PIPE UNIX

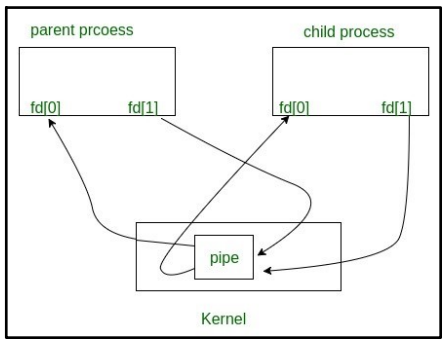
martedì 2 maggio 2023 14:31

PIPE in sistemi UNIX

Un processo ha un thread che chiama una system call per creare una PIPE. La system call è la seguente in UNIX:

int pipe(int fd[2])

Descrizione	invoca la creazione di una PIPE
Argomenti	fd: puntatore ad un buffer di due interi (in fd[0] viene restitutito il descrittore di lettura dalla PIPE, in fd[1] viene restituito il descrittore di scrittura sulla PIPE)
Restituzione	-1 in caso di fallimento



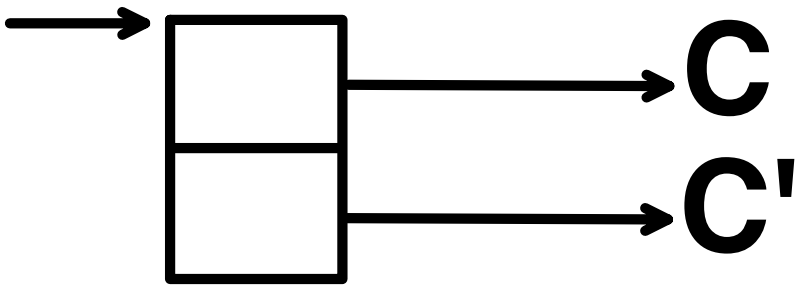
fd[0] è un canale aperto in lettura che consente di leggere dati da una PIPE

fd[1] è un canale aperto in scrittura che consente di immettere dati sulla PIPE

fd[0] e fd[1] possono essere usati come normali descrittori di file tramite `read()` e `write()`

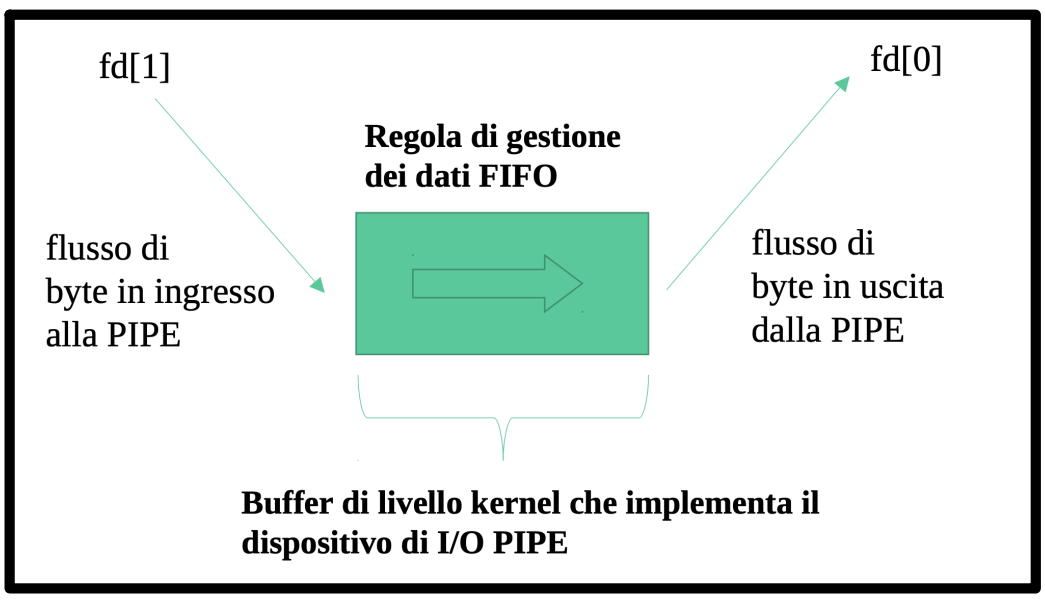
Quando creo una PIPE tipicamente viene modificata anche la mia tabella dei file descriptor, perché vengono cercate due entry da poter essere utilizzate sul primo estremo e sull'altro estremo. Questa tabella, quando io cerco di creare questa PIPE è già satura: QUESTO PUÒ ESSERE UN MOTIVO DI FALLIMENTO. Quindi non ha delle entry sufficienti libere per potermi permettere di acquisire le informazioni associate ai canali di I/O che mi servono per arrivare su questa PIPE.

Il parametro che noi passiamo è un'informazione che ci permette poi di acquisire i due canali per arrivare in I/O su questa PIPE.
Noi passiamo un array chiamato fd, quindi passiamo un indirizzo, puntiamo ad un intero che è composto da due elementi: l'area di memoria a cui questo puntatore che noi passiamo deve puntare, dovrà contenere due elementi interi.
Chiamiamo il servizio passando l'indirizzo di memoria di un'area dove possiamo scrivere due interi, in questo intero ci viene dato il codice del canale C, e in quest'altro ci viene dato il codice del canale C'.

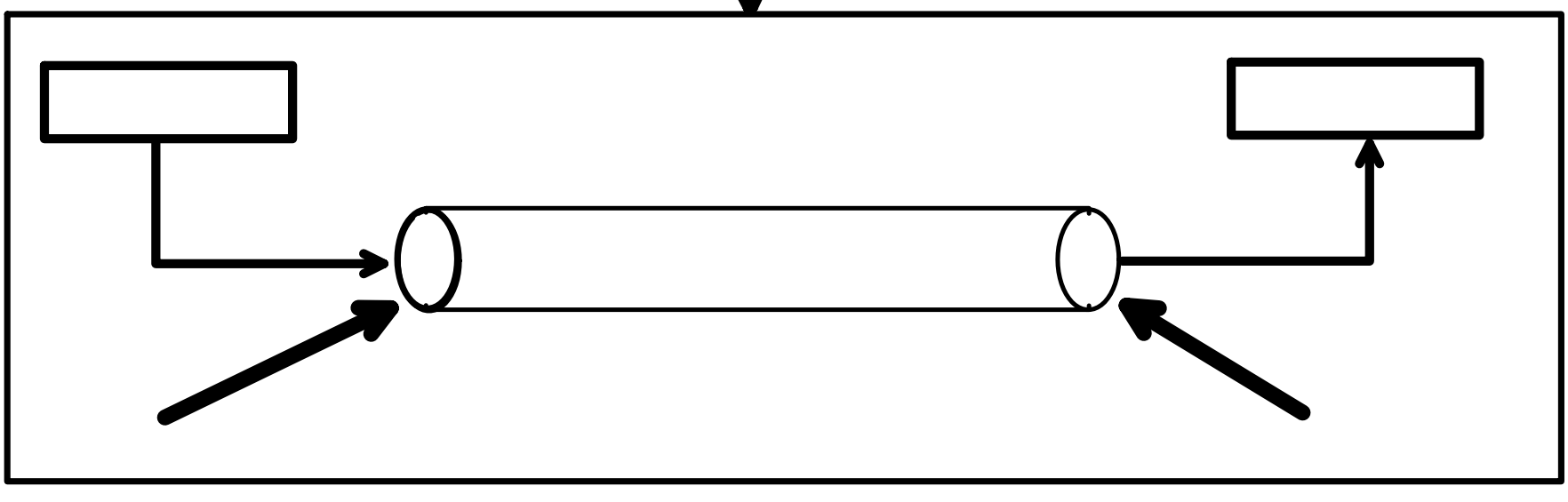


Questi sono i due canali che potremmo utilizzare per arrivare ovviamente a lavorare sulla PIPE.
Sono i canali che corrispondono alle entry della tabella dei file descriptor che possono essere utilizzate per andare a lavorare con questa PIPE.

Nel zeresimo elemento di questo array abbiamo un canale aperto in lettura, mentre il primo elemento ci porta sul canale di scrittura.
fd[0] ed fd[1] sono dei codici che ci ritornano per scrivere o leggere.



Nel momento in cui noi andiamo a cercare di scrivere dei dati all'interno della PIPE, nel nostro address space dobbiamo aver inserito questi dati all'interno di un'area e quando chiamiamo la write dobbiamo passare anche il puntatore a quest'area oltre che al canale che ci porta sull'estremo per andare ad inserire le informazioni nella PIPE.
Questi sono i due parametri che dobbiamo passare indicando ovviamente quanti di questi byte nell'area dovrebbero andare a finire all'interno della PIPE.
Stessa cosa sulla lettura i dati che vengono prelevati dalla pipe ci vengono consegnati in una specifica area di memoria di cui dobbiamo passare il puntatore quando chiamiamo l'operazione di read, oltre che il canale dell'estremo.



Avvertenze

- le PIPE non sono dispositivi fisici, ma logici
- la fine di uno “stream” su una PIPE è una condizione logica sullo stato della PIPE
- per convenzione, la fine dello stream viene rilevata quando la PIPE non ha più dati da consegnare e tutte le sessioni aperte in scrittura sono state chiuse
- tecnicamente, in questo caso la chiamata `read()` effettuata da un lettore restituisce zero
- allo stesso modo, se si tenta di scrivere sul descrittore fd[1] quando tutte le copie del descrittore fd[0] siano state chiuse (non ci sono lettori sulla PIPE), si riceve il “segnale SIGPIPE”, altrimenti detto Broken-pipe