Pipe
E un canale di comunicazione unidirezionale.
Permettono le comunicazioni tra processi che sono in esecuzione sullo stesso sistema operativo.
Ricordano in qualche modo i socket ma son diversi.
Funzionano sempre come due capi (end-point) uno in scrittura uno in lettura, dato quindi una pipe possiamo rilevare un capo solo in scrittura e una solo in lettura.
Quando un'info viene letta, essa sparisce e non è più disponibile per essere letta da altri processi (si può implementare protocollo che permette di inviare stessa info a più capi - oppure che permettono di reinviare info in caso di perdite).

A livello di sistema operativo il pipe è come un buffer solitamente di 4K.
Abbiamo dei produttori e consumatori della pipe (buffer) - quindi viene scritto da produttori e consumato(letto) da consumatori.
I produttori anche in questo caso si bloccano se buffer pieno, consumatori si bloccano se buffer vuoto.

I thread all'interno di uno stesso processo non comunicano tramite pipe xk spesso hanno memorie condivise.
Le Pipe consentono di fare letture e scritture sfruttando i descrittori, son quindi disponibili read() e write().
Ci saranno nella comunicazione due descrittori = uno x scrittura , uno x lettura.

Se i processi son relazionati (la loro esecuzione è legata es. padre che fa fork a processo figlio) c'e' possibilità di passare descrittori in maniera automatica.
quindi se processo padre crea pipe, e poi fà fork (), il processo figlio avrà gia descrittori x comunicare tramite pipe.

Se invece i processi non sono relazionati per passare i descrittori si crea una named PIPE (gli si dà quindi un nome per poi passarla).
Si crea quindi con chiamata mkfifo - FIFO è un altro modo per chiamare le pipe e determina anche il suo carattere.
Che prende in parametri , il nome, il mode.
Anche in questo caso come quelle normali ,x eliminare una fifo definitvamente, una volta che tutti hanno chiuso comunicazione = si

chiama ****.

La chiamata che permette di creare pipe è pipe(int fd[2]).
Come parametro si aspetta un puntatore ad un array di dimensione almeno due.
In posizione 0 = descrittore in lettura
in posizione 1 = descrittore in scrittura

NB.
Quando si riceve 0 dalla read() su pipe significa che la comunicazione è terminata.
In particolare se ho più lettori la read ritorna 0 quando tutti i lettori hanno terminato.

Se tutti i lettori hanno chiuso comunicazione, e lo scrittore tenta di scrivere, riceve un segnale che gli notifca che tutti i lettori hanno chiuso la comunicazione.

Ipotesi.
Processo padre, crea pipe, riceve descrittore lettura e scrittura, poi fa fork.
Processo figlio ha anche lui i due descrittori.
Ammettiamo che padre è scrittore e figlio lettore, allora anche se padre chiude sua scrittura, quello del figlio rimane aperto = **DEADLOCK.**
Per questo c'e' la tecnica di chiudere i descrittori che nn servono prima di iniziare comunicazione.
Se devo leggere, chiudo quello in scrittura. Se devo scrivere, chiudo quello in lettura.