Progetto di Sistemi Operativi Avanzati

A. Chillotti*

A.A. 2021/2022

1 Traccia del progetto

This specification is related to a Linux device driver implementing low and high priority flows of data. Through an open session to the device file a thread can read/write data segments. The data delivery follows a First-in-First-out policy along each of the two different data flows (low and high priority). After read operations, the read data disappear from the flow. Also, the high priority data flow must offer synchronous write operations while the low priority data flow must offer an asynchronous execution (based on delayed work) of write operations, while still keeping the interface able to synchronously notify the outcome. Read operations are all executed synchronously. The device driver should support 128 devices corresponding to the same amount of minor numbers. The device driver should implement the support for the ioctl(..) service in order to manage the I/O session as follows:

- setup of the priority level (high or low) for the operations
- blocking vs non-blocking read and write operations
- setup of a timeout regulating the awake of blocking operations

A a few Linux module parameters and functions should be implemented in order to enable or disable the device file, in terms of a specific minor number. If it is disabled, any attempt to open a session should fail (but already open sessions will be still managed). Further additional parameters exposed via VFS should provide a picture of the current state of the device according to the following information:

- enabled or disabled
- number of bytes currently present in the two flows (high vs low priority)
- number of threads currently waiting for data along the two flows (high vs low priority)

1.1 Spiegazione traccia del progetto

Bisogna implementare un driver che implementa delle file operations. Con queste file operations dobbiamo implementare un multi flow device file, ovvero un device file (file logico) ed è multi-flow perché ci sono diversi stream di byte. I sistemi sono: low-priority e high-priority.

Queste due rappresentano la modalità con la quale dobbiamo lavorare su questo stream. Entrambi gli stream sono FIFO, quindi inseriamo dei dati lì e li estriamo nello stesso ordine. Quando estriamo i dati (ovvero leggiamo), essi scompaiono e quindi questo device file ha delle similarità rispetto ad una FIFO tradizionale di Linux. La cosa interessante è che in questi due flussi lavoriamo secondo schemi differenti:

*alessandro.chillotti@outlook.it

- low priority: esecuzione asincrona, ovvero delayed. Quindi, un thread chiama `write` sullo stream low-priority, i dati vengono presi ma verranno messi più avanti. Quindi, i dati che questo thread consegna vanno in una staging area e poi verranno processatti più avanti[1]

  Quando scriviamo dei dati, in realtà la notifica della scrittura è comunque sincrona perché altrimenti andremo contro la specifica di `read/write` dei file all'interno di Linux (i.e. quando chiamiamo `read`, il ritorno della `read` è la conclusione di quella operazione di `read`, ma poi che stanno in una staging area è un'altra storia).

- high priority: le operazioni di scrittura sono sincrone, quindi un thread chiama una `write` e la `write` deve mettere i dati lì ora, quindi il thread non può riprendere il controllo a livello user finché la `write` non è conclusa.

Il driver è una solo, ma i minor number sono 128. Dentro il driver ?? il minor number e andiamo a lavorare dove vogliamo. Con una `ioctl()` all'interno di questo driver possiamo settare l'operatività della sessione di I/O verso questo file perché quando lavoriamo su questo file lavoriamo in un canale che ci porta su una sessione che ci porta su un file. Con `ioctl` possiamo:

- settare high o low priority.
- settare se le operazioni sono bloccati o non bloccanti (e.g. voglio dei dati in lettura, non ci sono e allora rimangono bloccato oppure non voglio aspettare)
- settare il timeout per sbloccare le operazioni bloccate.

Nei parametri del modulo Linux che andremo a montare, ci sono anche dei parametri per acquisire lo stato di esercizio di questo multi-flow device file. In particolare, la cosa interessante è:

- capire se il multi-flow device file è abilitato o disabilitato

  Osservazione: Abilitato o disabilitato significa che ci saranno delle variabili che possiamo controllare dal VFS per questo modulo per cui ciascuno dei quei minor number possiamo temporaneamente bloccare le aperture di sesssione. Quindi, c'è l'idea di abilitare o disabilitare uno specifico minor number e questo significa dire che non possiamo aprire sessioni verso quello specifico minor number.

- se e quanti sono i byte presenti nei due flussi
- se e quanti sono i thread che aspettano i dati sul loro flusso

[1] Il più avanti lo scegliamo noi come.