

ALTRE FUNZIONI

Abbiamo anche una variante della `wait`, che si chiama `waitpid`, che ci permette di indicare al kernel qual'è il processo che vogliamo attendere.

PID del processo da attendere

```
pid_t waitpid(pid_t pid, int *status, int options);
```

SYNOPSIS

```
#include <sys/types.h>
#include <sys/wait.h>
```

Parametrizzazione dell'esecuzione

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...

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>
```



```
pid_t getpid(void);
```

Proprio PID

```
pid_t getppid(void);
```

Parent PID

ESEMPIO

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>

int main(int a, char **b){

    pid_t pid;
    int exit_code;

    pid = fork();

    if (pid == 0){
        redo:
        printf("child process querying parent id is %d\n",getppid());
        if (getppid()==1) exit(0);
        sleep(3);
        goto redo;
    }

    sleep(1);
    exit(0);
}
```

Abbiamo un `main`, abbiamo una chiamata ad una `fork()` e ovviamente ci chiediamo chi siamo.

Il padre viene eseguito subito, e con `exit(0)` significa che sta completando la 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`.

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

