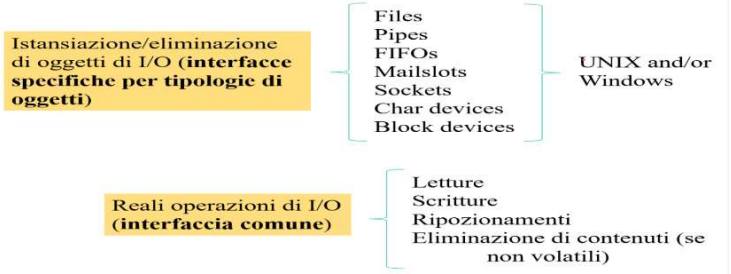
Virtual file system 1 Introduzione

martedì 28 ottobre 2025 15:47

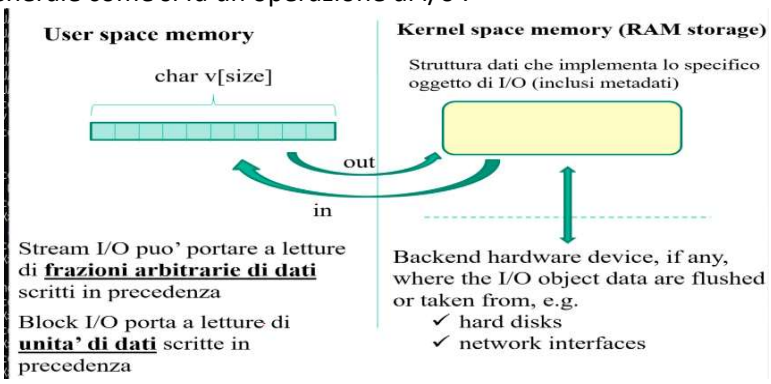
Andiamo ora a vedere il **virtual file system** ovvero l'insieme di tutti i moduli a livello kernel che supportano azioni di i/o. E vengono utilizzati per implementare delle operazioni di i/o, le quali vengono invocate con le system call secondo uno schema omogeneo (si lavora su oggetti eterogenei); inoltre ne esistono anche molte eterogenee (ad hoc -> si basano sul tipo dell'oggetto da istanziare) . Due modelli importanti :

1. **Stream i/o**
    a. Lettura di **frazioni arbitrarie di dati** scritti su oggetto di i/o in precedenza: lettura in modo disparato delle info.
2. **Block i/o**
    a. Lettura di **tutto il contenuto** dell'oggetto: non è possibile fare la lettura frammentata

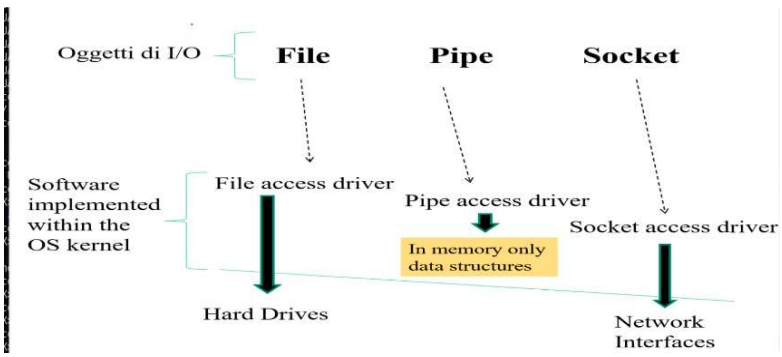
Vediamo ora i vari tipi di oggetti con le opportune operazioni possibili su loro stessi :



Andiamo ora in generale come si fa un'operazione di i/o :

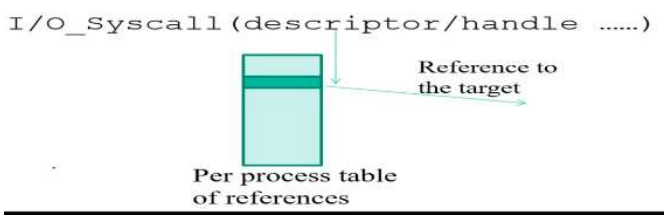


Quindi innanzitutto serve un'area di memoria (buffer) che serve per servire l'operazione . Se di output i dati prima vengono scritti nel buffer e poi il servizio del kernel (li preleva dal nostro address space) e li scrive nella struttura dati che rappresenta l'oggetto di i/o. Questa struttura ha non solo i dati ma anche dei metadati (dati per controllo esistenza + permessi di accesso) . Questi dati possono essere inoltrati (flush) verso dispositivi hw (hard disk) che fanno da back end . Può capitare che questi dati vengano eliminati. Mentre se operazione di input , le info vengono prese da memoria (struttura o hard drive) e copiati nel buffer. Abbiamo detto che usiamo questi oggetti, ma come?? Attraverso i **driver** : l'insieme dei moduli sw di livello kernel che abbiamo per eseguire le operazioni afferenti ad un qualunque oggetto di i/o : ogni tipologia di un oggetto ha i propri driver. Vediamo ora uno schema :

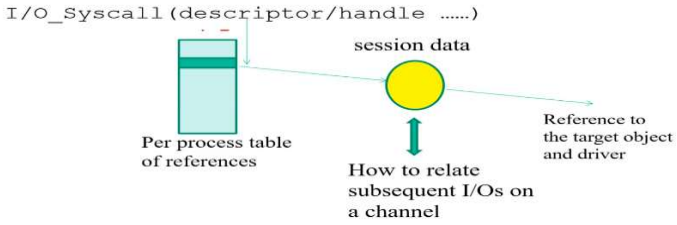


Come si esegue un'operazione di i/o? Quindi avendo oggetto i/o e system call ma mancano i dettagli per lavorare su oggetto i/o : si lavora **quindi su un canale di i/o associato ad un oggetto i/o .** Quindi

intendiamo dei codici (chiavi per accedere all'istanza) numerici per eseguire realmente operazioni di i/o . Questa associazione viene mantenuta nella **per process table references** (tabella che contiene entry dove ci sono info sul target) :



Dove il descriptor si intende in UNIX , mentre l'handle in WINDOWS. Quindi in generale il processo che si ha è il seguente:



Dove si parte dalla entry della tabella che punta verso l'oggetto , si arriva ad una struttura intermedia che rappresenta la **sessione di i/o ,** le quali a loro volta identificano il target object ed il driver da usare. Questa sessione viene mandata in setup dal sw del so quando facciamo il setup di un canale. Quindi in questa struttura si hanno dati temporanei , i quali potranno o meno essere cancellati. Attenzione se unica sessione usata da più processi (magari in seguito a fork).