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

/* Maximum number of children */
#define MAX_NO_CLD 10

int main ()
{
   int i, ncld, wtime, status;
   pid_t cpid, mypid, parpid;

   /* Parent process gets its own ID and its parent's ID */
   mypid = getpid();
   parpid = getppid();
   printf("Parent: PID = %u, PPID = %u\n", mypid, parpid);

   /* Parent process obtains the number of children from the user */
   printf("Parent: Number of children = "); scanf("%d", &ncld);
   if ((ncld < 0) || (ncld > MAX_NO_CLD)) ncld = 5;
   printf("\n");

   /* Child creation loop */
   for (i=0; i<ncld; ++i) {
      /* Create the next child */
      cpid = fork();

      /* The child process executes the following conditional block */
      if (cpid == 0) {
         /* Child process gets its own ID and its parent's ID */
         mypid = getpid();
         parpid = getppid();
         srand(mypid);
         wtime = 1 + rand() % 10;
         printf("Child %d: PID = %u, PPID = %u, work time = %d\n",
                 i, mypid, parpid, wtime);

         /* Child process does some work (sleeping is a hard work!) */
         sleep(wtime);
         printf("\nChild %d: Work done...\n", i);

         /* Child process exits with status i (the loop variable) */
         exit(i);
      }

      /* The parent continues to the next iteration */
   }

   /* Parent waits for all the children to terminate */
   for (i=0; i<ncld; ++i) {
      /* Parent waits for any child to terminate */
      /* Use waitpid() if the wait is on a specific child */
      wait(&status);

      /* Parent uses the exit status to identify the child */
```

```
      printf("Parent: Child %d terminates...\n", WEXITSTATUS(status));
   }

   printf("\nParent terminates...\n");
   exit(0);
}
```