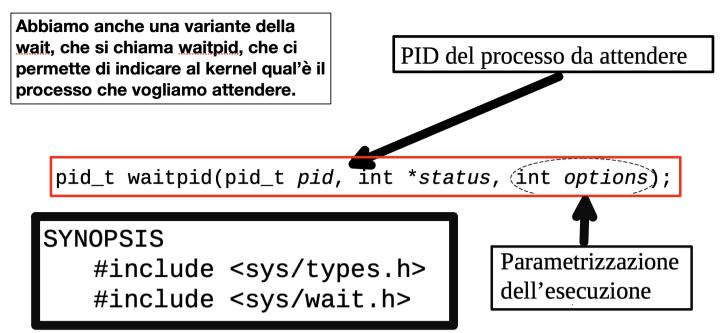
giovedì 30 marzo 2023 15:25

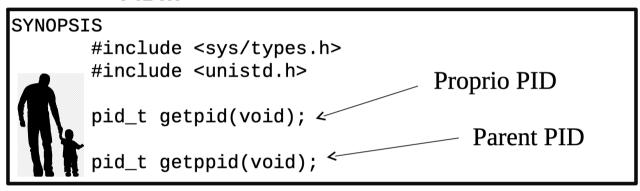
ALTRE FUNZIONI



Mentre invece nella WAIT noi diciamo che vogliamo attendere UNO QUALSIASI dei nostri CHILD.

pid\_t wait(int \*status);

Altre due funzioni che lavorano con il PID...



## **ESEMPIO**

```
tinclude<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
int main(int a, char **b){
                          Abbiamo un main, abbiamo una
       int exit_code;
                          chiamata ad una fork() e ovviamente ci
       pid = fork();
                          chiediamo chi siamo.
       if (pid == 0){
               printf("child process querying parent id is %d\n",getppid());
               if (getppid()==1) exit(0);
               sleep(3);
               goto redo;
       sleep(1);
                       Il padre viene eseguito subito, e con
       exit(0);
                       exit(0) significa che sta completando
                       a sua esecuzione.
```

Se siamo il Child eseguiamo il blocco di codice che è all'interno di questo IF. All'interno il figlio chiede quale sia suo padre.

Se il Pid del padre è uguale ad 1, esco con exit(0).

Il figlio avrà sempre un solo padre. Il padre ha un solo figlio in questo caso. Ma se il padre termina la sua esecuzione?

Il figlio non potrà mai sapere la corretta indicazione del processo padre, getppid(), se il padre è morto.

Ma noi abbiamo il Child ancora attivo, che si sta chiedendo nel GOTO, qual'è il PID del parent.

```
OCESSES-AND-THREADS/UNIX> ./a.out
child process querying parent id is 25384
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING
OCESSES-AND-THREADS/UNIX> child process querying parent id is 1
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING
OCESSES-AND-THREADS/UNIX>
```

1 è un processo che diventa padre di tutti gli altri processi quando questi non hanno più padre. Quindi il child può tranquillamente andare avanti.