Zombies

MSc. Ivan A. Escobar Broitman

Procesos zombies en servidores

- Proceso que no tiene recursos asignados, pero que ocupa un lugar en la tabla de procesos
- No es posible matarlo
- Dos formas de prevenirlos:
  - ignorando la señal (Sistema V)
  - atrapando la señal (BSD)

Ejemplo procesos zombies

```
iescobar@localhost:25> ps -l
PID    PPID    TT      S       TIME  COMMAND
  :      :       :      :       :       :
  :      :       :      :       :       :
5525  3660   ----    IW     1:41     server
5527  5525   ----     Z      0:00  <defunct>
5529  5525   ----     Z      0:00  <defunct>
5531  5525   ----     Z      0:00  <defunct>
5537  36060 ----    S       0:16       tcsh
iescobar@localhost:26>
```

Características Zombies

- Todos tienen el mismo padre
- Aunque los procesos han terminado (exited), poseen una pequeña pieza de información: su status de salida
- Esperan pasarle dicho status a su padre
- Usualmente:
  - padre ejecuta un wait() para sincronizarse con la terminación del hijo
  - el padre muere, y el estatus del hijo es recuperado por el “abuelo” (i.e. shell o init)

Ignorando la señal
(SVR4 UNIX)

- Necesario modificar el código del padre
- Ignorar la señal con el parámetro SIG_IGN de la llamada signal()
- Ejemplo:

```
#include <signal.h>
signal(SIGCHLD, SIG_IGN);
```

El código del servidor

```
#include <signal.h>
main( )
{
    signal(SIGCHLD, SIG_IGN);

    sd = socket(AF_UNIX, SOCK_STREAM, 0 );
    bind(sd, (struct sockaddr *)&saddr, addrlen );
    listen(sd, 5);
    while(1)  {
        fd = accept(sd, 0, 0 );
        if ( fork() = = 0 ) {          /*código proceso hijo */
            atencion_servicio(fd);
            exit(0);
        }
       else
            close(fd);                 /* padre no usa conexión */
     }
}
```

Comentarios solución SVR4

- Disposición por default de la señal SIGCHLD es SIG_IGN
  - la llamada no tiene sentido
- Es la forma usual (y documentada) de prevenir la formación de zombies, bajo las circunstancias descritas

Previniendo zombies en BSD

- En sistemas Unix derivados de BSD, la solución anterior no funciona
- Una solución más compleja es necesaria:
  - atrapar la señal SIGCHLD
  - ejecutar una llamada wait()

Primera opción solución

```
#include <signal.h>
void espera()
{
   wait(0);
   signal(SIGCHLD, espera);
}

main( )
{
    signal(SIGCHLD, espera);

    sd = socket(AF_UNIX, SOCK_STREAM, 0 );
    bind(sd, (struct sockaddr *)&saddr, addrlen );
    listen(sd, 5);
    while(1)  {
        fd = accept(sd, 0, 0 );
        if ( fork() = = 0 ) {          /*código proceso hijo */
                :
                :
```

Precauciones a tomar

- Existen llamadas (read(), accept() y select()) que pueden ser interrumpidas si una señal es entregada
- Si esto sucede:
  - llamada regresa un aparente error con el valor EINTR asignado a errno
  - en un servidor el proceso padre estará bloqueado cuando la señal llegue
  - es necesario modificar la llamada para que atrape el regreso de error y restablecer la llamada accept()

Código de la solución

```
#include <signal.h>
        :
        :
void espera()
{
   wait(0);
   signal(SIGCHLD, espera);
}
        :
        :
        :
```

```
main( )
{
    signal(SIGCHLD, espera);
                :
                :
     while (1) {
        ctl_len = sizeof(client);
        DENUEVO:   msgsock= accept(sock, 0, 0)
       if ( msgsock < 0 ) {
          if ( errno = EINTR )    /* accept() interrumpido por la señal */
             goto DENUEVO;   /* por lo que hay intentar de nuevo   */
          else  {
             perror("accept");
             exit(3);
           }
       }

      if ( fork() = = 0 ) {          /*código proceso hijo */
         atencion_servicio(fd);
```

2

2

```
            :
            :
            :
if  ( fork() = = 0 ) {
    close(sock);
    servicio(msgsock)
    exit(0);
  }
  else
     close(msgsock);
}                                   /* del if ( msgsock < 0 )  */
```