Progetto Sistemi Operativi Avanzati 2021/2022

Multiflow Device Driver

Danilo Dell'Orco 0300229

1 Introduzione

Lo scopo del documento è quello di presentare l'implementazione del multiflow_driver, documentando i componenti del modulo, come interagire con esso, e motivando le scelte progettuali effettuate.

1.1 Specifica – Multiflow Device

La specifica è relativa a un device driver 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 un' operazione 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 su delayed work) delle operazioni di scrittura, mantenendo comunque l'interfaccia in grado di notificare in modo sincrono il risultato. Le operazioni di lettura sono invece eseguite tutte 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.

Devono essere implementati anche alcuni parametri del modulo Linux ed alcune funzioni 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).

2 Implementazione

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.

int

3 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.

4 Utilizzo del Modulo

4.1 Organizzazione della Repository

La directory principale del progetto è `soa-project`, che mantiene al suo interno due directory `driver` e `user`.

- **`/driver`**: contiene il codice `multiflow_driver.c` del modulo e lo script `reinstall_module.sh` per smontare versioni precedenti del driver, ricompilare il modulo e montarlo nuovamente nel kernel.
- **`/user`**: contiene il codice `user_cli.c` e l'eseguibile `user_cli` che implementa una semplice CLI per interagire con i dispositivi del driver.

4.2 Montaggio e Rimozione del Modulo

Per installare il modulo si può eseguire lo script `driver/reinstall_module.sh`, oppure eseguire manualmente `make all` per compilare il modulo, e `insmod multiflow_driver.ko` per montarlo. Quando il modulo viene montato, nella `init_module` viene registrato il char device tramite `__register_chrdev`, allocando dinamicamente un nuovo major number per il dispositivo.

Il major number assegnato viene stampato tramite `printk` sul buffer del kernel, quindi può essere recuperato lato utente con il comando `dmesg`. si verifica il major number assegnato al driver, stampato tramite `printk` nella funzione `init_module()`.

Per rimuovere il modulo è possibile sfruttare il comando `rmmod multiflow_driver`, mentre tramite `make clean` vengono rimossi tutti i file generati in fase di compilazione.

4.3 User CLI

Lanciando tramite sudo il programma `user/user_cli` è possibile interagire con il modulo `multiflow_device`. Questo accetta due argomenti da riga di comando, che sono:

- `major` (`argv[1]`): Major number del device installato, che deve essere recuperato tramite `dmesg`.
- `device_path` (`argv[2]`): Percorso del VFS dove verranno installati i dispositivi. Se non viene passato questo parametro si utilizza un path di default (`/dev/mflow-dev`).

Prima di operare con il Char Device è necessario creare i device file che rappresentano i dispositivi sul VFS. Questo può essere fatto:

- Manualmente tramite `mknod dev/nome_device MAJOR MINOR`
- Utilizzando il comando `11` *(Create device nodes)* da `user_cli`, che genera automaticamente *128* file sempre utilizzando `mknod`. I file avranno tutti il major number passato da riga di comando, minor numbers da *0* a *127*, e saranno nominati `c`.

4.3.1 Operazioni sui device

Descriviamo le operazioni offerte dalla CLI per operare con un dispositivo.

- **Open a device file** (**0**): Chiede all'utente di inserire un minor number `N` e viene aperto il relativo file `/dev/mflow-devN`.
- **Write on the device file** (**1**): Effettua la scrittura sul file aperto tramite la system call `write()`.
- **Read from the device file** (**2**): Effettua la lettura dal file aperto, tramite la system call `read()`.

4.3.2 Operazioni sulla sessione

Tramite CLI è possibile modificare alcuni parametri della sessione, che vanno a cambiare il comportamento delle operazioni di write/read. Tutti questi comandi fanno utilizzo di ioctl per andare ad operare sullo stato della sessione.

- **Switch to LOW/HIGH priority** (**3**/**4**): Modifica il parametro priority della sessione, cambiando quindi il flusso dati da HIGH a LOW o viceversa.
- **Use BLOCKING/NON-BLOCKING operations** (**5**/**6**): Modifica il parametro blocking della sessione, cambiando quindi il tipo delle operazioni successive da non-bloccanti a bloccanti o viceversa.
- **Set timeout** (**7**): Modifica il timeout di attesa del lock nelle operazioni bloccanti.

4.3.3 Gestione dei dispositivi

Tramite VFS vengono esposti diversi parametri che rappresentano lo stato del dispositivo, che possono essere letti o manipolati direttamente dalla CLI.

- **Enable/Disable a device file** (**8**/**9**): Richiede un minor number all'utente e abilita o disabilita il dispositivo associato a quel minor. Per fare ciò scrive il valore 0 o 1 nel file `/sys/module/multiflow_driver/parameters/device_enabling`, nella posizione specifica associata al dispositivo.
- **See device status** (**10**): Visualizza tutte le informazioni sullo stato di un dispositivo, specificato dall'utente tramite il minor number. Si accede in lettura ai parametri del modulo `/sys/module/multiflow_driver/parameters/`, mostrando le seguenti informazioni:
  - Stato del dispositivo: `ENABLED`/`DISABLED`
  - Spazio libero disponibile in bytes.
  - Bytes presenti nei due flussi ad alta e bassa priorità
  - Thread in attesa sui due flussi ad alta e bassa priorità
  - Valore del timeout per il lock in millisecondi.