```c
/*
   Esempio di fork. Un certo numero di fork, e sia il figlio che il
   padre non fanno altro che scrivere i due ID. Il padre aspetta i
   figli "tutti insieme" dopo le fork, in un loop in cui usa wait
   (generica). Quindi, anche se i figli "ritardano" non succede nulla
   di male!  Le due opzioni -c e -p permettono di scegliere di quanto
   "ritardare" l'esecuzione della printf nei figli e nel padre; il
   default è zero per tutti.
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>             /* fork */
#include "fun.h"

int main(int argc, char *argv[])
{
  int count = 3;

  pid_t pid;
  int i;
  int sleep_child = 0;
  int sleep_parent = 0;

  get_opt(argc, argv,
          &sleep_child, &sleep_parent,
          NULL);

  for (i=0; i<count; i++) {

    pid = fork();
    if (pid == 0) {

      sleep(sleep_child);
      printf("child:  %d with parent: %d\n",
             getpid(), getppid());
      return;
    }
  }
  for (i=0; i<count; i++) {

    pid_t pid;
    int status;
    pid = wait(&status);
    printf("parent: %d with child:  %d\n",
           getpid(), pid);
  }
  return 0;
}
```