (outfh == 0) Abort("Dest File Error",2);

start = atoi(argv[4])-1;
stop  = atoi(argv[5])-1;
nchr  = 0;

if (start > stop) swap(&start,&stop);

if (argv[1][0]=='v') for (i=0;i<j;i++)
   if (k<start) k++;
   else if (k<=stop)
             {
             buffout[nchr++]=sorgente[i];
             if (nchr==BUFSIZ-1)
                 {
                 Write(outfh,buffout,nchr);
                 nchr=0;
                 }
             k++;
             }
        else {
             buffout[nchr++]=0x0a;
             Write(outfh,buffout,nchr);
             nchr=k=0;
             while (sorgente[i] != 0x0a) i++;
             }
else {
     k=-1;
     i=0;
     while (++k<start)
        while (sorgente[i++] != 0x0a);
     while (k++<=stop)
           do {
              buffout[nchr++]=sorgente[i];
              if (nchr==BUFSIZ-1)
                 {
                 Write(outfh,buffout,nchr);
                 nchr=0;
                 }
              } while (sorgente[i++] != 0x0a);
     if (nchr!=0) Write(outfh,buffout,nchr);
     }
Close(outfh);
FreeMem(sorgente,len);
printf("\n");

} /* End of Program */

void swap(var1,var2)
int *var1,*var2;
{
int *temp;
temp = var1;
var1 = var2;
var2 = temp;
}

void Abort(ErrorString,QuitCode)
char ErrorString[];
int  QuitCode;
{
printf("\n\nCut_Error: %s\n\n",ErrorString);
if (QuitCode == 2) FreeMem(sorgente,len);
if (QuitCode == 1) Close(infh);
Exit(QuitCode);
}
```

dalla seconda. Ovvero ogni riga del file destinazione sarà formata da una riga del primo file sorgente e una riga del secondo e così via, fino a quando non abbiamo appaiato tutte le righe. Se un file «finisce prima» ovvero contiene un numero inferiore di righe non provoca alcun problema particolare: il programma infatti continua copiando le rimanenti linee del file più lungo.

In pratica questo comando Paste permette di saldare assieme varie fette di file ottenute dal Cut verticale, per ottenere nuovi file con tutti i campi che desideriamo escludendo quelli ai quali non siamo interessati. La sintassi è quanto mai semplice:

Paste File1 File2 Dest

dove File1 e File2 sono i file da incollare e Dest è il nome che daremo al file creato.

Note tecniche

La struttura del programma Cut è simile a quella del programma Replace presentato sul numero scorso. Per lavorare velocemente il file indicato come sorgente (ovvero quello da «tagliare») è dapprima caricato in memoria allocando uno opportuno spazio attraverso la funzione AllocMem. Per conoscere la quantità di spazio atta a contenere in memoria il file, facciamo uso della funzione Examine, una volta eseguita la Loc sul file in questione. Di tutto questo è stato ampiamente trattato nella rubrica «Programmare in C su Amiga» di Dario de Judicibus, alla quale vi rimandiamo per maggiore conoscenza. Ovviamente il programma Cut è diviso in due parti, che sono eseguite (in maniera mutuamente esIcusiva) a seconda che intendiamo effettuare un taglio verticale o orizzontale. Le eventuali eccezioni sono manipolate dalla già utilizzata funzione Abort che, prima di far terminare anzitempo il programma, chiude e libera tutto ciò che è rimasto aperto o occupato (file e memoria).

Il programma Paste, di contro, esegue le sue operazioni «al volo», senza cioè caricare in memoria i file da incollare. Mantiene ovviamente aperti contemporaneamente in lettura i file sorgenti e man mano che legge dall'uno o dall'altro file, scarica nel file destinazione. Non vi nascondo che questo «ballo dei due file» mi ha fatto tribolare non poco dovendo tenere sotto controllo contemporaneamente ben due buffer di ingresso, un buffer d'uscita e il flusso dei dati veri e propri. Adesso «pare» che funzioni. Scherzo, naturalmente...

```
/************************************
*                                   *
*                                   *
*        P A S T E - F I L E        *
*                                   *
*                                   *
*       ----------------------      *
*       (c) 1989  ADP  SOFTWARE     *
*       ----------------------      *
*                                   *
************************************/


#include "exec/types.h"
#include "exec/memory.h"
#include "libraries/dosextens.h"
#include "stdio.h"

void                          Abort(char *, int);
extern struct FileHandle     *Open();
char                         *sorgente;
struct FileHandle            *infh1,*infh2,*outfh;

VOID main(argc,argv)
int argc;
char **argv;
{

int                     i=0,j=0,k=0,nchr1,nchr2;
char                    buffin1[BUFSIZ],buffin2[BUFSIZ],buffout[BUFSIZ];

if (argc!=4 || argv[1][0] == '?' )
Abort("Uso: Paste Source1 Source2 Dest\n\n** (c) 1989  ADP SOFTWARE **",0);


infh1  = Open(argv[1],MODE_OLDFILE);
if (infh1 == 0) Abort("Source1 File Not Found",0);
infh2  = Open(argv[2],MODE_OLDFILE);
if (infh2 == 0) Abort("Source2 File Not Found",1);


outfh = Open(argv[3],MODE_NEWFILE);
if (outfh == 0) Abort("Dest File Error",2);

nchr1 = Read(infh1,buffin1,BUFSIZ);
nchr2 = Read(infh2,buffin2,BUFSIZ);

while (nchr1 || nchr2)
{
while ((nchr1) && buffin1[j]!=0x0a)
            {
            if (j>=nchr1)
               {
               nchr1 = Read(infh1,buffin1,BUFSIZ);
               j=0;
               }
            if (nchr1!=0) buffout[i++] = buffin1[j++];
            if (i==BUFSIZ)
               {
               Write(outfh,buffout,i);
               i=0;
               }
            }
while ((nchr2) && buffin2[k]!=0x0a)
            {
            if (k>=nchr2)
               {
               nchr2 = Read(infh2,buffin2,BUFSIZ);
               k=0;
               }
            if (nchr2!=0) buffout[i++] = buffin2[k++];
            if (i==BUFSIZ)
               {
               Write(outfh,buffout,i);
               i=0;
               }
            }
j++;k++;
if (nchr1 || nchr2) buffout[i++]='\n';
Write(outfh,buffout,i);
i=0;
}
Close(infh1);
Close(infh2);
Close(outfh);
printf("\n");

} /* End of Program */

void Abort(ErrorString,QuitCode)
char ErrorString[];
int  QuitCode;
{
printf("\n\nPaste_Error: %s\n\n",ErrorString);
if (QuitCode == 2) Close(infh2);
if (QuitCode >= 1) Close(infh1);
Exit(QuitCode);
}
```

POSTAL COMPUTER

PC XT IBM COMPATIBILE
L. 750.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA CGA O HERCULES, 256K ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI

PC XT IBM COMPATIBILE
L. 1.200.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA GRAFICA HERCULUS O CGA, 1 HARD DISK 20 MEGA, 256 ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI.

PC PHILIPS 9111
768K 1 DRIVE 5 1/4'' e 1 DRIVE 3 1/2''
L. 1.200.000

MANNESMANN MT 81
L. 290.000

PC AT IBM COMPATIBILE
L. 1.890.000
SCHEDA MADRE 80286, 12 MHZ, O WAIT, 512K ESPANDIBILE A 1024K, 1 DRIVE 5,25'' DA 1.2 MB 1 HARD DISK DA 20 MB SCHEDA HERCULES O CGA TASTIERA AVANZATA 101 TASTI.

TELEFAX MURATA M-1
L. 1.300.000

- COMPATIBILITÀ: G2 G3
- VELOCITÀ DI TRASMISSIONE 15 SECONDI
- APPARECCHIO TELEFONICO A TASTIERA INCORPORATO
- FOTOCOPIATORE
- RICEZIONE AUTOMATICA
- ROTOLO CARTA TERMICA 216 mm x 30 metri.
- OROLOGIO/CALENDARIO DIGITALE

HARD DISK SEAGATE 20 MB	L. 350.000
HARD DISK SEAGATE 40 MB	L. 660.000
HARD DISK CONTROLLER PER XT	L. 100.000
HARD DISK CONTROLLER PER AT	L. 220.000
SCHEDA GRAFICA E.G.A.	L. 300.000
SCHEDA VGA	L. 430.000
SCHEDA SERIALE	L. 40.000
SCHEDA PARALLELA	L. 35.000
SCHEDA PORTA JOYSTICK	L. 28.000
SCHEDA MADRE XT	L. 140.000
SCHEDA MADRE AT (6 MHZ 0 WAIT)	L. 450.000
TASTIERA AVANZATA 101 TASTI	L. 110.000
DRIVE 5,25 360KB	L. 110.000
DRIVE 5,25 1,2MB	L. 170.000
DRIVE 3,50 720KB	L. 150.000
DRIVE CONTROLLER	L. 49.000
CAVO PARALLELO	L. 15.000
DATA SWITCH A 2 PORTE	L. 60.000
MOUSE ANKO	L. 59.000
JOYSTICK I.B.M. ANKO	L. 45.000

STAMPANTI CITIZEN GRAFICA - NLQ

CITIZEN 120 D L. 335.000 120 CPS, SET. EPSON IBM 80 COL. TRATO IN TRAZIONE, FRIZIONE INTER. OPZIONALE IBM/COMMODORE	**CITIZEN MSP 50** L. 950.000 250/300 CAR/SEC., 80 COL.
CITIZEN LSP 100 L. 550.000 -160 cps, 80 COL.	**CITIZEN MSP 55** L. 1.040.000 250/300 CAR/SEC, 136 COL.
CITIZEN MSP 10E L. 650.000 - 160 CAR/SEC, 80 COL.	**CITIZEN HQP 40** L. 920.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 15E L. 539.000 160 CAR/SEC, 136 COL.	**CITIZEN HQP 45** L. 1.350.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 40 L.610.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN 180E** COMPLETA DI INTERFACCIA IBM O COMMODORE - L.340.000
CITIZEN MSP 45 L. 750.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN OVERTURE 110** L.3.600.000 - STAMPANTE LASER

TUTTI I PRODOTTI CITIZEN SONO COPERTI DA CERTIFICATO DI GARANZIA DELLA VALIDITÀ DI DUE ANNI

OFFERTA MONITOR

PHILIPS			**Segue PHILIPS**		
MONITOR 8875 14'' MULTISINK	L. 935.000	colore	MONITOR 7749 14'' TTL compatibile IBM sist. 2	L. 210.000	F/B
MONITOR 8833 14'' CGA	L. 450.000	colore	MONITOR 7513 12'' TTL	L. 136.000	F/V
MONITOR 8802 14'' COLORI	L. 360.000	colore	MONITOR 7713 14'' TTL	L. 183.000	
MONITOR 9043 14'' EGA	L. 535.000	colore	**ANTAREX**		
MONITOR 9053 14'' EGA	L. 595.000	colore	BOXER 14'' P39 JAN DUAL	L. 190.000	F/V o F/B
MONITOR 9073 14'' EGA	L. 680.000	colore	BIM 12'' PC DM 216B	L. 135.000	F/V
MONITOR 7723 14'' TTL	L. 192.000	F/A	CT 9000 SHR EGA JAN	L. 670.000	colore
MONITOR 7743 14'' TTL	L. 205.000	F/B	CT 9000/L MR14 DIM 414	L. 430.000	colore
MONITOR 9082 14'' VGA	L. 700.000	colore			

PREZZI SU RICHIESTA

GARANZIA 12 MESI

PREZZI IVA ESCLUSA
SPESE DI SPEDIZIONE ESCLUSE

TEL. 06/3651688

TELEFONATECI