Il file descriptor è un'astrazione per permettere l'accesso ai file. È un numero intero che è salvato *all'interno di ogni processo* in una **tabella di file aperti del processo**: questa contiene una serie di *file descriptor* relativi ai file che il processo sta utilizzando. Questi puntano a loro volta ad una **tabella di sistema**, la quale contiene tutte le informazioni dei file utilizzati dai processi al momento operativi. Due processi possono avere *file descriptor* di ugual valore, ma puntano comunque a diversi file all'interno della tabella di sistema.

Quando un qualsiasi programma entra in esecuzione, l'ambiente del sistema operativo apre automaticamente **3 flussi di input/output standard**:

- ***Standard input (stdin)***: Utilizzato per l'input di dati, di default da tastiera. Il numero di *file descriptor* utilizzato è **0**.
- ***Standard output (stdout)***: Utilizzato per l'output di dati, di default sulla linea *tty* al momento in uso. Il numero di *file descriptor* utilizzato è **1**.
- ***Standard error (stderr)***: Utilizzato per l'output di messaggi di errore, di default sulla linea *tty* al momento in uso. Il numero di *file descriptor* utilizzato è **1**.

In particolare, ciascuno di questi flussi di dato ottiene i dati in maniera differente:

- Generano dati in *standard output* i comandi `echo` e `printf` del linguaggio `bash` e `printf` e `fprintf` del linguaggio C.
- Generano dati in *standard error* gli errori dei comandi in `bash` e `fprintf` del linguaggio C.
- Generano dati in *standard input* il comando `read` del linguaggio `bash` e i comandi `fgets` e `scanf` del linguaggio C.

Nota fondamentale dei file descriptor è la loro ereditarietà alle *subshell*: **una subshell B figlia di una *shell* A riceve una copia della tabella dei file aperti in A**. Ciò significa che *shell* padre e *shell* figlia **hanno stream di scrittura e lettura comuni**: questo significa per esempio che l'output della *shell* figlia è leggibile direttamente da quella padre.

Command substitution

La ***command substitution*** viene usata per inserire all'interno dell'output in *standard output* di un programma l'output in *standard output* di un altro programma.
Essa avviene tramite il *quoting* tra **backticks**: `.

```
# Sia primo.exe un programma che a un certo punto restituisca la stringa
"Ciao!" in standard output.

OUT=`./primo.exe`
```

```
echo "L'output del processo e' ${OUT}"
# Output: L'output del processo e' Ciao!
```

oppure tramite il costrutto `$()`.

In entrambe le sintassi le wildcards **vengono valutate**. Per disabilitare la loro valutazione, la riga di comando va opportunamente circondata da caratteri opportuni. Esistono due metodologie di quoting:

- ***Quoting a doppi apici*** `""`: Il quoting a doppi apici non permette l'interpretazione del metacarattere separatore `;` e delle *wildcards*, ma permette **variable e command substitution**.
- ***Quoting a singoli apici*** `''`: La stringa quotata tra singoli apici viene interpretata così com'è, senza effettuare operazioni di sostituzione o espansioni di wildcards.

```
VAR=Ciao!
echo "${VAR} Come stai? `ls`" # Output: Ciao! Come stai? <Lista di file>
echo '${VAR} Come stai? `ls`' # Output: ${VAR} Come stai? `ls`
```