Pandos+: Fase 3

Relazione per il progetto del corso di Sistemi Operativi

J.G. Jaramillo Saa

Università di Bologna
August 7, 2022

1 Risorse

Durante lo sviluppo di questa fase, ho consultato abitualmente e regolarmente le risorse che ci sono state messe a disposizione dal professore (link) oltre al gruppo telegram del progetto.

Inoltre mi sono state d'aiuto le risorse messe a disposizione dagli studenti degli anni scorsi, in particolare ho consultato il seguente repository quando mi sono trovato in difficoltà phase3.

2 Introduzione

Questa fase del progetto si occupa di implementare il livello 4 del sistema operativo: il livello di supporto
Il supporto si occupa di:

1. Memoria virtuale
2. Mediazione semplificata con il livello 3

3 Moduli

Per questa fase è stato deciso di seguire la guida per pandosplus, suddividendo l'implementazione del livello di supporto in 3 file:

- **initProc.c**: Questo modulo implementa `test` e dichiara le variabili globali del livello di supporto (e.g. i semafori dei device e il `master_semaphore`.
- **vmSupport.c**: Questo modulo implementa il pager del sistema operativo. Inoltre contiene la funzione per leggere/scrivere da/sui device di tipo flash. Inoltre la swap pool table e il semaforo della swap pool sono dichiarati in questo modulo.
- **sysSupport.c**: Questo modulo implementa il general exception handler, il syscall exception handler e il program trap exception handler del livello di supporto.

3.1 initProc.c

In questo file viene implementata la funzione di test, che si occupa di:

1. inizializzare la tabella della swap pool e il semaforo della swap pool
2. dichiarare i mutex dei dispositivi di I/O e inizializzarli (per garantire la mutua esclusione sull'accesso dei registri dei dispositivi
3. inizializzare e lanciare `UPROCMAX` U-proc
4. effettuare una **P** sul `master_semaphore`

3.1.1 Inizializzazione degli U-proc

Per lanciare un U-proc è necessario settare i parametri da passare alla **NSYS1**:

1. lo stato del processo da lanciare
2. la support struct del processo da lanciare

Lo stato del processo da lanciare, è stato inizializzato [Section 4.9.1-*pandosplus*] nel seguente modo:

1. **PC** (e il registro *t9*) sono stati settati all'indirizzo `0x8000.00B0` i.e. l'indirizzo di inizio della sezione **.text**
2. **SP** è stato settato a `0xC000.0000`
3. **Status** è stato settato a `(IEPON or IMON or USERPON or TEBITON)` (interrupt abilitati, PLT abilitato, kernel mode disattivata)
4. **EntryHi.ASID** è inizializzato ad un asid compreso tra 1 e 8

La struttura di supporto è stata inizializzata [Section 4.9.1-*pandosplus*] nel seguente modo:

1. `sup_asid`: settato all'asid del processo
2. `sup_exceptContext[2]`: i due contesti (**PC/SP/Status**) sono stati inizializzati a puntare agli handler (`pager`/`general_exception_handler`) con status di interrupt abilitati e kernel mode con PLT abilitato e gli stack pointer sono stati inizializzati in modo tale che utilizzino due frame della RAM a partire dal penultimo frame della RAM
3. `sup_privatePgTbl[32]`: la tabella del processo e' inizializzata in modo tale che i numeri delle pagine comincino da `0x8000.0000` e in modo tale per ogni pagina, il bit `V` sia posto a 0 (la pagina non si trova in memoria) e il bit `D` a 1 (protezione della memoria disabilitata).

3.2 vmSupport.c

In questo file è implementato il pager del sistema operativo, insieme alla funzione di inizializzazione delle strutture dati utilizzate dal pager e la funzione che permette la lettura/scrittura da/verso i flash devices.

3.2.1 Pager

Il pager del sistema operativo è chiamato quando si verificano eccezioni di tipo `page fault`: `TLBL, TLBS, Mod`. Per gestire un page fault, il pager ottiene il puntatore alla struttura di supporto del processo corrente e determina la causa del page fault.

Poichè la swap pool table, è una struttura dati condivisa, essa deve essere acceduta in mutua esclusione. Per garantire tale modalità di accesso è stato deciso di utilizzare un mutex `swap_pool_semaphore` e un vettore di `UPROCMAX` di 0 e 1 utile per tenere traccia quale degli `UPROCMAX` processi sta utilizzando lo `swap_pool_semaphore`, in modo da effettuare una `V` in caso il processo venga terminato prima di arrivare alla fine del codice del pager.

L'algoritmo di rimpiazzamento utilizzato è di tipo FIFO: viene scelto come frame vittima il frame da più tempo in memoria (non è un buon algoritmo ma è molto facile da implementare)

3.2.2 Flash device operation

Quando accade un page fault, potrebbe essere necessario effettuare una operazione di lettura, scrittura o entrambe sul flash device del processo. Per questo tipo di operazioni è stato deciso di utilizzare una funzione `flash_device_operation` che in base al valore del parametro formale `operation` esegue una lettura/scrittura sul flash device associato al processo con struttura di supporto `curr_support`. Infatti data tale struttura di supporto, trovare il *device register* del flash device associatovi, è semplice, basta effettuare la seguente operazione:

```
/* Ricavo l'indirizzo del device register associato al
   flash device dell'asid associato alla struttura di
   supporto passata come parametro */
memaddr dev_reg_addr = (memaddr) (DEVREGSTRT_ADDR + ((FLASHINT - 3)
                       * 0x80) + (asid * 0x10));
```

dove `0x80` è la dimensione di una linea data da `DEVPERINT * DEVREGSIZE` e `0x10` è `DEVREGSIZE`.

Per evitare race condition sui registri dei device **prima** di scrivere sul campo `data0` del registro, è necessario garantire la mutua esclusione. Perciò una operazione di `P` è effettuata sul semaforo `flash_sem[asid]`.

3.3 sysSupport.c

Il gestore delle eccezioni del livello di supporto gestisce tutte le system call con valore `>=` 1 e tutte le program trap (che devono essere trattate come `NSYS2`).

4 Debugging

Per debuggare l'intero codice, ho utilizzato gli strumenti messi a disposizione dall'emulatore tra cui la possibilità di aggiungere breakpoints, le suspect memory regions e le traced memory regions, insieme alla libreria che ci è stata fornita per la stampa di messaggi in memoria cosí da poterli visualizzare.

In particolare ho avuto problemi con un componenete della fase 2: lo scheduler. Il problema era che il primo interrupt ad essere gestito dopo aver effettuato una operazione di `WAIT` era un interrupt di tipo PLT. Utilizzando le suspect, le traced memory regions e visualizzando i contenuti dei registri in binario del processore attraverso l'interfaccia dell'emulatore, sono riuscito a capire quando e dove il codice era sbagliato.