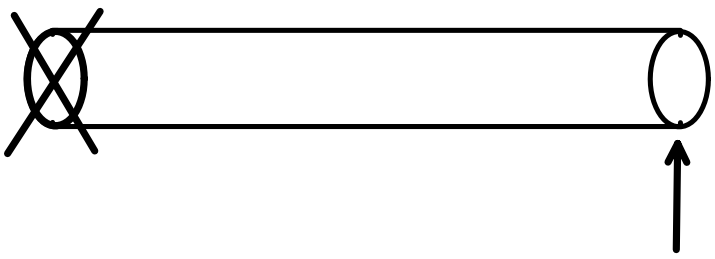
PIPE e stalli (deadlock)

Quello che può capitare su una PIPE è una situazione di stallo. Abbiamo un thread che è in esercizio sul sistema, questo thread chiama un servizio del kernel del sistema operativo per andare a lavorare su una pipe, in particolare lo stallo può essere causato quando noi cerchiamo di leggere dati da una pipe. Quindi questo servizio va sull'estremo a prelevare dati, abbiamo quindi chiamato una read e stiamo cercando di estrarre le informazioni dalla pipe.
Però attenzione, l'estremo di scrittura non è chiuso. Ma i dati da prelevare non ci sono.
Questo thread viene messo in stato di BLOCCO. Quindi non è più ready per riprendere il controllo della CPU perché è in attesa che alcuni dati possano arrivare su questa PIPE.
Supponiamo che questi dati, dentro a questa PIPE, li possa immettere solo questo thread - quindi questo è l'unico thread di un processo P che ha nella tabella dei file descriptor una entry all'estremo di lettura e una entry all'estremo di scrittura - e la entry di lettura è la stessa che viene usata nel servizio del kernel che avevamo chiamato prima. Ma questo stesso thread è l'unico thread di quest'unico processo che può scrivere su questa PIPE: ebbene questo thread è marcato per l'attesa indefinita. Sta cercando di estrarre i dati da una PIPE all'interno della quale "SOLO LUI" PUÒ INSERIRLI. Ma lui è bloccato per estrarre i dati E QUINDI non può inserirne.
Questo è un deadlock/stallo, ossia un attesa di un evento: c'è un lettore che è in attesa che i dati vengano scritti, ma l'unico che può scrivere quei dati è ancora il lettore, quindi quell'evento di scrittura non potrà mai capitare.
Quando noi abbiamo pipe e in uno scenario abbiamo un thread che vorrà solo estrarre i dati chiamando il kernel, per andare sull'estremo della PIPE a prelevare le informazioni, allora il processo per il quale questo thread vive, dovrà stare molto attento a chiudere l'estremo di inserimento (quello a sinistra di una PIPE).



In modo tale che non sia possibile avere nella tabella i due canali (uno per scrivere ed uno per leggere) validi, e quindi quando noi andiamo a chiamare una read per andare a leggere da quell'estremo di destra, il sistema operativo si potrà accorgere che lo stream associato a questa PIPE è del tutto completo, se non ci sono altri processi e quindi thread di altri processi che eventualmente possono produrre dati che possono arrivare su questa PIPE.

Queste sono operazioni che dobbiamo andare a fare in maniera esplicita; In particolare la chiusura di fd[1] nel momento in cui non siamo interessati a scrivere i dati ma siamo interessati solo a leggerli dalla PIPE, oppure chiudiamo fd[0] nel momento in cui siamo interessati solo a scrivere.

- per fare in modo che tutto funzioni correttamente e non si verifichino situazioni di deadlock è necessario che tutti i processi chiudano i descrittori di PIPE che non gli servono, usando una normale `close()`
- si noti che ogni processo lettore che erediti la coppia (fd[0],fd[1]) deve chiudere la propria copia di fd[1] prima di mettersi a leggere da fd[0] “dichiarando” così di non essere uno scrittore
- se così non facesse, l'evento “tutti gli scrittori hanno terminato” non potrebbe mai avvenire se il lettore è impegnato a leggere, e si potrebbe avere un deadlock