

Dispositivi

File «lib/sys/os16.h» e directory «lib/sys/os16/»	3093
File «kernel/devices.h» e «kernel/devices/...»	3094
Numero primario e numero secondario	3098
Dispositivi previsti	3099

devices.h 3094 devices/dev_tty.c 3094 DEV_CONSOLE 3099
DEV_CONSOLEn 3099 dev_dsk.c 3094 DEV_DSKn 3099
dev_io() 3094 dev_io.c 3094 dev_kmem.c 3094
DEV_KMEM_FILE 3099 DEV_KMEM_INODE 3099
DEV_KMEM_MMP 3099 DEV_KMEM_PS 3099 DEV_KMEM_SB 3099
DEV_MEM 3099 DEV_NULL 3099 DEV_PORT 3099
DEV_TTY 3099 DEV_ZERO 3099 os16.h 3093

La gestione dei dispositivi fisici, da parte di os16, è molto limitata, in quanto ci si avvale prevalentemente delle funzionalità già offerte dal BIOS. In ogni caso, tutte le operazioni di lettura e scrittura di dispositivi, passano attraverso la gestione comune della funzione *dev_io()*.

File «lib/sys/os16.h» e directory «lib/sys/os16/»

Listato [u0.12](#) e altri.

Una parte delle definizioni che riguardano la gestione dei dispositivi, necessaria al kernel, è disponibile anche alle applicazioni attraverso il file 'lib/sys/os16.h'. Al suo interno, tra le altre cose, è definito un insieme di macro-variabili, con prefisso **DEV_...**, con cui si distinguono i numeri attribuiti ai dispositivi. Per esempio,

DEV_DSK_MAJOR corrisponde al numero primario (*major*) per tutte le unità di memorizzazione, mentre ***DEV_DSK1*** corrisponde al numero primario e secondario (*major* e *minor*), in un valore unico, della seconda unità a disco.

File «kernel/devices.h» e «kernel/devices/...»

«

Listati [u0.2](#) e altri.

Il file ‘kernel/devices.h’ incorpora il file ‘lib/sys/os16/os16.h’, per acquisire le funzionalità legate alla gestione dei dispositivi che sono disponibili anche agli applicativi. Successivamente dichiara la funzione ***dev_io()***, la quale sintetizza tutta la gestione dei dispositivi. Questa funzione utilizza il parametro ***rw***, per specificare l’azione da svolgere (lettura o scrittura). Per questo parametro vanno usate le macro-variabili ***DEV_READ*** e ***DEV_WRITE***, così da non dover ricordare quale valore numerico corrisponde alla lettura e quale alla scrittura.

```
ssize_t dev_io (pid_t pid, dev_t device, int rw, off_t offset,  
               void *buffer, size_t size, int *eof);
```

Sono comunque descritte anche altre funzioni, ma utilizzate esclusivamente da ***dev_io()***.

La funzione ***dev_io()*** si limita a estrapolare il numero primario dal numero del dispositivo complessivo, quindi lo confronta con i vari tipi gestibili. A seconda del numero primario seleziona una funzione appropriata per la gestione di quel tipo di dispositivo, passando praticamente gli stessi argomenti già ricevuti.

Va osservato il caso particolare dei dispositivi ‘**DEV_KMEM_...**’. In un sistema operativo Unix comune, attraverso ciò che fa capo al file di dispositivo ‘`/dev/kmem`’, si ha la possibilità di accedere all’immagine in memoria del kernel, lasciando a un programma con privilegi adeguati la facoltà di interpretare i simboli che consentono di individuare i dati esistenti. Nel caso di `os16`, non ci sono simboli nel risultato della compilazione, quindi non è possibile ricostruire la collocazione dei dati. Per questa ragione, le informazioni che devono essere pubblicate, vengono controllate attraverso un dispositivo specifico. Quindi, il dispositivo ‘**DEV_KMEM_PS**’ consente di leggere la tabella dei processi, ‘**DEV_KMEM_MMAP**’ consente di leggere la mappa della memoria, e così vale anche per altre tabelle.

Per quanto riguarda la gestione dei terminali, attraverso la funzione `dev_tty()`, quando un processo vuole leggere dal terminale, ma non risulta disponibile un carattere, questo viene messo in pausa, in attesa di un evento legato ai terminali.

`os16`, non disponendo di un sistema di trattenimento dei dati in memoria (*cache*), esegue le operazioni di lettura e scrittura dei dispositivi in modo immediato. Per questo motivo, la distinzione tra file di dispositivo a blocchi e a caratteri, rimane puramente estetica, ovvero priva di un’utilità concreta.

Figura u147.1. Interdipendenza tra la funzione *dev_io()* e le altre. I collegamenti con le funzioni *major()* e *minor()* sono omesse.

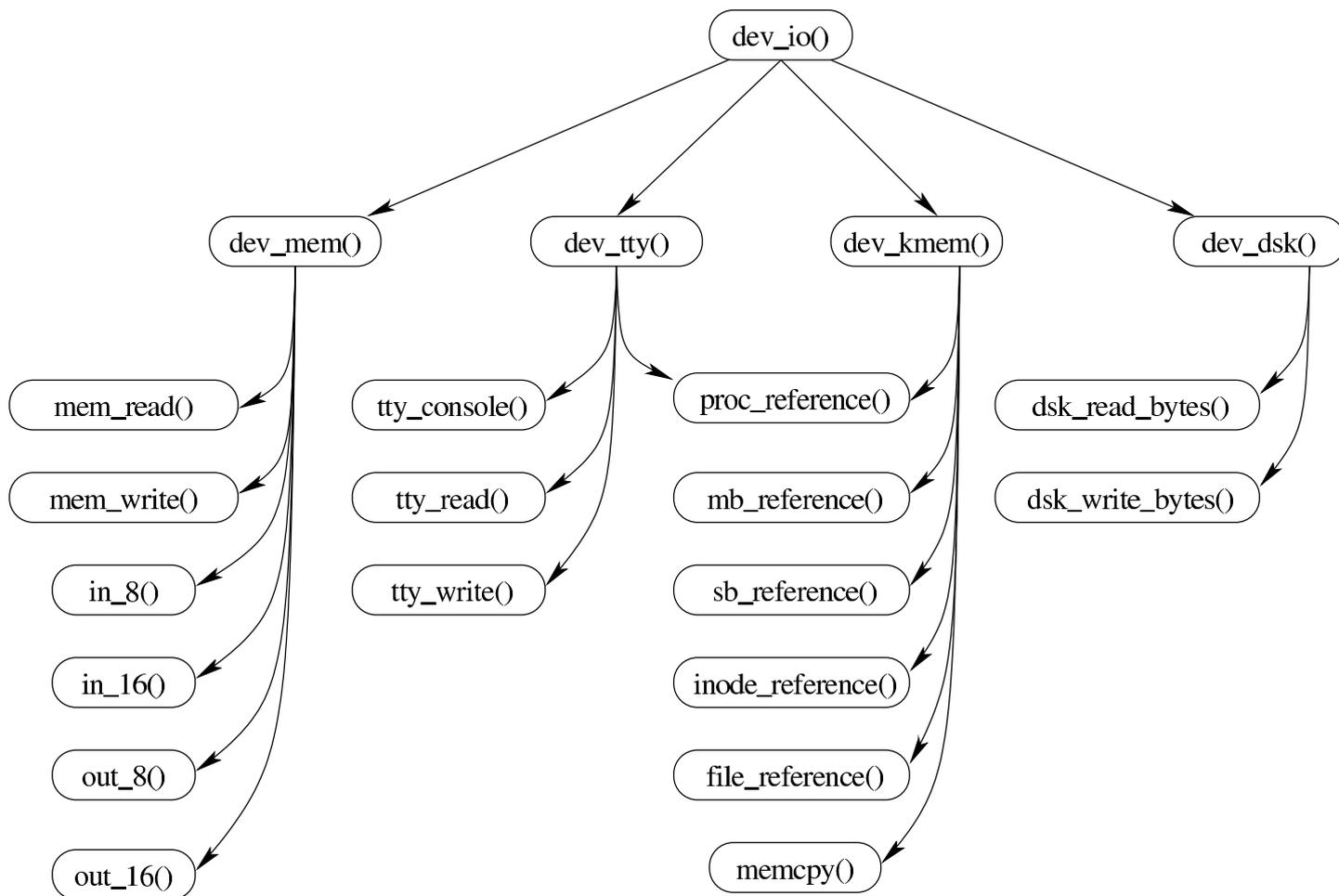
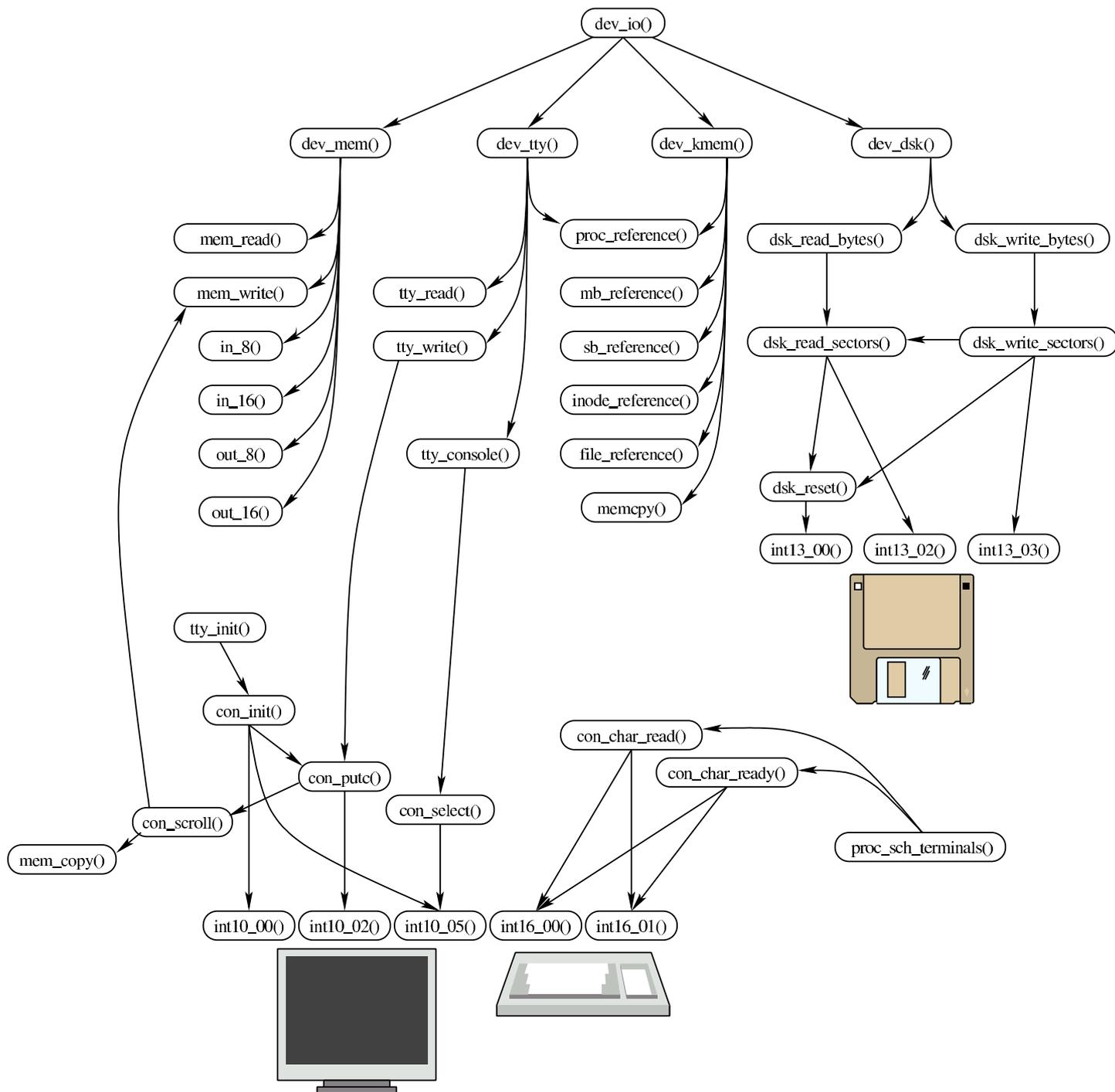


Figura u147.2. Schema più ampio delle dipendenze che hanno origine dalla funzione *dev_io()*.



Numero primario e numero secondario

«

I dispositivi, secondo la tradizione dei sistemi Unix, sono rappresentati dal punto di vista logico attraverso un numero intero, senza segno, a 16 bit. Tuttavia, per organizzare questa numerazione in modo ordinato, tale numero viene diviso in due parti: la prima parte, nota come *major*, ovvero «numero primario», si utilizza per individuare il tipo di dispositivo; la seconda, nota come *minor*, ovvero «numero secondario», si utilizza per individuare precisamente il dispositivo, nell'ambito del tipo a cui appartiene.

In pratica, il numero complessivo a 16 bit si divide in due, dove gli 8 bit più significativi individuano il numero primario, mentre quelli meno significativi danno il numero secondario. L'esempio seguente si riferisce al dispositivo che genera il valore zero, il quale appartiene al gruppo dei dispositivi relativi alla memoria:

DEV_MEM_MAJOR	01 ₁₆
DEV_ZERO	0104 ₁₆

In questo caso, il valore che rappresenta complessivamente il dispositivo è 0104₁₆ (pari a 260₁₀), ma si compone di numero primario 01₁₆ e di numero secondario 04₁₆ (che coincidono nella rappresentazione in base dieci). Per estrarre il numero primario si deve dividere il numero complessivo per 256 (0100₁₆), trattenendo soltanto il risultato intero; per filtrare il numero secondario si può fare la stessa divisione, ma trattenendo soltanto il resto della stessa. Al contrario, per produrre il numero del dispositivo, partendo dai numeri primario e secondario separati, occorre moltiplicare il numero primario per

256, sommando poi il risultato al numero secondario.

Dispositivi previsti

L'astrazione della gestione dei dispositivi, consente di trattare tutti i componenti che hanno a che fare con ingresso e uscita di dati, in modo sostanzialmente omogeneo; tuttavia, le caratteristiche effettive di tali componenti può comportare delle limitazioni o delle peculiarità. Ci sono alcune questioni fondamentali da considerare: un tipo di dispositivo potrebbe consentire l'accesso in un solo verso (lettura o scrittura); l'accesso al dispositivo potrebbe essere ammesso solo in modo sequenziale, rendendo inutile l'indicazione di un indirizzo; la dimensione dell'informazione da trasferire potrebbe assumere un significato differente rispetto a quello comune.

Tabella u147.4. Classificazione dei dispositivi di os16.

Dispositivo	Let- tura e scrit- tura r/w	Ac- cesso diret- to o se- quen- ziale	Annotazioni
DEV_MEM	r/w	diret- to	Permette l'accesso alla memo- ria, in modo indiscriminato, per- ché os16 non offre alcun tipo di protezione al riguardo.
DEV_NULL	r/w	nes- suno	Consente la lettura e la scrittura, ma non si legge e non si scrive alcunché.

Dispositivo	Let- tura e scrit- tura r/w	Ac- cesso diret- to o se- quen- ziale	Annotazioni
DEV_PORT	r/w	se- quen- ziale	Consente di leggere e scrivere da o verso una porta di I/O, individuata attraverso l'indirizzo di accesso (l'indirizzo, o meglio lo scostamento, viene trattato come la porta a cui si vuole accedere). Tuttavia, la dimensione dell'informazione da trasferire è valida solo se si tratta di uno o di due byte: per la dimensione di un byte si usano le funzioni <i>in_8()</i> e <i>out_8()</i> ; per due byte si usano le funzioni <i>in_16()</i> e <i>out_16()</i> . Per dimensioni differenti la lettura o la scrittura non ha effetto.
DEV_ZERO	r	se- quen- ziale	Consente solo la lettura di valori a zero (zero inteso in senso binario).
DEV_TTY	r/w	se- quen- ziale	Rappresenta il terminale virtuale del processo attivo.
DEV_DSK n	r/w	diret- to	Rappresenta l'unità a dischi n . os16 non gestisce le partizioni.

Dispositivo	Let- tura e scrit- tura r/w	Ac- cesso diret- to o se- quen- ziale	Annotazioni
DEV_KMEM_PS	r	diret- to	Rappresenta la tabella contenente le informazioni sui processi. L'indirizzo di accesso indica il numero del processo di partenza; la dimensione da leggere dovrebbe essere abbastanza grande da contenere un processo, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
DEV_KMEM_MMP	r	se- quen- ziale	Rappresenta la mappa della memoria, alla quale si può accedere solo dal suo principio. In pratica, l'indirizzo di accesso viene ignorato, mentre conta solo la quantità di byte richiesta.

Dispositivo	Let- tura e scrit- tura r/w	Ac- cesso diret- to o se- quen- ziale	Annotazioni
DEV_KMEM_SB	r	diret- to	Rappresenta la tabella dei su- per blocchi (per la gestione delle unità di memorizzazio- ne). L'indirizzo di accesso ser- ve a individuare il super blocco; la dimensione richiesta dovreb- be essere abbastanza grande da contenere un super blocco, ma anche richiedendo una dimen- sione maggiore, se ne legge uno solo.
DEV_KMEM_INODE	r	diret- to	Rappresenta la tabella degli ino- de (per la gestione delle unità di memorizzazione). L'indiriz- zo di accesso serve a individua- re l'inode; la dimensione richie- sta dovrebbe essere abbastanza grande da contenere un inode, ma anche richiedendo una di- mensione maggiore, se ne legge uno solo.

Dispositivo	Let- tura e scrit- tura r/w	Ac- cesso diret- to o se- quen- ziale	Annotazioni
DEV_KMEM_FILE	r	diret- to	Rappresenta la tabella dei file (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il file; la dimensione richiesta dovrebbe essere abbastanza grande da contenere le informazioni di un file, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
DEV_CONSOLE	r/w	se- quen- ziale	Legge o scrive relativamente alla console attiva la quantità di byte richiesta, ignorando l'indirizzo di accesso.
DEV_CONSOLE n	r/w	se- quen- ziale	Legge o scrive relativamente alla console n la quantità di byte richiesta, ignorando l'indirizzo di accesso.

