www.asap-firmware.com

Prefazione

Input e Output nel linguaggio C
Softune Workbench C Compiler
La soluzione ASAP
Esempi in linguaggio C

Titolo: FWS0004A

Stream Out

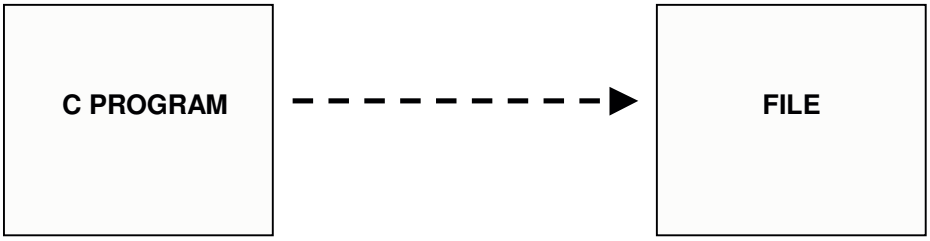
Input e Output nel linguaggio C

L'operazione di Input/Output su file è uno degli aspetti più delicati di un linguaggio di programmazione poiché è strettamente collegato al sistema operativo, in quanto può essere diverso il modo con il quale questo permette l'accesso ai dati nei file o nei dispositivi.

Queste variazioni rendono difficile progettare funzioni di Input/Output (I/O) che siano portabili su diverse implementazioni di un linguaggio di programmazione.

Il linguaggio C compie I/O attraverso un ampio set di funzioni di libreria.

Non viene fatta distinzione tra i dispositivi (terminale, periferica, file su disco) e in tutti i casi la funzione di I/O viene eseguita attraverso una ordinata serie di bytes (stream) associata con il file o il dispositivo in oggetto.



Esistono tre standard streams che vengono aperte automaticamente e cioè *stdin*, *stdout* e *stderr*; in genere queste sono associate al terminale video ma è possibile re-indirizzarle ad esempio su un file.

Il formato dei bytes nello stream può essere *testo* o *binario*: il formato testo consiste in una serie di linee terminate ognuna da un carattere newline; storicamente infatti i dispositivi (es. stampanti seriali) memorizzavano queste linee sincronizzandosi sul carattere newline; il formato binario è utilizzato per dati non testuali che non devono essere interpretati ma devono mantenere la stessa struttura.

Un altro aspetto da considerare è la modalità di trasmissione di queste streams.

In genere sono indirizzate verso dispositivi molto più lenti del tempo impiegato dalla CPU per compiere l'operazione; per ridurre il più possibile il numero di accessi di lettura/scrittura si utilizza un'area di memoria temporanea (buffer) in cui i dati vengono trasferiti prima di essere inviati alla loro destinazione finale.

I sistemi operativi utilizzano buffer di dimensione fissa (es. 512 o 1024 bytes) ed è quindi possibile agire su blocchi di dimensioni molto ampie con una sola operazione.

Softune Workbench C Compiler

Nella famiglia di microcontrollori Fujitsu F2MC-16LX la compilazione del programma in linguaggio C, avviene nell'ambiente di sviluppo integrato denominato Softune Workbench.

Le funzioni di stream sono implementate nella libreria *stdio,* richiamabili per mezzo delle funzioni elencate nel file di header <stdio.h>.

In questo caso si analizzerà la funzione di stream out *printf(),* utilizzata soprattutto in fase di debug; consente infatti di convertire il valore dei dati secondo una tabella di formattazione (es. decimale, Hex, floating point) e visualizzarli in un formato testo.

Nel mondo embedded in genere questa funzione viene re-indirizzata verso una porta seriale utilizzando una periferica UART interna al microcontrollore.

Durante lo svolgimento del programma è quindi possibile trasmettere queste stringhe di caratteri ASCII via RS232 ad un PC e visualizzarle sul video utilizzando appositi programmi (es. Hyperterminal).

E' comunque possibile decidere di indirizzare le stringhe verso altri dispositivi (es. un LCD alfanumerico), creando delle funzioni a basso livello (driver) che si interfacciano verso il dispositivo desiderato.

Nel manuale C Softune queste funzioni sono denominate LOW LEVEL FUNCTION e nel caso della *printf()* viene utilizzata la funzione "write()".

Per utilizzare la funzione *printf()* è necessario eseguire le seguenti operazioni:

- aggiungere i moduli C LOWLEVEL.C e SBRK.C
- impostare nel file START.ASM il valore di `CLIBINIT` su `ON`
- includere il file header <stdio.h> dove viene chiamata *printf*

La libreria standard chiama la funzione "sbrk()" per allocare il buffer (heap memory) per ogni stream, in ordine su stdio, stdout e stderr; la dimensione per ogni blocco viene passata come parametro di ingresso ed è di 512 bytes (totale quindi 1536 bytes); se la dimensione supera le dimensioni del buffer (heap) lo stream relativo non viene aperto.

La soluzione ASAP

La soluzione ASAP consente di utilizzare la libreria standard, riducendo in modo considerevole la quantità di memoria necessaria; per questo è stata modificata la funzione "sbrk()" in modo da allocare memoria solo per il buffer dello stream stdout.

Oltre a questo è stato studiato un altro metodo che sfrutta la funzione "sprintf()" e semplifica ulteriormente tale implementazione; "sprintf()" è identica alla "printf()" ma indirizza la stringa formattata su un buffer e non si appoggia su altre funzioni "driver".

Il risultato è la funzione "mystream_printf()", che usa gli stessi parametri di ingresso della "printf()" e formatta i dati verso il buffer di memoria relativo a stdout; per utilizzarla non serve né aggiungere i moduli C LOWLEVEL.C e SBRK.C, né impostare nel file START.ASM il valore di `CLIBINIT` su `ON`, ma solo includere i file di header relativi.

La funzionalità di stream out su seriale nelle librerie standard trasmette la stringa di stream su RS232 utilizzando una UART in modo sospensivo, attendendo per ogni dato la fine della trasmissione; questo si traduce nell'introduzione di ritardi di decine di millisecondi, che alterano in modo sensibile le tempistiche di esecuzione del flusso di programma.

Una ulteriore miglioria introdotta con questa soluzione consiste nel fatto che lo svuotamento del buffer verso il dispositivo non è sospensiva: la trasmissione avviene in interrupt e non in polling.

In caso di monitor di variabili di applicativo con tempi di refresh di millisecondi, è possibile inserire dove necessario le chiamate a "printf()" e "mystream_printf()" senza alterare le tempistiche del flusso di programma standard, avvicidandosi quindi alla condizione di reale funzionamento.

Esempi in linguaggio C

In questa soluzione è stato realizzato un progetto per microcontrollore Fujitsu MB90F497 con Softune C Worbench versione 3.4, contenuto nel file FWS0004A.ZIP.

Esso contiene :

UART497.C : modulo di gestione UART1

VECTORS.C : modulo gestione vettori di interrupt

MAIN.C : modulo con main loop

Modificando le funzioni di gestione UART(driver) è possibile utilizzare questa soluzione con altri microcontrollori della stessa famiglia.

La modularità con il quale è stato progettato il software consente il porting anche su microcontrollori di marche diverse.

Definizione dimensione buffer circolare relativo allo stream stdout :

```
#define _PRINTF_BUFF_SIZE 64
```

Definizione dimensione buffer di formattazione relativo a printf() e mystream_printf():

```
#define HEAP_SIZE_SMALL 32
```

Definizione valori del baudrate configurabili :

```
enum{_BPS_19200,_BPS_9600,_BPS_4800,_BPS_2400,_NO_BPS};
```

Funzione di allocazione della memoria heap ridotta solo per stream stdout :

```
static unsigned char NFile= STDIN;
char HeapStdOut[HEAP_SIZE_SMALL];
extern char *sbrk(int size)
{
  if (NFile++==STDOUT)
      return(HeapStdOut);
 else
     return((char*)-1);
}
```

Alcune note per l'uso corretto del software :

1. la dimensione massima della stringa su printf deve essere inferiore a `HEAP_SIZE_SMALL` bytes
2. chiamate in successione a printf() e mystream_printf() non devono saturare il buffer circolare, e questo dipende dal baudrate impostato e dalla lunghezza delle stringhe da trasmettere