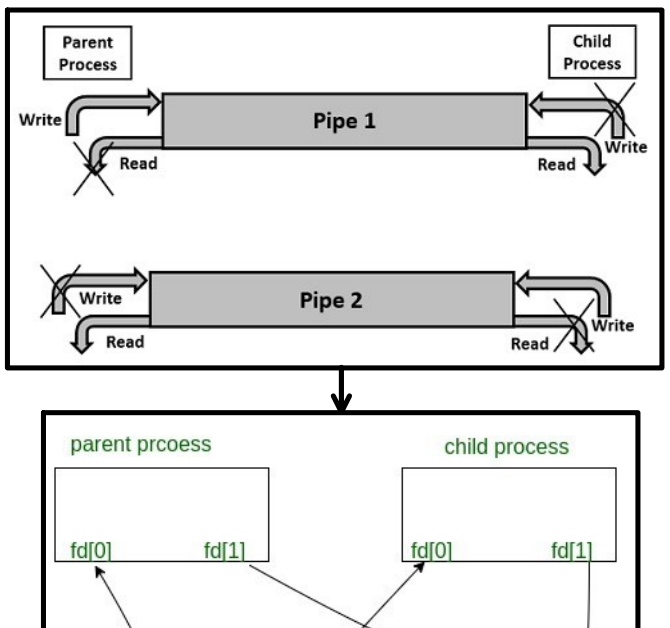
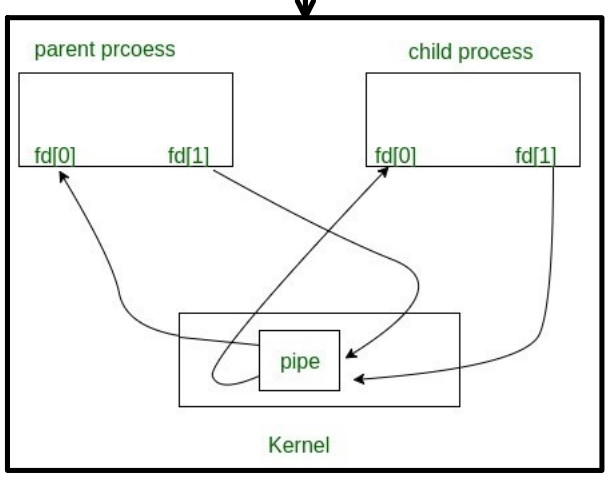
PIPES

martedì 2 maggio 2023 11:10

Concetti di base sulle PIPE

Per andare ad utilizzare le PIPE possiamo lavorare in I/O e l'I/O model che andiamo ad utilizzare è esattamente il modello stream. Quando noi abbiamo una PIPE come oggetto di I/O che serve per la comunicazione fra processi, possiamo immettere su di essa uno stream di byte - come se scrivessimo su un file - ed estrarre da questa PIPE uno stream di byte.

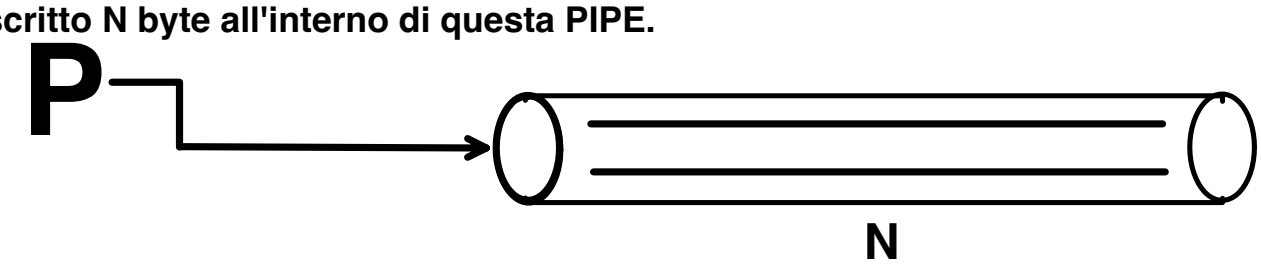




- permettono comunicare usando in I/O il modello stream
- il termine “pipe” significa tubo in Inglese, e la comunicazione avviene in modo monodirezionale
- una volta lette, le informazioni spariscono dalla PIPE e non possono più ripresentarsi, a meno che non vengano riscritte all'altra estremità
- a livello di sistema operativo, i PIPE non sono altro che buffer di dimensione più o meno grande (ad esempio 4096 byte)

Le pipe adottano una politica FIFO, perché se immetto un byte all'interno della PIPE, questo byte sarà il primo che verrà estratto dall'altro lato.
La PIPE rispetto ai file e alla gestione dei loro stream di byte, è un oggetto un po' più limitato: non ci permette di riposizionarci per rileggere delle informazioni.
Le informazioni possono essere scritte o lette ma non c'è possibilità di riposizionarsi in un punto qualsiasi per scartare alcune informazioni che sono presenti e muoversi solo su una zona di queste informazioni, ed inoltre non c'è la possibilità di muoversi per rileggere le stesse informazioni più di una volta.
Una pipe è un oggetto in cui noi possiamo andare, chiamando una system call, ad entrare e chiedere al kernel di registrare dei byte su di essa, questa PIPE a livello kernel è un'area di memoria gestita sotto il controllo del kernel stesso.
Quindi alla fine non sono altro che dei BUFFER di livello kernel che servono per poter consegnare al KERNEL dei dati e poter far sì che questi dati possano essere estratti/letti e quindi consegnati a qualche altro.

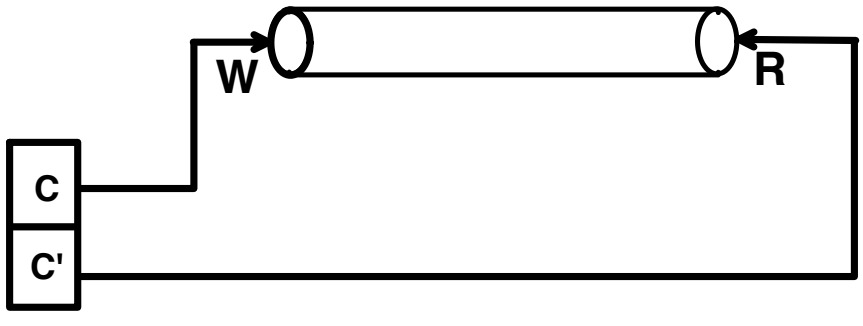
Supponiamo di avere una certa taglia N byte di un buffer di PIPE. E supponiamo di avere un processo che sta producendo delle informazioni che devono andare ad essere immesse all'interno di questa area di memoria. Questa area ne può ospitare al più N, e supponiamo che P abbia scritto N byte all'interno di questa PIPE.



Bene, questo tubo è pieno di informazioni che eventualmente poi potranno essere estratte dall'altro lato se si desidera leggerle. Ma nel momento in cui queste informazioni sono all'interno di questa PIPE stanno anche occupando lo spazio associato a questa area di memoria che abbiamo riservato per questo oggetto di I/O.
È possibile quindi che, quando il processo P vorrà cercare di inserire ulteriori dati all'interno di questa PIPE (chiamando un'operazione di I/O del vfs per andare a consegnare questi byte al kernel del sistema operativo) è possibile che questa operazione MANDI IN BLOCCO questo processo P. In particolare il thread lungo il quale viene ad essere chiamata questa operazione per andare ad inserire i dati all'interno di questa PIPE.
Le pipe avendo un'area limitata e anche piccola per quanto riguarda il mantenimento dei dati, ci portano eventualmente ai thread dei processi produttori ad essere bloccati quando chiamano i servizi di I/O per andare ad inserire nuovi dati all'interno della PIPE.
Possono usare questa PIPE soltanto i processi che sono relazionati tra di loro.

- i processi che usano PIPE devono essere “relazionati”

In realtà queste PIPE non hanno un nome: nel momento in cui creiamo una PIPE e quindi creiamo questo oggetto di I/O, nella nostra tabella dei canali validi ci vengono consegnati DUE CANALI, uno per arrivare sul lato iniziale ed uno per arrivare sul lato finale.



Uno serve per andare ad eseguire le Write all'interno di questo oggetto di informazioni gestito dal kernel, ed uno serve per eseguire le Read, ossia estrarre informazioni.

Questi due codici C e C' sono canali di I/O che possiamo utilizzare verso questo oggetto di I/O.
Ma questo oggetto di I/O non ha un nome.
Quindi nel momento in cui noi lo abbiamo creato e ci siamo fatti ridare questi due codici, nessun altro processo può utilizzare questa stessa PIPE eventualmente per parlare con me (processo creatore della PIPE), eccetto che se eredita questa tabella, quindi ha la possibilità anche lui di arrivare tramite questa tabella sui due estremi di questa PIPE.
L'ereditarietà di una tabella dei file descriptor all'interno di un sistema operativo UNIX, quindi in modo tale da creare un'altra tabella per un altro processo P', a partire da un processo P, si ottiene soltanto quando P' è il risultato di un'operazione di FORK() da parte di P.

Quindi nella tabella di P' abbiamo gli stessi canali validi della tabella di P.
Quindi possono arrivare sulla stessa PIPE che era stata creata dal processo genitore.
ENTRAMBI i processi arrivano sugli estremi di questa PIPE.
Questo limite che potremmo identificare nel permettere di parlare soltanto a processi relazionati di scambiare informazioni tramite una PIPE, viene risolto tramite le NAMED PIPE.

- le named PIPE (FIFO) permettono la comunicazione anche tra processi non relazionati

Sono delle PIPE a cui associamo anche un nome. Quindi quando creiamo questo oggetto di I/O - come quando creiamo un file - attribuiamo esattamente un nome a questo oggetto e avendo un nome ovviamente è possibile che altri processi possano utilizzare quel nome per poter chiamare il servizio di apertura di questa named pipe.
In questo modo possiamo ottenere i descrittori per arrivare sulla named pipe indipendentemente da se siamo processi relazionati rispetto al creatore della named pipe stessa.

- in sistemi UNIX l'uso della PIPE avviene attraverso la nozione di descrittore, in sistemi Windows attraverso la nozione di handle