PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Semestre 2020-1
Laboratorio 1

Todos los programas base que necesita lo puede copiar a su *home*, con el siguiente comando:

```
cp /home/ladmin/Lab1449/lab1material.zip  .
```

1) (7 puntos) El siguiente código genera un abanico de procesos.

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

/* Este programa crea un abanico de procsos.                    */
/* Ejm 2.6 del libro UNIX Programacion Practica - Kay Robbins    */
/*                                                Steve Robbins  */

#define   N    8

int main(void)
{ int i,status;
  pid_t child,pid_padre;

  pid_padre=getpid();
  for(i=0;i<N; ++i)
     if((child=fork())<=0) break;
  if(pid_padre==getpid()) for(i=0;i<N;++i) wait(&status);
  return 0;
}
```

Modifique el código para que el último proceso creado por el padre, elimine a todos los procesos hermanos, y también al proceso padre, enviando la señal `SIGKILL`.

El proceso “asesino”, antes de proceder con su tarea, debe de ejecutar `ps -o pid,ppid,cmd` con la función de librería `system()`. Y después de terminar su trabajo, debe volver a ejecutar la misma función. De forma que se muestre que al inicio hay 8 procesos y que al final hay 1. Y este, es el último proceso creado. La salida debe ser análoga a la siguiente:

```
  PID  PPID CMD
 8670  8664 /bin/bash
23778  8670 ./abanico
23779 23778 ./abanico
23780 23778 ./abanico
23781 23778 ./abanico
23782 23778 ./abanico
23783 23778 ./abanico
23784 23778 ./abanico
23785 23778 ./abanico
23786 23778 ./abanico
23787 23786 sh -c ps -o pid,ppid,cmd
23788 23787 ps -o pid,ppid,cmd
```

```
  PID  PPID CMD
 8670  8664 /bin/bash
23786 23252 ./abanico
23792 23786 sh -c ps -o pid,ppid,cmd
23793 23792 ps -o pid,ppid,cmd
```

Importante: No puede usar pipes.

2) (6 puntos) A continuación se tiene el siguiente programa:

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

int main(void) {
    if(!fork()) {
           char cadena3[]="UNIVERSIDAD";
           char cadena4[]="CATOLICA";
           char cadena5[]="PERU";

           printf("%s ",cadena3);
           printf("%s ",cadena4);
           printf("%s ",cadena5);
           exit(0);
    } else {
         char cadena1[]="PONTIFICIA";
         char cadena2[]="DEL";
         printf("%s ",cadena1);
         printf("%s ",cadena2);
    }
    exit(0);
}
```

Al ejecutarlo la salida es la siguiente:

```
PONTIFICIA DEL UNIVERSIDAD CATOLICA PERU
```

El objetivo es que se imprima el mensaje correctamente:

```
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
```

Para lograr su objetivo usted debe de intercambiar los mensajes usando *pipes*. La condición es que **el padre debe imprimir 3 cadenas y el hijo 2.** Además no se debe modificar la definición de las cadenas. Solo puede intercambiar las cadenas usando *pipes*.

3) (7 puntos) La conjetura de Collatz. Esta conjetura afirma que para cualquier entero mayor que 0, si se sigue la siguiente regla:

- Si el número es par, se divide entre 2.
- Si el número es impar, se multiplica por 3 y se suma 1.

La conjetura afirma que siempre alcanzaremos el número 1.

A continuación se ha implementado un programa empleando *pipes*. Donde un hijo realiza la tarea cuando el número es par, y el otro cuando el número es impar.

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

int main(void) {
   int n;
   int fdpares[2], fdimpares[2];

   srand(time(NULL));
   n = (rand() % 30)+1;
   pipe(fdpares);
   pipe(fdimpares);
   if(n & 1) write(fdimpares[1],&n,sizeof(n));
   else write(fdpares[1],&n,sizeof(n));
   if(!fork()) {          /******  Inicio de hijo 1 ****/
      while(1) {
         read(fdpares[0],&n,sizeof(n));
         while(!(n & 1)) {
             fprintf(stderr,"%d\n",n);
             n = n / 2;
          }
          write(fdimpares[1],&n,sizeof(n));
          if(n==1) exit(0);
      }
   }                              /******* Fin de hijo 1 *****/
   if(!fork()) {                  /******  Inicio de hijo 2 ****/

          while(1) {
              read(fdimpares[0],&n,sizeof(n));
              if(n==1){
                  fprintf(stderr,"%d\n",n);
                  return 0;
               }
              else {
                   fprintf(stderr,"%d\n",n);
                   n = 3*n +1;
                  write(fdpares[1],&n,sizeof(n));
              }
          }
    }                              /******* Fin de hijo 2 *****/
    wait(NULL);
    wait(NULL);
    return 0;
}
```

Al ejecutar el programa, una de las salidas es:

```
7
22
11
34
17
52
26
13
40
20
10
5
```

```
16
8
4
2
1
```

Se le solicita que el código tanto de `hijo 1` como de `hijo 2` deben ser compilados como programas independientes (deben contener la función `main()`). El primer programa debe tener nombre *pares.c* y el segundo *impares.c*.

Usted debe hacer las modificaciones necesarias de tal forma que en lugar del código del `hijo 1` se sustituya por el ejecutable de *pares.c* y en lugar del código de `hijo 2`, se sustituya por el ejecutable de *impares.c*. En ambos casos usando la función de librería `execl()`.

Habrá logrado su objetivo si el resultado es el mismo.

Porf. Alejandro T. Bello Ruiz