Drivers

Introduzione

Tipologie

Struttura

Interazione con il kernel

Driver

- **Un driver è un modulo del sistema operativo**
  - Esterno al kernel
  - Dedicato alla gestione di una specifica periferica
- **Come altre funzionalità del sistema operativo**
  - Fornisce una virtualizzazione delle periferiche
- **Il programmatore che utilizza i servizi offerti da un driver**
  - Utilizza sempre poche semplici funzioni
  - Non ha necessità di conoscere i dettagli hardware della specifica periferica
- **Un driver interagisce con**
  - L'applicazione utente
    - Mediante un'interfaccia standard
  - Il file system
    - Tutte le periferiche sono viste come file speciali
    - In questo consiste l'aspetto principale della virtualizzazione
  - Il kernel
    - Per la gestione degli eventi e della sincronizzazione

Driver e periferiche

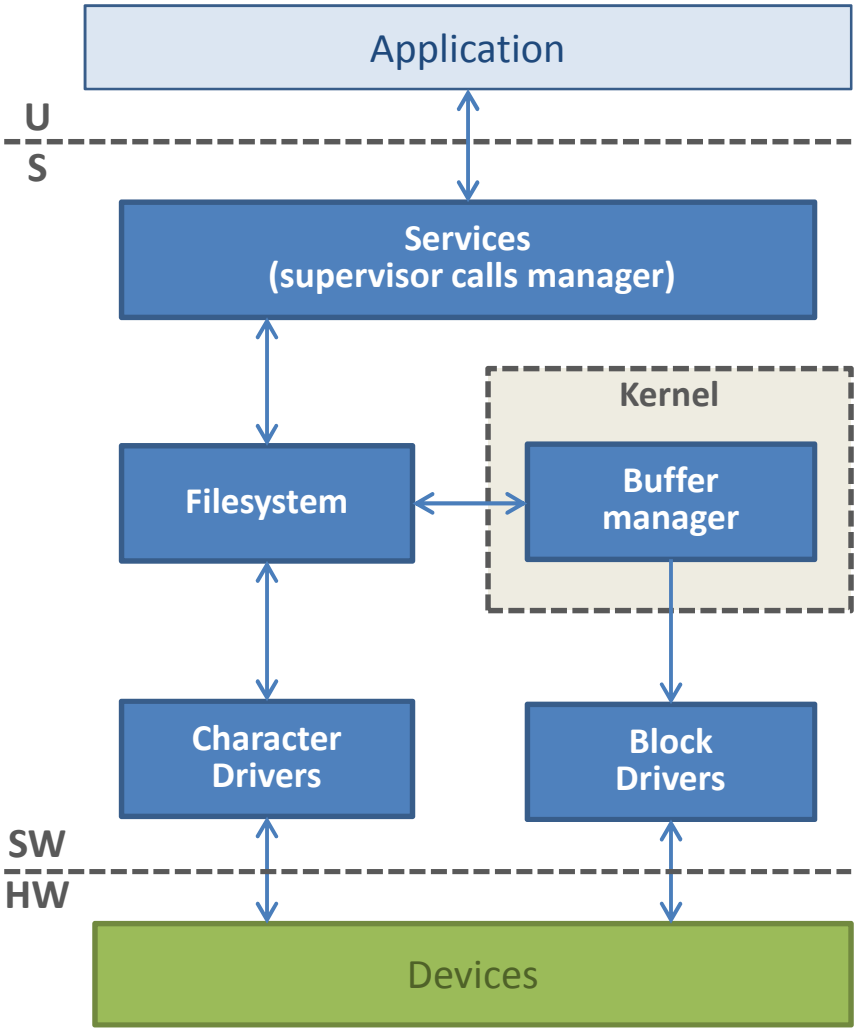
- **UNIX in generale, e Linux nello specifico distinguono le periferiche in**
  - Periferiche a carattere
  - Periferiche a blocchi
- **Periferiche a carattere (character device)**
  - Sono periferiche che consentono unicamente un accesso sequenziale
    - Terninale, stampante, interfaccia di rete, ...
  - Il trasferimento può avvenire
    - Un byte alla volta, come caso tipico
    - Un blocco di byte alla volta, purché l'accesso sia sequenziale
- **Periferiche a blocchi (block device)**
  - Sono periferiche che supportano l'accesso casuale
    - Dischi, CD/DVD, schede di memoria flash, ...
  - Per accedere a un blocchо è necessario indicizzarlo
  - Il trasferimento
    - Avviene per blocchi di dimensione arbitraria
    - Si appoggia ad un servizio di gestione dei buffer offerto dal sistema operativo

Driver e periferiche

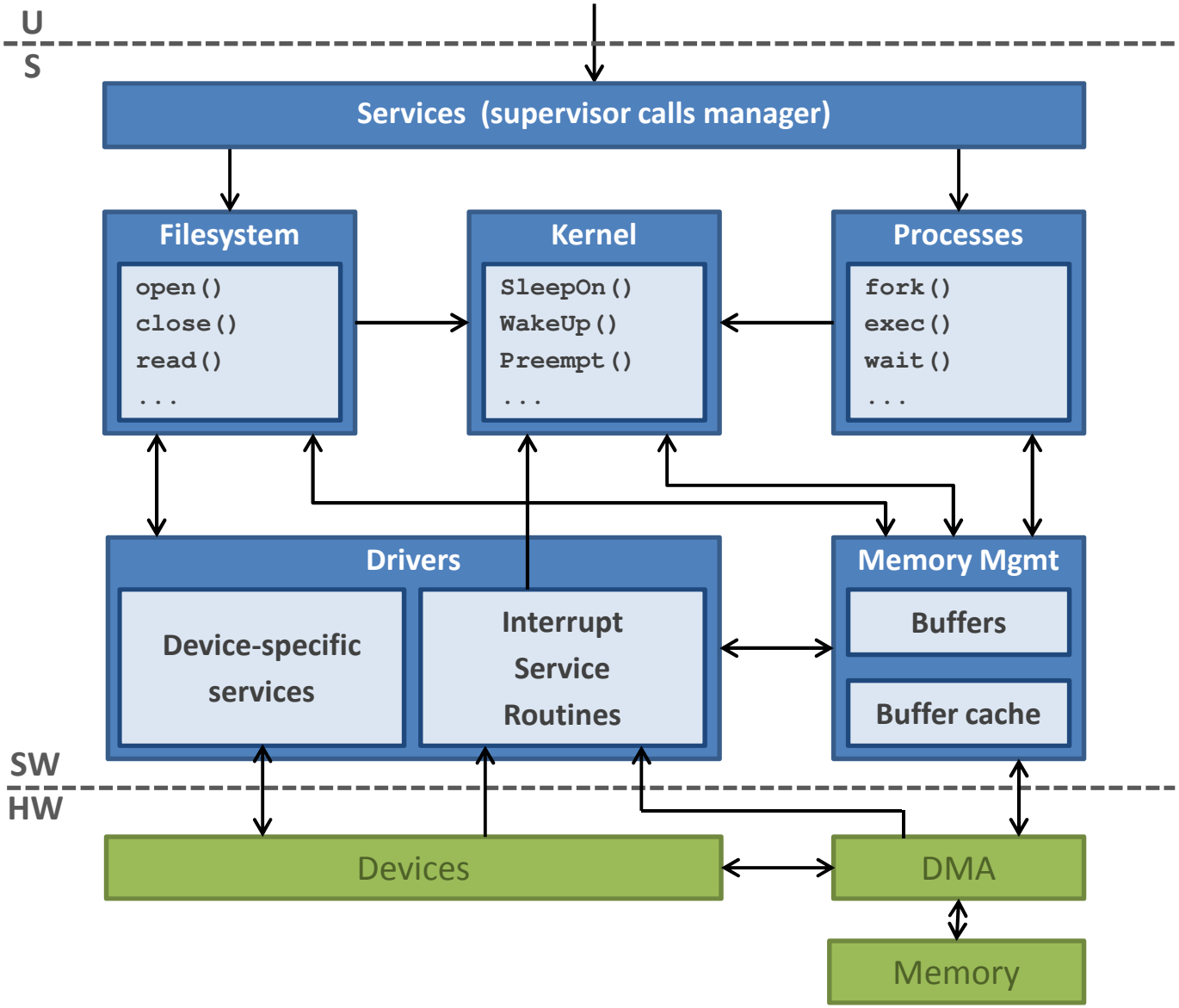
- **Conseguentemente esistono due tipologie di driver**

- **Driver a carattere (character driver)**
  - Interagiscono in modo diretto con il file system

- **Driver a blocchi (block driver)**
  - Interagiscono con il filesystem attraverso i buffer di sistema
  - Questi driver utilizzano in particolare
    - Il supporto del gestore della memoria
    - Una tecnica di trasferimento dati nota come Direct Memory Access o DMA (trattato nella parte di architettura)



Interazione con il kenrel



File speciali

- **Ad ogni periferica è associato un file speciale**
  - Non è un vero file
  - Ha un nome e un i-node associato
    - Nell'i-node sono contenute informazioni relative alla periferica/driver
- **I file speciali**
  - Sono contenuti nella directory /dev
  - Vengono mostrati in un modo specifico dal comando ls
  - Hanno nomi che indicano il tipo di periferica a cui si riferiscono
- **L'i-node associato ad un file speciale indica**
  - Il tipo di periferica/driver
    - c: Caratttere
    - b: Blocchi
  - Un "major number"
    - Indica il tipo di periferica (disco, terminale, ...)
  - Un "minor number"
    - Indica la specifica periferica, tra tutte quelle dello stesso tipo disponibili nel sistema

File speciali

- **L'accesso a un dispositivo avviene mediante chiamate a funzioni del filesystem**

```
...
fd = open( "/dev/lp0", O_WRONLY );
write( fd, buffer, buffer_length );
close( fd );
...
```

- **Il processo che consente l'accesso corretto è il seguente**
  - Mediante il nome del file speciale viene individuato l'i-node corrispondente
    - Esattamente come per qualsiasi tipo di file
    - Una parte dell'i-node specifica che si tratta di un file speciale e indica il tipo, il major number ed il minor number
  - Grazie al major number specificato nell'i-node si accede al driver corretto
    - Possiamo immaginare un driver come un insieme di funzioni
  - Il nome della funzione identifica il servizio effettivamente richiesto
    - Si sceglie cioè uno tra i servizi specifici del driver
  - Il minor number viene passato come argomento alla specifica funzione di servizio
    - In questo modo è possibile accedere univocamente a ogni device del sistema

File speciali

- **Per poter gestire con un'interfaccia comune tutte le possibili periferiche è necessario definire un meccanismo di accesso omogeneo**
- **Ciò è realizzato mediante la struttura dati seguente, detta "file operations"**

```
struct file_operations {
  int   (*lseek)( );
  int   (*read)( );
  int   (*write)( );
  ...
  int   (*ioctl)( );
  int   (*open)( );
  void  (*release)( );
  ...
};
```

- **Si tratta di un insieme di puntatori a funzione**
  - Ogni campo punta ad una funzione che implementa uno specifico servizio
  - Queste informazioni vengono copiate in una opportuna tabella del sistema operativo nel momento in cui la periferica viene inizializzata al boot del sistema operativo

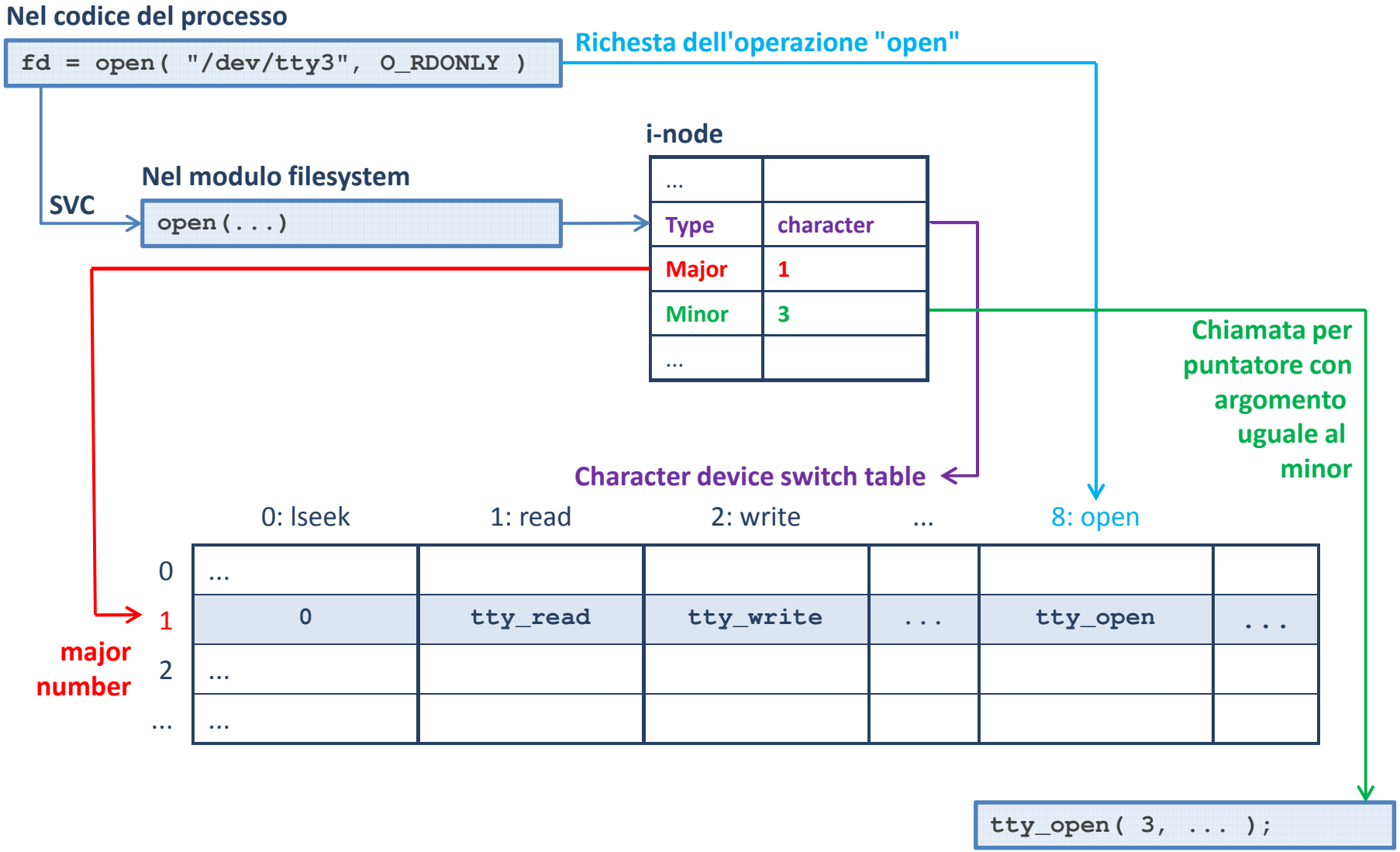
File speciali

- **All'atto dell'inizializzazione viene invocata una speciale funzione per ogni driver**
  - Esegue operazioni di inizializzazione specifiche del dispositivo
  - Restituisce un puntatore alla struttura delle file operations
- **Il sistema operativo copia i puntatori alle funzioni in una tabella**
  - Block device switch table, per i dispositivi a blocchi
  - Character device switch table, per i dispositivi a caratteri
- **Il major number identifica un device specifico, quindi una riga della tabella**
- **L'operazione richiesta (open, close, ...) indica una colonna della tabella**
- **In questo modo si accede, tramite puntatore, alla specifica funzione del driver**

operation

major number	0: lseek	1: read	2: write	...	8: open	
0	...					
1	`(*lseek)()`	`(*read)()`	`(*write)()`	`...`	`(*open)()`	`...`
2	...					
...	...					

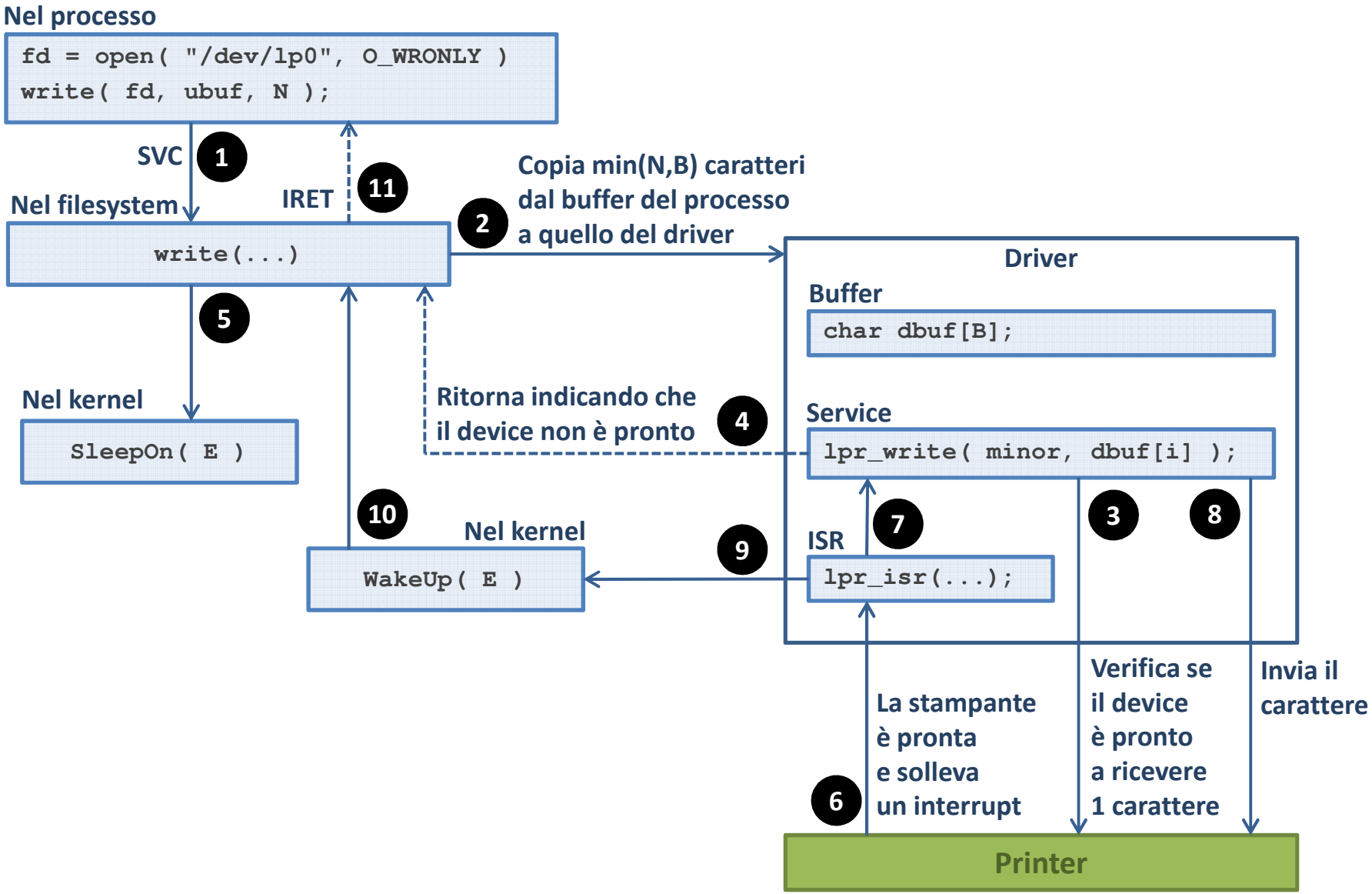
Accesso ai servizi di una periferica



Accesso a una periferica a carattere

- **Consideriamo il caso in cui la periferica a carattere sia gestita a interrupt**
  - Il processo non rimane in busy waiting ad attendere che la pariferica sia pronta
- **Il processo P richiede una operazione al device D**
  - A un certo punto viene eseguita una service call
    - Eseguirà la specifica funzione nel contesto di P
  - Se D è pronto, l'operazione viene completata immediatamente
  - Se D non è pronto
    - Il processo P viene sospeso in attesa di un evento E(D) associato al device D
    - Il sistema pone in esecuzione un altro processo Q
- **Quando il device D è pronto**
  - Solleva un interrupt che viene intercettato da una specifica ISR del driver
- **La ISR**
  - Viene eseguita mente un Q, e non P, è in esecuzione
    - I dati scambiati con la periferica devono essere mantenuti in un buffer del device driver
  - Notifica l'evento E(D)
  - Risveglia i processi sospesi su E(D) mendiante l'apposita funzione del kernel

Accesso a una periferica a carattere



Accesso a una periferica a blocchi

- **Nel caso di una periferica a blocchi**
  - Il sistema demanda al DMA l'operazione di trasferimento dei dati
- **Il processo P richiede una operazione al device D**
  - A un certo punto viene eseguita una service call
    - Eseguirà la specifica funzione nel contesto di P
  - La funzione del driver
    - Configura il DMA per il trasferimento dati e lo avvia
    - Ritorna alla routine di servizio del file system indicando che il trasferimento non è completo
    - Il processo P viene sospeso in attesa di un evento E(DMA) associato al DMA
    - Il sistema pone in esecuzione un altro processo Q
- **Il trasferimento via DMA avviene in parallelo all'eseczuione del software**
  - E' interamente gestito dall'hardware
- **Quando il DMA conclude il trasferimento solleva un interrupt di DMA**
- **La ISR che serve l'interrupt del DMA**
  - Viene eseguita mente un Q, e non P, è in esecuzione
  - Notifica l'evento E(DMA)
  - Risveglia i processi sospesi su E(DMA) mendiante l'apposita funzione del kernel

Accesso a una periferica a blocchi