

IPC Y SEÑALES EN LINUX

Tuberías sin nombre

Ingresa al sistema como **usuario no privilegiado**. Cree un directorio donde almacenar los programas de este práctico dentro de su directorio HOME. Luego sitúese en el directorio recién creado.

Escriba el siguiente programa y nómbrelo **"prog_0701.pp"** y compílelo.

```
Program prog_0701;

Uses BaseUnix, Unix;
Var atras, frente: Text;
    s: String;
begin
  Writeln('Creando Pipe.');
```

if assignpipe(frente, atras)<>0 then
 Writeln('Error creando pipes! ' , fpgeterrno);
if fpfork()>0 then {padre}
begin
 Writeln('(P) Escribiendo en el PIPE.');

Writeln(atras, 'Hola hijo!');

Close(atras); {el close cierra el pipe}
WaitProcess(0);{Esperando a cualquiera de hijo para
terminar}
Writeln('(P) Terminando...');

end
else begin {hijo}
Writeln('(C) Leyendo el PIPE...');

Readln(frente, s);
Close(frente);
Writeln('(C) Leido: ', s);
Writeln('(C) Terminando...');

end;
end.

(1) Ejecute el programa y observe su comportamiento. Explique que hace y cómo lo hace.

Escriba el siguiente programa y nómbrelo “**prog_0702.pp**” y compílelo.

```
Program prog_0702;

Uses BaseUnix, Unix, crt;
Var atras, frente: Text;
    s: String;
    i: Integer;
begin
  Writeln('Creando Pipe.');
```

```
  if AssignPipe(frente, atras)<>0 then
    Writeln('Error creando pipes! ', fpgeterrno);
  if fpfork()>0 then begin {Padre == Productor}
    for i:=1 to 10 do begin
      Writeln('(P) Escribiendo ',i ,' en el PIPE.');
```

```
      Writeln(atras, i);
      Delay(Random(1000));
    end;
    Writeln(atras, '-1');
    WaitProcess(0); {Espera al hijo para terminar}
    Writeln('(P) Terminando...');
```

```
    Close(frente);Close(atras);
  end
  else begin {Hijo == Consumidor}
    Writeln('(C) Leyendo el PIPE...');
```

```
    repeat
      Delay(Random(1500));
      ReadLn(frente, s);
      Writeln('(C) Leido: ', s);
    until s='-1';
    Writeln('(C) Terminando...');
```

```
    Close(frente);
  end;
end.
```

(2) Ejecute el programa y observe su comportamiento. Explique que hace y cómo lo hace.

Manejo de señales

La forma mas primitiva de coordinación y comunicación entre procesos es el envío de señales. Los lenguajes de programación en general cuentan con primitivas POSIX para enviar y recibir señales.

Para esto se emplean (en Freepascal) las funciones de la unit BaseUnix:

| | |
|--------------|---|
| fpAlarm | Envía una señal de alarma a si mismo. |
| fpKill | Envía una señal arbitraria a un proceso. |
| fpPause | Espera hasta la llegada de una señal. |
| fpSignal | Establece la acción a tomar ante la llegada de una señal. |
| fpSigPending | Ve si hay señales pendientes. |
| fpSigRaise | Envía una señal a si mismo. |

Las señales más utilizadas son:

| | |
|----------------|--|
| SIGHUP | se detecta que se colgó la terminal o el proceso asociado. |
| SIGKILL | señal de terminación incondicional. |
| SIGALRM | señal del reloj desde alarm(2). |
| SIGTERM | señal de terminación. |
| SIGUSR1 | señal definida por el usuario 1. |
| SIGUSR2 | señal definida por el usuario 2. |
| SIGCHLD | proceso hijo terminado o detenido. |
| SIGCONT | continúa si estaba detenido. |
| SIGSTOP | detiene el proceso. |

Para ver otras señales utilice las paginas man (**man 7 signal**).

Además de poder enviar señales desde los procesos los usuarios pueden enviar señales a un proceso determinado.

Para esto se emplea el comando kill:

```
kill -SIGNAL PID
```

Donde **SIGNAL** es la señal a enviar y **PID** es el process ID del proceso al que queremos enviar dicha la señal.

Si no se especifica ninguna señal, se envía por defecto la señal **SIGTERM**.

Escriba el siguiente programa y nómbrelo "**prog_0703.pp**" y compílelo.

```
Program prog_0703;  
  
uses unix, baseunix;  
  
{Defino un procedimiento para atender a las seniales.}  
Procedure MuestraSenial(sig : Longint);cdecl;  
begin  
    writeln('Recibiendo senial: ',sig);  
end;  
  
begin  
{Establezco la accion a realizar al recibir SIGUSR1.}  
    fpSignal(SigUsr1,@MuestraSenial);  
{Espero a que presione Enter para salir..}  
    writeln ('Envie senial USR1 o <ENTER> para salir');  
    writeln ( ' (ejecute kill -SIGUSR1 ',fpgetpid,' )');  
    readln;  
end.
```

(3) Ejecute el programa y desde otra consola envíe la señal. Observe los resultados en la consola donde se está ejecutando el programa. Por qué nos muestra "Recibiendo senial: 10" en lugar de "Recibiendo senial: SIGUSR1"?

Las paginas man de kill(1) y de signal(7) pueden ser de gran ayuda para responder a la pregunta anterior.

(4) Ahora desde la misma consola envíe el mensaje **SIGUSR2**. Qué ocurre? Por qué no nos muestra el cartel "Recibiendo señal..."?

(5) Cree una nueva versión de este programa que muestre también la señal **USR2**. Guárdelo como "**prog_0704.pp**". Compílelo y verifique que funciona.

Escriba el siguiente programa y nómbrelo “**prog_0705.pp**” y compílelo.

```
Program prog_0705;

uses BaseUnix;

var
    proceso: LongInt;
    senial: Integer;

begin
    Write('Proceso destino: ');
    Readln(proceso);
    Write('Senial a enviar: ');
    Readln(senial);
    kill(proceso, senial);
end.
```

En otra consola ejecute el programa **prog_0703** y emplee el programa recién creado para enviar la señal.

(6) Desarrolle su propia versión del comando **kill** y guárdelo con el nombre “**prog_0706.pp**”.

Éste deberá cumplir con los siguientes requisitos:

- Tomar los parámetros de la línea de comandos.
- Solamente aceptar señales numéricas (para mantener la sencillez).
- No se podrá omitir la señal a enviar (siempre recibirá dos parámetros).

Pruebe el programa recién creado, por ejemplo, usándolo para desarrollar la actividad (1). Tenga en cuenta que **SIGUSR1** es 10 y **SIGUSR2** es 12.

Escriba el siguiente programa y nómbrelo “**prog_0707.pp**” y compílelo.

```
Program prog_0707;

uses BaseUnix;

Procedure MuestraSenial(sig : Longint);cdecl;
begin
    writeln('Recibiendo senial: ', sig);
end;

begin
    fpSignal(SigUsr1, @MuestraSenial);
    Writeln ('Envie senial USR1 para salir');
    Writeln (' (ejecute kill -SIGUSR1 ', fpgetpid, ' )');
    {Queda esperando que llegue una senial...}
    fpPause;
    Writeln('Terminando...');
end.
```

Note que ya no usamos **Readln** para que el programa se detenga hasta recibir la señal.

(7) Ejecute el programa y desde otra consola envíele la señal **SIGUSR1**. Vuelva a ejecutarlo y esta vez envíele la señal **SIGUSR2**. Por qué sale aunque no le mandemos la señal **SIGUSR1**? A qué se deben las diferencias en las salidas del programa?

(8) Desarrolle un programa que cree dos procesos (usando **fork()**), uno de los cuales envía una señal que el otro esta esperando, cuando el segundo recibe la señal muestra un cartel y entonces terminan ambos. Guárdelo con el nombre "**prog_0708.pp**".

Confeccione un informe **original, conciso y completo** donde se dé respuesta a las preguntas y consignas precedidas por un número encerrado entre paréntesis. Éste informe deberá ser confeccionado y entregado por cada grupo que llevó a cabo las actividades.
La longitud máxima del informe es de dos páginas (sin contar las líneas correspondientes a código fuente).