Il file '`kernel/ibm_i86.h`' e quelli contenuti nella directory '`kernel/ibm_i86/`', raccolgono il codice del kernel che è legato strettamente all'hardware; a questi file vanno anche aggiunti '`lib/sys/os16.h`' e la directory '`lib/sys/os16/`', della libreria, utilizzati anche dalle applicazioni, a vario titolo. In generale si può osservare la presenza di funzioni che si avvalgono direttamente di alcune interruzioni del BIOS (fondamentalmente per la gestione del video e per l'accesso ai dischi); funzioni che permettono di leggere il valore di alcuni registri; funzioni per leggere e scrivere la memoria, in posizioni arbitrarie; funzioni per facilitare la lettura e la scrittura nei dischi, anche a livello di byte.

Salvo poche eccezioni, le funzioni scritte in linguaggio assemblatore hanno nomi che iniziano con un trattino basso, ma a fianco di queste sono anche disponibili delle macroistruzioni, con nomi equivalenti, senza il trattino basso iniziale, per garantire che gli argomenti della chiamata abbiano il tipo corretto, restituendo un valore intero «normale», quando qualcosa deve essere restituito.

Libreria: «lib/sys/os16.h» e «lib/sys/os16/...»

Listato u0.12 e successivi.

Nel file '`lib/sys/os16.h`' e in quelli della directory '`lib/sys/os16/`' si raccolgono, tra le altre, delle funzioni di basso livello che possono essere utili per il kernel e per le applicazioni. Si tratta di ***_seg_i()*** e ***_seg_d()*** (ovvero le macroistruzioni ***seg_i()*** e ***seg_d()***), con cui si ottiene, rispettivamente, il numero del segmento codice (istruzioni) e il numero del segmento dati. Inoltre, per poter verificare gli altri registri di segmento e i registri di gestione della pila, si aggiungono le funzioni ***_cs()***, ***_ds()***, ***_ss()***, ***_es()***, ***_sp()*** e ***_bp()***; le quali, rispettivamente, consentono di leggere il valore dei registri ***CS***, ***DS***, ***SS***, ***ES***, ***SP*** e ***BP*** (le macroistruzioni equivalenti sono ***cs()***, ***ds()***, ***ss()***, ***es()***, ***sp()*** e ***bp()***).

Tabella u144.1. Funzioni e macroistruzioni legate strettamente all'hardware, dichiarate nel file di intestazione '`lib/sys/os16.h`'. Tali funzioni e macroistruzioni possono essere utilizzate sia dal kernel, sia dalle applicazioni.

Funzione o macroistruzione	Descrizione
`uint16_t _seg_i (void);` `unsigned int seg_i (void);`	Restituisce il numero del segmento codice (*instruction*). Listati u0.12 e i161.12.6.
`uint16_t _seg_d (void);` `unsigned int seg_d (void);`	Restituisce il numero del segmento dati. Listati u0.12 e i161.12.5.
`uint16_t _cs (void);` `unsigned int cs (void);`	Restituisce il valore del registro ***CS***. Listati u0.12 e i161.12.2.
`uint16_t _ds (void);` `unsigned int ds (void);`	Restituisce il valore del registro ***DS***. Listati u0.12 e i161.12.3.
`uint16_t _ss (void);` `unsigned int ss (void);`	Restituisce il valore del registro ***SS***. Listati u0.12 e i161.12.8.
`uint16_t _es (void);` `unsigned int es (void);`	Restituisce il valore del registro ***ES***. Listati u0.12 e i161.12.4.
`uint16_t _sp (void);` `unsigned int sp (void);`	Restituisce il valore del registro ***SP***. Il valore che si ottiene si riferisce allo stato del registro, prima di chiamare la funzione. Listati u0.12 e i161.12.7.
`uint16_t _bp (void);` `unsigned int bp (void);`	Restituisce il valore del registro ***BP***. Il valore che si ottiene si riferisce allo stato del registro, prima di chiamare la funzione. Listati u0.12 e i161.12.1.

Funzioni di basso livello dei file «kernel/ibm_i86/*»

« Listato u0.5 e successivi.

Le funzioni con nomi che iniziano per '**`_int`*nn*`…()`**', dove ***nn*** è un numero di due cifre, in base sedici, consentono l'accesso all'interruzione ***nn*** del BIOS dal codice in linguaggio C.

Le funzioni con nomi del tipo '**`_in_`*n*`()`**' e '**`_out_`*n*`()`**' consentono di leggere e di scrivere un valore di ***n*** bit in una certa porta.

La funzione ***cli()*** disabilita le interruzioni hardware, mentre ***sti()*** le riabilita. Queste due funzioni vengono usate pochissimo nel codice del kernel. A loro si aggiungono le funzioni ***irq_on()*** e ***irq_off()***, per abilitare o escludere selettivamente un tipo di interruzione hardware. Queste funzioni vengono usate in una sola occasione, quando si predispone la tabella IVT e poi si abilitano esclusivamente le interruzioni utili.

La funzione ***_ram_copy()*** si occupa di copiare una quantità stabilita di byte da una posizione della memoria a un'altra, entrambe indicate con segmento e scostamento (la funzione ***mem_copy()*** elencata in '`kernel/memory.h`' si avvale in pratica di questa).

Per agevolare l'uso di queste funzioni, senza costringere a convertire i valori numerici, sono disponibili diverse macroistruzioni con nomi equivalenti, ma privi del trattino basso iniziale.

Tabella u144.2. Funzioni e macroistruzioni di basso livello, dichiarate nel file di intestazione '`kernel/ibm_i86.h`' e descritte nei file della directory '`kernel/ibm_i860/`'. Le macroistruzioni hanno argomenti di tipo numerico non precisato, purché in grado di rappresentare il valore necessario.

Funzione o macroistruzione	Descrizione
`void _int10_00 (uint16_t` ***video_mode***`);` `void int10_00 (`***video_mode***`);`	Imposta la modalità video della console. Questa funzione viene usata solo da ***con_init()***, per inizializzare la console; la modalità video è stabilita dalla macro-variabile ***IBM_I86_VIDEO_MODE***, dichiarata nel file '`kernel/ibm_i86.h`'.

`void _int10_02 (uint16_t` ***page***`,`
`uint16_t` ***position***`);`
`void int10_02 (`***page***`,` ***position***`);`
	Colloca il cursore in una posizione determinata dello schermo, relativo a una certa pagina video. Questa funzione viene usata solo da ***con_putc()***.

`void _int10_05 (uint16_t` ***page***`);` `void int10_05 (`***page***`);`	Seleziona la pagina attiva del video. Questa funzione viene usata solo da ***con_init()*** e ***con_select()***.
`void _int12 (void);` `void int12 (void);`	Restituisce la quantità di memoria disponibile, in multipli di 1024 byte.
`void _int13_00 (uint16_t` ***drive***`);` `void int13_00 (`***drive***`);`	Azzera lo stato dell'unità a disco indicata, rappresentata da un numero secondo le convenzioni del BIOS. Viene usata solo dalle funzioni '**`dsk_…()`**' che si occupano dell'accesso alle unità a disco.

`uint16_t _int13_02 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_02 (`***drive***`,` ***sectors***`,`
cylinder`,` ***head***`,`
sector`,` ***buffer***`);`
	Legge dei settori da un'unità a disco. Questa funzione viene usata soltanto da ***dsk_read_sectors()***.

`uint16_t _int13_03 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_03 (`***drive***`,` ***sectors***`,`
cylinder`,` ***head***`,`
sector`,` ***buffer***`);`
	Scrive dei settori in un'unità a disco. Questa funzione viene usata solo da ***dsk_write_sectors()***.

`uint16_t _int16_00 (void);` `void int16_00 (void);`	Legge un carattere dalla tastiera, rimuovendolo dalla memoria tampone relativa. Viene usata solo in alcune funzioni di controllo della console, denominate '**`con_…()`**'.
`uint16_t _int16_01 (void);` `void int16_01 (void);`	Verifica se è disponibile un carattere dalla tastiera: se c'è ne restituisce il valore, ma senza rimuoverlo dalla memoria tampone relativa, altrimenti restituisce zero. Viene usata solo dalle funzioni di gestione della console, denominate '**`con_…()`**'.

Funzione o macroistruzione	Descrizione
`void _int16_02 (void);` `void int16_02 (void);`	Restituisce un valore con cui è possibile determinare quali funzioni speciali della tastiera risultano inserite (inserimento, fissamaiuscole, blocco numerico, ecc.). Al momento la funzione non viene usata.
`uint16_t _in_8 (uint16_t port);` `void in_8 (port);`	Legge un byte dalla porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.
`uint16_t _in_16 (uint16_t port);` `void in_16 (port);`	Legge un valore a 16 bit dalla porta di I/O indicata. Questa funzione viene usata solo da ***dev_mem()***.

`void _out_8 (uint16_t port,`
`uint16_t value);`
`void out_8 (port, value);`
	Scrive un byte nella porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.

`void _out_16 (uint16_t port,`
`uint16_t value);`
`void out_16 (port, value);`
	Scrive un valore a 16 bit nella porta indicata. Questa funzione viene usata solo da ***dev_mem()***.

`void cli (void);`	Azzera l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '**`CLI`**' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void sti (void);`	Attiva l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '**`STI`**' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void irq_on (unsigned int irq);`	Abilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.
`void irq_off (unsigned int irq);`	Disabilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.

`void _ram_copy (segment_t org_seg,`
`offset_t org_off,`
`segment_t dst_seg,`
`offset_t dst_off,`
`uint16_t size);`
`void ram_copy (org_seg,`
`org_off,`
`dst_seg,`
`dst_off,`
`size);`
	Copia una certa quantità di byte, da una posizione di memoria all'altra, specificando segmento e scostamento di origine e destinazione. Viene usata solo dalle funzioni '**`mem`**_...**`()`**'.

Gestione della console

«

Listato u0.5 e successivi.

La console offre solo funzionalità elementari, dove è possibile scrivere o leggere un carattere alla volta, sequenzialmente. Ci sono al massimo quattro console virtuali, selezionabili attraverso le combinazioni di tasti [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*] (ma nella configurazione predefinita vengono attivate solo le prime due) e non è possibile controllare i colori o la posizione del testo che si va a esporre; in pratica si opera come su una telescrivente. Le funzioni di livello più basso, relative alla console hanno nomi che iniziano per '**`con`**_...**`()`**'.

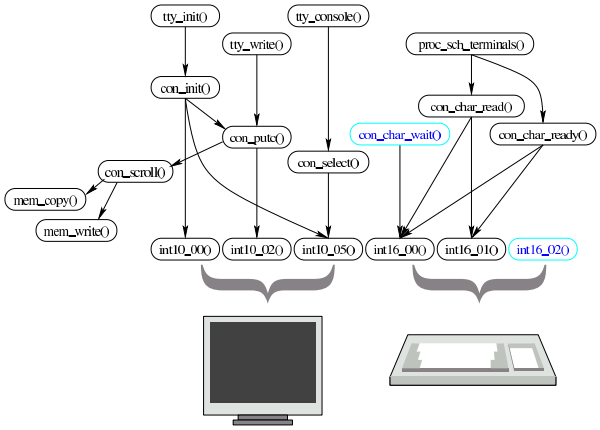
Nel codice del kernel si vede usata frequentemente la funzione ***k_printf()***, la quale va a utilizzare la funzione ***k_vprintf()***, dove poi, attraverso altri passaggi, si arriva a utilizzare la funzione ***con_putc()***.

Tabella u144.3. Funzioni per l'accesso alla console, dichiarate nel file di intestazione '`kernel/ibm_i86.h`' e descritte nei file contenuti nella directory '`kernel/ibm_i86/`'.

Funzione	Descrizione
`int con_char_read (void);`	Legge un carattere dalla console, se questo è disponibile, altrimenti restituisce il valore zero. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`int con_char_wait (void);`	Legge un carattere dalla console, ma se questo non è ancora disponibile, rimane in attesa, bloccando tutto il sistema operativo. Questa funzione non è utilizzata.
`int con_char_ready (void);`	Verifica se è disponibile un carattere dalla console: se è così, restituisce un valore diverso da zero, corrispondente al carattere in attesa di essere prelevato. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`void con_init (void);`	Inizializza la gestione della console. Questa funzione viene usata solo da ***tty_init()***.
`void con_select (int console);`	Seleziona la console desiderata, dove la prima si individua con lo zero. Questa funzione viene usata solo da ***tty_console()***.
`void con_putc (int console,` `int c);`	Visualizza il carattere indicato sullo schermo della console specificata, sulla posizione in cui si trova il cursore, facendolo avanzare di conseguenza e facendo scorrere il testo in alto, se necessario. Questa funzione viene usata solo da ***tty_write()***.
`void con_scroll (int console);`	Fa avanzare in alto il testo della console selezionata. Viene usata internamente, solo dalla funzione ***con_putc()***.

Nella figura successiva si vede l'interdipendenza tra le funzioni relative alla gestione di basso livello della console. In un altro capitolo si descrivono le funzioni '**`tty`**_...**`()`**', con le quali si gestiscono i terminali in forma più astratta. Nello schema successivo si può vedere che la funzione ***con_scroll()*** si avvale di funzioni per la gestione della memoria: infatti, lo scorrimento del testo dello schermo si ottiene intervenendo direttamente nella memoria utilizzata per la rappresentazione del testo sullo schermo.

Figura u144.4. Interdipendenza tra le funzioni relative alla gestione della console.

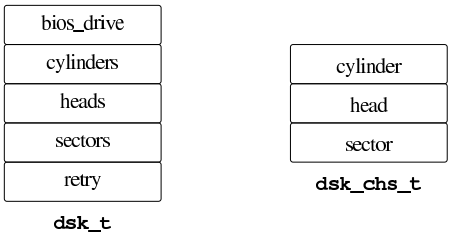


Gestione dei dischi

«

Listato u0.5 e successivi.

Nel file ‘`ibm_i86.h`’ vengono definiti due tipi derivati: ‘**`dsk_t`**’ per annotare le caratteristiche di un disco; ‘**`dsk_chs_t`**’ per annotare simultaneamente le coordinate di accesso a un disco, formate dal numero del cilindro, della testina e del settore.



Le funzioni con nomi del tipo ‘**`dsk_…()`**’ riguardano l’accesso ai dischi, a livello di settore o di byte, e utilizzano le informazioni annotate nell’array ***dsk_table[]***, composto da elementi di tipo ‘**`dsk_t`**’. In pratica, l’array ***dsk_table[]*** viene creato con ‘**`DSK_MAX`**’ elementi (pertanto solo quattro), uno per ogni disco che si intende gestire. Quando le funzioni ‘**`dsk_…()`**’ richiedono l’indicazione di un numero di unità (*drive*), si riferiscono all’indice dell’array ***dsk_table[]*** (al contrario, le funzioni ‘**`_int13_…()`**’ hanno come riferimento il codice usato dal BIOS).

La funzione ***dsk_setup()*** compila l’array ***dsk_table[]*** con i dati relativi ai dischi che si utilizzano; la funzione ***dsk_reset()*** azzera la funzionalità di una certa unità; la funzione ***dsk_sector_to_chs()*** converte il numero assoluto di un settore nelle coordinate corrispondenti (cilindro, testina e settore).

Le funzioni ***dsk_read_sectors()*** e ***dsk_write_sectors()*** servono a leggere o scrivere una quantità stabilita di settori, usando come appoggio un’area di memoria individuata da un puntatore generico. Le funzioni ***dsk_read_bytes()*** e ***dsk_write_bytes()*** svolgono un compito equivalente, ma usando come riferimento il byte; in questo caso, restituiscono la quantità di byte letti o scritti rispettivamente.

Tabella u144.6. Funzioni per l’accesso ai dischi, dichiarate nel file di intestazione ‘`kernel/ibm_i86.h`’.

Funzione	Descrizione
`void dsk_setup (void);`	Predispone il contenuto dell’array ***dsk_table[]***. Questa funzione viene usata soltanto da ***main()***.
`int dsk_reset (int drive);`	Azzera lo stato dell’unità corrispondente a ***dsk_table[drive].bios_drive***. Viene usata solo internamente, dalle altre funzioni ‘**`dsk_…()`**’.
`void dsk_sector_to_chs (int drive, unsigned int sector, dsk_chs_t *chs);`	Modifica le coordinate della variabile strutturata a cui punta l’ultimo parametro, con le coordinate corrispondenti al numero di settore fornito. Viene usata solo internamente, dalle altre funzioni ‘**`dsk_…()`**’.
`int dsk_read_sectors (int drive, unsigned int start_sector, void *buffer, unsigned int n_sectors);`	Legge una sequenza di settori da un disco, mettendo i dati in memoria, a partire dalla posizione espressa da un puntatore generico. La funzione è ricorsiva, ma oltre che da se stessa, viene usata internamente da ***dsk_read_bytes()*** e da ***dsk_write_bytes()***.
`int dsk_write_sectors (int drive, unsigned int start_sector, void *buffer, unsigned int n_sectors);`	Scrive una sequenza di settori in un disco, traendo i dati da un puntatore a una certa posizione della memoria. La funzione è ricorsiva, ma oltre che da se stessa, viene usata solo internamente da ***dsk_write_bytes()***.
`size_t dsk_read_bytes (int drive, off_t offset, void *buffer, size_t count);`	Legge da una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.
`size_t dsk_write_bytes (int drive, off_t offset, void *buffer, size_t count);`	Scrive su una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.

Figura u144.7. Interdipendenza tra le funzioni relative all’accesso ai dischi.