Partie 3 : Signaux
Le corrigé

Solution 1

```
/*0*/
void handler(int sig){
        if (sig == SIGCHLD) wait(NULL);
        }
int main(int argc, char *argv[])
{
        /*1*/
        signal(SIGCHLD, handler);
        if (!fork())
        {
                /*2*/
                for (int i = 0 ; i <10 ; i++) ; //simule un petit calcul
                /*3*/
                exit(1) ;
                /*4*/
        }
        /*5*/
        while(1) ; //Simule un calcul infini
        /*6*/
}
```

Solution 2

Le processus doit capter le signal SIGPIPE. L'instruction « signal (SIGPIPE, action) ; » permet au processus de capter ce signal. Cette instruction doit précéder toutes les opérations d'écritures dans les pipes.

Le gestionnaire du signal « action » doit comporter le message d'erreur à afficher et se terminer par un exit.

Pour afficher le descripteur, il faut définir une variable globale par exemple « int CurrentFd ». Cette variable sera utilisée dans la fonction action.

Chaque opération d'écriture dans un pipe doit être précédée de l'affectation du descripteur d'écriture dans CurrentFd.

Solution 3

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include<stdio.h>
#include <signal.h>
#define N 5
/*0*/  void action(int signum) {  printf("capture de SIGCONT %d\n", getpid() );}
int main( )
{       pid_t pid[N];
        int i ;
        /*1*/    signal(SIGCONT,action);
        for ( i=0; i<N;i++)
            if ((pid[i]=fork())==0)
            {    /*2*/  pause() ;
                 while(1)
                     printf("ici fils %d ", i);
            }

       /*3*/    while (1)
                 for(i=0; i<N;i++)
                 { kill(pid[i], SIGCONT); sleep(1); kill(pid[i], SIGSTOP); }
}
```