Progetto Sistemi Operativi Avanzati 2021/2022

Multiflow Device Driver

Danilo Dell’Orco 0300229

# Introduzione

Lo scopo del seguente documento è esporre e spiegare le scelte progettuali effettuate e i meccanismi adottati nello sviluppo di un modulo per il kernel Linux.

## Specifica – Multiflow Device

La specifica è relativa a un driver di dispositivo Linux che implementa flussi di dati a bassa e alta priorità. Attraverso una sessione aperta al device file un thread può leggere e scrivere segmenti di dati. La trasmissione dei dati segue una politica di First-in-First-out lungo ciascuno dei due diversi flussi di dati (bassa e alta priorità). Dopo le operazioni di lettura, i dati letti scompaiono dal flusso. Inoltre, il flusso di dati ad alta priorità deve offrire operazioni di scrittura sincrone, mentre il flusso di dati a bassa priorità deve offrire un'esecuzione asincrona (basata sul lavoro ritardato) delle operazioni di scrittura, pur mantenendo l'interfaccia in grado di notificare in modo sincrono il risultato.

Le operazioni di lettura sono tutte eseguite in modo sincrono. Il driver del dispositivo dovrebbe supportare 128 dispositivi corrispondenti alla stessa quantità di minor number. Il driver del dispositivo dovrebbe implementare il supporto per il servizio ioctl(..) al fine di gestire la sessione di I/O come segue:

* Impostare il livello di priorità (alto o basso) per le operazioni.
* Operazioni di lettura e scrittura bloccanti o non bloccanti.
* Configurare un timeout che regoli il risveglio delle operazioni bloccanti.

Alcuni parametri e funzioni del modulo Linux dovrebbero essere implementati per abilitare o disabilitare il device file, in termini di uno specifico minor number. Se disabilitato, qualsiasi tentativo di aprire una sessione dovrebbe fallire (ma le sessioni già aperte saranno ancora gestite). Ulteriori parametri aggiuntivi esposti tramite VFS dovrebbero fornire un’immagine dello stato attuale del dispositivo in base alle seguenti informazioni:

* Abilitato o disabilitato;
* Numero di byte attualmente presenti nei due flussi (alta o bassa priorità);
* Numero di thread attualmente in attesa di dati lungo i due flussi (alta o bassa priorità).

# Operazioni preliminari

## Identificazione MAJOR e creazione del nodo

Si compila il modulo con make all, e si installa tramite sudo insmod module.ko. Tramite sudo dmesg si verifica il major number assegnato al driver, stampato tramite printk nella funzione init\_module(). A questo punto possiamo creare il char device che utilizzerà effettivamente il driver appena installato.

Tramite sudo /dev/test-dev mknod c MAJOR 0 creiamo un nuovo char device test-dev in /dev, con MAJOR identificato in precedenza e MINOR pari a 0.

## Scelte Progettuali

Il driver può essere utilizzato da differenti sessioni di I/O, ma dobbiamo gestire la concorrenza sul singolo oggetto di I/O. Per questo abbiamo un mutex che gestisce questa cosa.

# Operazioni low priority

// Una scrittura low priority non può fallire, quindi il processo attende attivamente di ottenere il lock. Infatti viene controllato prima se c'è spazio disponibile sul device.

// - Anche se non-bloccante, devo notificare in modo sincrono il risultato della write al client.

// - Quindi si cerca di prendere il lock solo quando viene schedulato il lavoro deferred.

// - Non è possibile prevedere se il lock verrà preso e quindi se la scrittura verrà effettuata.

// - Si assume che nessuna scrittura low priority possa fallire.