Sulla mia scrivania continuano ad arrivare montagne di programmi inutilizzabili per la rubrica Sotfware Amiga. Pensavo di essere stato fin troppo chiaro a riguardo, ma, tanto per vie telematiche, quanto per telefono, quanto «a quattr'occhi» proprio in questi giorni allo SMAU, continuo a ricevere domande del tipo: come fare per pubblicare un programma, quali accorgimenti prendere, quanto deve essere lungo un articolo, quanto il programma, ecc. ecc.
C'è finanche chi mi ha scritto «... so che questo programma è troppo lungo, e pur contravvenendo alle tue indicazioni ho deciso di mandarlo ugualmente, non si sa mai...». Segue articolo di venti pagine accompagnato da listato di 6 metri. Come non si sa mai? Ma che stiamo scherzando? Un altro lettore mi ha confessato che il suo programma è sì lunghissimo, ma si tratta di una utility, quindi perfettamente pubblicabile. Secondo lui...
Vogliamo, per favore, intenderci una volta per tutte che il problema è puramente metrico? Siamo d'accordo tutti sul fatto che un articolo che parla di un programma non pubblicato è utile solo a quelle quattro-cinque persone (credeteci, non di più) che acquisteranno il dischetto presso la redazione?
La bontà di un elaborato (di qualsiasi tipo) non si misura certo col metro, ovvio, però nel caso della rubrica Software Amiga è semplicemente limitato superiormente per dimensioni. Il programma che segue è un esempio «concreto» di ottimo elaborato di dimensioni «ragionevoli». Prima di fare il vostro prossimo pacchetto, abbiate cura di usare il giusto metro, tutto qui. E grazie per la collaborazione...

adp

MapMem

di Michele Iacobellis - Bari

Chi, come me, è passato dal C64 all'Amiga conosce bene le enormi differenze di programmazione tra le due macchine. Mentre, con il 64, il programmatore conosce benissimo le posizioni in memoria di ogni cosa (programmi Basic, registri di controllo dell'Hardware, locazioni delle routine del Sistema Operativo) nell'Amiga la situazione è drammaticamente opposta. Il programma presentato è un Tool del Workbench che permette di avere una visione immediata e globale dello stato di occupazione della memoria, utile sia per sapere in tempo reale la quantità e qualità di memoria disponibile, sia per osservare il modo di funzionamento di programmi che allocano dinamicamente la memoria (compilatori ecc.).

Qualche nozione tecnica

Ormai lo sanno anche le pietre che nell'Amiga esiste una sola locazione assoluta (il puntatore ad Exec in 0x000004), mentre tutto il resto viene gestito in maniera totalmente dinamica. La memoria non fa eccezione a questa regola generale. Quindi se vogliamo un po' di memoria per i nostri smanettamenti, non possiamo prendercela e basta, ma la dobbiamo umilmente chiedere al S.O. che ce la concederà in uso, per poi doverla restituire, pena la perdita irrimediabile del contenuto della memoria stessa. Allo stesso tempo è di importanza vitale non sconfinare in nessun modo in zone di memoria a noi non riservate.

Per realizzare questo flessibilissimo modo di gestire la memoria, come per altro avviene per tutte le risorse condivisibili, Exec (il modulo più importante del S.O. dell'Amiga) usa la struttura dati più flessibile che ci sia: la lista. Se non vi ricordate cosa è una lista, andate a riguardare «Appunti di informatica» su MC numero 82. In una macchina inespansa (A500 con 512K di RAM) esiste un solo tipo di memoria, la cosiddetta «Chip Ram», mentre nelle macchine con un MegaByte e oltre, è presente anche la «Fast Ram» che si differenzia dalla prima per il fatto che non è visibile ai coprocessori, e quindi è a completa disposizione del solo 68000. Per ognuno di questi due tipi di memoria (che per comodità chiamerò Blocchi), esiste una lista che ne descrive i «pezzi» (Chunks in amighese) liberi. Ogni nodo della lista contiene due informazioni: indirizzo del prossimo nodo e lunghezza del Chunk in byte. È importante notare che il nodo (che ha una dimensione di 8 byte) si trova nelle prime 8 locazioni del Chunk che descrive. Per questo motivo ogni Chunk è lungo almeno 8 byte, e quindi la memoria minima allocabile è di

```
/* Questa è la definizione del Chunk, ovvero del nodo della lista della memoria
libera */
struct MemChunk {
    struct MemChunk *mc_Next;  /* Prossimo Chunk */
    LONG mc_Bytes;             /* Lunghezza del Chunk */
};

/* Questa è la testa della lista della memoria libera */
struct MemHeader {
    struct Node mh_Node;       /* Nodo della lista dei Blocchi di memoria */
    UWORD mh_Attributes;        /* Tipo della memoria */
    struct MemChunk *mh_First;   /* Primo Chunk della lista */
    APTR   mh_Lower;            /* Inizio di questa memoria */
    APTR   mh_Upper;            /* Fine di questa memoria */
    ULONG  mh_Free;             /* Bytes liberi totali */
};
```

Figura 1 - Definizioni tratte dall'include file di sistema "exec/memory.h".

8 byte (a fare gli opportuni arrotondamenti ci pensa Exec).

Ogni lista di memoria libera ha una testa (Header) che ne descrive le caratteristiche: tipo della memoria (Chip o Fast), locazioni di inizio e fine, byte liberi totali, ecc. In figura 1 sono riportate le definizioni sia dell'Header che del Chunk. I vari Header fanno a loro volta parte di una lista circolare doppia, la cui testa si trova nella struttura ExecBase, che è il cuore di Exec.

Quindi, per scandire la lista della memoria libera, è sufficiente leggere da ExecBase la testa della lista dei Blocchi, percorrere la lista dei Blocchi, e per ogni Blocco scandire la lista dei Chunks liberi in quel blocco. Quest'ultima operazione bisogna effettuarla con cautela. Infatti, poiché Exec è un Sistema Operativo multitasking, potrebbe accadere che mentre stiamo leggendo la lista della memoria libera, un altro task allochi memoria, modificando la lista e rendendo privo di significato ciò che stiamo leggendo (con una inevitabile Guru Meditation). Per ovviare a questo spiacevole inconveniente, bisogna, prima di accedere alla lista, disabilitare il multitasking. Ciò è possibile mediante due funzioni di Exec: Forbid() e Permit(). Forbid() disabilita il multitasking, lasciando inalterati gli interrupt del sistema, rendendo attivo solo il nostro task; Permit() lo riabilita. Ma quando il multitasking è disattivo, tutto l'input/output della macchina (mouse, tastiera ecc.) è congelato. Quindi per ovviare a rallentamenti del movimento del mouse, che farebbero degradare le prestazioni di Intuition la disabilitazione del multitasking deve durare il meno possibile.

Il programma

MapMem può essere lanciato sia da Workbench, che da CLI. Il programma, che riconosce automaticamente la configurazione a 512K o ad un Mega della macchina, accetta un solo parametro: l'intervallo di tempo (in cinquantesimi di secondo) tra una scansione e l'altra della memoria. Al lancio, appare nella parte superiore del WB una finestra a sviluppo orizzontale con due barre (o una per i 512K) che sono colorate a bande verticali. Ogni banda indica un Chunk di memoria libera o occupata a seconda del colore, come si può vedere dalla legenda in basso. Il tutto completato da una scala graduata che aiuta a localizzare il Chunk nella memoria e dall'indicazione dei KiloByte liberi sull'estrema destra. La finestra può essere spostata sullo schermo e anche in profondità, mentre per far terminare il programma basta clickare sul gadget di chiusura della finestra. Alla luce di quanto detto in precedenza, il listato è veramente molto semplice ed è corredato da numerosi commenti esplicativi. Un'ultima annotazione sulla compilazione: Compilatore: Manx Aztec C v.3.6a Compilazione: cc +l MapMem.c Linking: ln MapMem.o –lc132 (c132.lib è la libreria standard a 32 bit dell'Aztec). In ogni caso non ci dovrebbero essere problemi con altri compilatori. MC

1

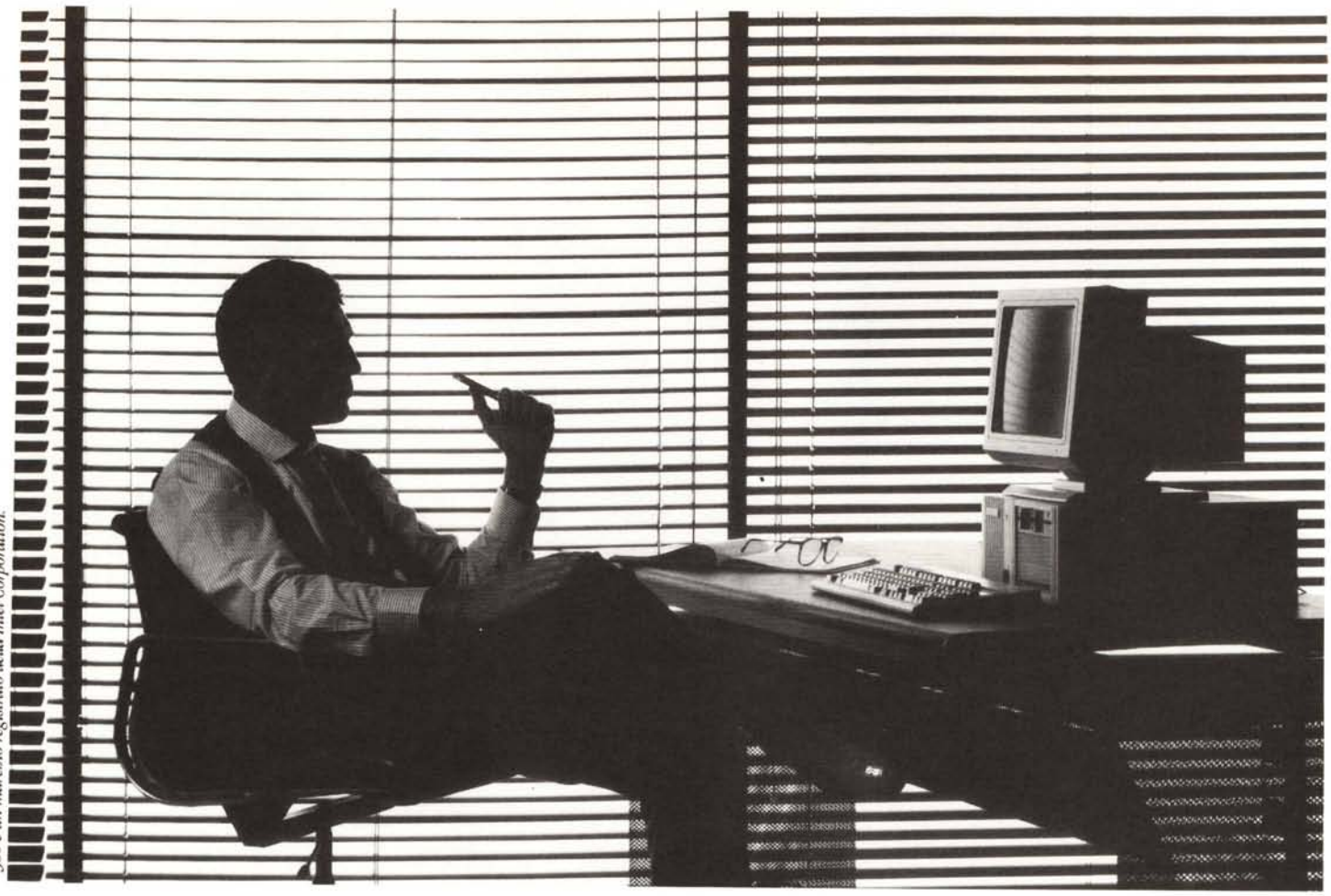
```
/*****************************************************************************
*                                                                            *
*                  MapMem v.1.0 - di Michele Iacobellis                      *
*                                                                            *
*                                                                            *
*****************************************************************************/

#include "exec/types.h"
#include "exec/execbase.h"
#include "exec/memory.h"
#include "intuition/intuition.h"

#define MAXCHUNKS 100          /* Numero massimo di Chunks gestibili */
#define FINE_LISTA -1L         /* Marcatore per la fine della lista */

/* Definizioni per la gestione di Intuition */
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Window *Window;
struct RastPort *RP;
struct NewWindow NewWindow =
 {0,10,610,54,0,1,CLOSEWINDOW|WINDOWDRAG|WINDOWDEPTH,NULL,NULL,"MapM
em v.1.0 - JAC Soft",NULL,NULL,0,0,0,0,WBENCHSCREEN};
struct IntuiMessage *Mess;

/* Definizioni per leggere le liste */
struct ExecBase *EB;
struct MemHeader *MH;

/* Dati del programma */
ULONG ChipList[2*MAXCHUNKS];    /* Qui ricopia la lista della Chip Ram */
ULONG FastList[2*MAXCHUNKS];    /* Qui quella della Fast Ram */
UWORD Tipo, MemFree, Frequenza;

/* Questa funzione scorre la lista di memoria di Header MH e la confronta con
la lista salvata in precedenza, resituendo TRUE se sono uguali. In ogni caso
alla fine in OldList c'e' la lista attuale della memoria. */
BOOL
ReadFreeList (MH, OldList)
struct MemHeader *MH;           /* Header della lista da esaminare */
ULONG *oldList;                 /* Lista con cui confrontarla */
{
BOOL Uguale;                    /* La lista della memoria e' cambiata? */
ULONG ThisChunk;                /* Bytes contenuti nel Chunk attuale */
SHORT ChunkNumber;              /* Numero d'ordine del Chunk attuale */
struct MemChunk *MC;            /* Puntatore al Chunk attuale */

Uguale = TRUE;                  /* All'inizio le liste sono sempre uguali */
ChunkNumber = 0;
Forbid();                       /* Disabilita il Multitasking */
MC = MH->mh_First;
while ((MC != NULL) && (ChunkNumber < MAXCHUNKS))
    {
     ThisChunk = MC->mc_Bytes;
     if ((((ULONG)MC != *oldList) || (ThisChunk != *(oldList+1)))
          Uguale = FALSE;
     if (! Uguale)
          {
           *oldList = (ULONG)MC;
           *(oldList+1) = ThisChunk;
          }
     oldList += 2;
     ChunkNumber++;
     MC = MC->mc_Next;
    }
Permit();
if (ChunkNumber < MAXCHUNKS)    /* Riabilita il Multitasking */
     *oldList = FINE_LISTA;     /* Marca la fine della lista se */
return (Uguale);                /* non e' del tutto piena */
}
```

2

```
main (argc, argv)
int argc;
char **argv;
{
/* Se non c'e' alcun parametro, si deve usare quello di default */
Frequenza = (argc > 1) ? atoi(argv[1]) : 10;

/* Apertura delle librerie e della finestra */
IntuitionBase = (struct IntuitionBase *)OpenLibrary("intuition.library",0);
if (IntuitionBase == NULL) Esci();
GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0);
if (GfxBase==NULL) Esci();
if ((Window = (struct Window *)OpenWindow(&NewWindow))==NULL) Esci();
RP = Window->RPort;       /* RastPort per tutte le operazioni grafiche */

/* Layout grafico della finestra */
SetAPen (RP, 2);
RectFill (RP, 42, 43, 50, 50);
SetAPen (RP, 1);
RectFill (RP, 220, 43, 228, 50);
Move (RP, 2, 17);   Text (RP, "Tipo", 4);
Move (RP, 2, 30);   Text (RP, "CHIP", 4);
Move (RP, 2, 40);   Text (RP, "FAST", 4);
Move (RP, 58, 50);  Text (RP, "Memoria Occupata", 16);
Move (RP, 236, 50); Text (RP, "Memoria Libera", 14);
Move (RP, 40, 20);  Draw (RP, 551, 20);
Move (RP, 40, 19);  Draw (RP, 40, 21);
Move (RP, 40, 17);  Text (RP, "0Kb", 3);
Move (RP, 551, 19); Draw (RP, 551, 21);
Move (RP, 531, 17); Text (RP, "512Kb",5);
Move (RP, 168, 19); Draw (RP, 168, 21);
Move (RP, 148, 17); Text (RP, "128Kb", 5);
Move (RP, 296, 19); Draw (RP, 296, 21);
Move (RP, 276, 17); Text (RP, "256Kb", 5);
Move (RP, 424, 19); Draw (RP, 424, 21);
Move (RP, 404, 17); Text (RP, "384Kb", 5);

/* ciclo principale infinito (a meno di un click sul CLOSEGADGET) */
for (;;)
        {
        EB = *(ULONG *)0x000004;              /* Recupera ExecBase */
        MH = EB->MemList.lh_Head;             /* Prende la testa della lista */
        Tipo = MH->mh_Attributes;             /* Tipo di questa memoria */
        MemFree = (UWORD)(MH->mh_Free >> 10); /* Bytes liberi */
        if ((Tipo & MEMF_CHIP) && (! ReadFreeList (MH, ChipList)))
                RefreshDisplay (Tipo, ChipList, MemFree);
        if ((Tipo & MEMF_FAST) && (! ReadFreeList (MH, FastList)))
                RefreshDisplay (Tipo, FastList, MemFree);
        MH = MH->mh_Node.ln_Succ;     /* Prossimo Blocco di memoria */
        if (MH->mh_Node.ln_Succ)      /* Da esaminare solo se esiste (1 Mb) */
                {                     /* Ripete le operazioni precedenti */
                Tipo = MH->mh_Attributes;
                MemFree = (UWORD)(MH->mh_Free >> 10);
                if ((Tipo & MEMF_CHIP) && (! ReadFreeList (MH, ChipList)))
                        RefreshDisplay (Tipo, ChipList, MemFree);
                if ((Tipo & MEMF_FAST) && (! ReadFreeList (MH, FastList)))
                        RefreshDisplay (Tipo, FastList, MemFree);
                }
        Delay (Frequenza);            /* Aspettiamo un po' prima di ripetere */
        if (Mess = GetMsg (Window->UserPort))  /* se c'e' un click si esce */
                break;
        }
ReplyMsg(Mess);       /* Risponde per bene al messaggio di Intuition */
Esci();               /* Chiude tutto prima di terminare */
}

/* Disegna nella finestra la situazione della Lista */
RefreshDisplay (Tipo, Lista, MemFree)
UWORD Tipo, MemFree;   /* Tipo e Bytes liberi di questo Blocco */
```

3

```
ULONG *Lista;          /* Lista descrittiva del Blocco */
{
SHORT Inizio, Fine, Count, LimSup, LimInf;
char str[5];

if (Tipo & MEMF_CHIP)          /* Seleziona la barra della Chip Ram */
        {
        LimSup = 23;
        LimInf = 31;
        Move (RP, 560, 30);
        }
else                           /* Altrimenti quella della Fast Ram */
        {
        LimSup = 33;
        LimInf = 41;
        Move (RP, 560, 40);
        }
SetAPen (RP, 1);               /* Scrive a destra il numero dei Kb liberi */
Byte2String(MemFree, &str);
Text (RP, &str, 5);

/* Scorre la lista di questo blocco disegnando una banda di colore 2 per
la memoria libera e di colore 1 per quella occupata */
Fine = 39;
for (Count = 0; Count < MAXCHUNKS; Count++)
        {
        if (*Lista == FINE_LISTA)
                break;
        Inizio = ((*(Lista++) & 0x07ffff) >> 10) + 40;
        if (Inizio > Fine)
                {
                SetAPen (RP, 2);
                RectFill (RP, Fine+1, LimSup, Inizio, LimInf);
                }
        Fine = Inizio + (*(Lista++) >> 10) ;
        SetAPen (RP, 1);
        RectFill (RP, Inizio, LimSup, Fine, LimInf);
        }
if (Fine < 551)
        {
        SetAPen (RP,2);
        RectFill (RP, Fine+1, LimSup, 551, LimInf);
        }
}

/* Trasforma Num (< 512 ) in una stringa di 5 caratteri con il formato ###Kb*/
Byte2String (Num, str)
UWORD Num;
char *str;
{
*(str++) = Num / 100 + '0';
Num %= 100;
*(str++) = Num / 10 + '0';
*(str++) = Num % 10 + '0';
*(str++) = 'K';
*(str++) = 'b';
}

/* Chiude solo cio' che era stato aperto ed esce */
Esci ()
{
if (Window != NULL) CloseWindow(Window);

if (GfxBase != NULL) CloseLibrary (GfxBase);

if (IntuitionBase != NULL) CloseLibrary (IntuitionBase);

exit (0);
}
```



LEO BURNETT

* 386 è un marchio registrato della Intel Corporation.

Prima di dare il posto a un computer leggete il suo curriculum.

1975. Quando Jugi Tandon arriva in California, a Silicon Valley, è il boom. La sua azienda comincia dalle testine di lettura e scrittura per i disk drive. In soli due anni è leader con l'80% del mercato.

1979. Dalla Tandon escono i primi drive completi per floppy disk. Già dopo un anno è leader nel nuovo settore.

1985. Nasce la linea di PC Tandon. Caratteristiche chiave, la compatibilità e l'ottimo rapporto prezzo/prestazioni.

1986. In soli tre paesi europei, Tandon vende 55.000 unità in un anno. Un record mai raggiunto.

1987. Il Personal Data Pac Tandon è il primo hard disk estraibile. Per il computer significa memoria illimitata, portatilità e sicurezza dati. In 4 paesi europei il PAC 286 Tandon è il "Personal Computer of the Year".

1988. Tandon è ormai fra i primi 5 produttori di PC compatibili standard in Europa, con 137.701 sistemi venduti. Mentre si avvia un centro di produzione europeo in Austria, apre la filiale italiana.

1989. 8 marzo: al CEBIT di Hannover, Tandon annuncia il primo personal 386*/33 Mhz, il più potente al mondo.

5/9 ottobre: è allo SMAU con una gamma completa di PC professionali, dall'XT ai 386, e novità ricche di interesse. Per le aziende che, come noi in questi 15 anni, hanno un solo obiettivo: essere i più competitivi.

Tandon

USA TECHNOLOGY MADE IN EUROPE

TANDON COMPUTER S.p.A. - Via E. Fermi 20 - 20094 Assago (MI) - Tel. 02/4883352