Signalling (IPC)

Signalling (segnalazione)

è un particolare tipo di Inter-Process Communications che permettere di sincronizzare dei processi spedendo e ricevendo dei messaggi minimali che informano dell'accadimento di un evento (ad esempio, morte di un processo figlio).

Alcuni segnali sono per uso predefinito, ad esempio per ordinare ad un altro processo di terminare (kill -9 pid).

Alcuni segnali (SIGUSR1 e SIGUSR2) sono a disposizione dell'utente.

Attesa terminazione di processo

Un processo padre può generare un processo figlio, farlo eseguire in background poi aspettare fino a che il processo figlio termina.
Per fare questo il processo può invocare
- il comando wait (se il padre è una shell di comandi,
- la chiamata a funzione waitpid (se il processo è scritto in linguaggio C.

In entrambi i casi, comando o funzione invocano una system call che aspetta la terminazione del processo figli.

Abbiamo ora quattro dubbi:

1. Perché un processo padre potrebbe decidere di non chiamare la waitpid?

2. Cosa accade se il processo padre continua a eseguire e non fa mail la waitpid?

3. Cosa accade se il processo padre termina senza avere mai chiamato waitpid?

4. Se il processo padre esegue per un lungo periodo e
   - vuole evitare di lasciare in giro dei figli zombie,
   - ma **vuole anche evitare di bloccarsi in attesa della terminazione dei figli**,

   come può fare?
   signal !!!

wait, processi Zombie, processi Orfani, processo init (1)

Un processo (padre) genera un processo figlio e lo esegue in background.
Il processo figlio prima o poi **termina** e restituisce un valore intero.
Il processo padre vuole sapere quale è il valore restituito dal figlio per sapere se tutto è andato bene o no. A questo scopo, il processo padre esegue il comando **wait pidfiglio** che attende la terminazione del processo figlio avente pid pidfiglio e restituisce il risultato del processo figlio.

1. **Se il processo figlio termina prima del processo padre**, il sistema operativo rilascia le risorse occupate dal processo figlio **ma mantiene nelle sue tabelle una struttura pcb (process control block) con una descrizione del processo terminato**. Questa struttura conserva anche il **pid** e il **risultato** del processo figlio. In questo momento il processo figlio è uno **zombie** poiché è terminato ma la sua struttura descrittiva è ancora presente nelle tabelle del sistema operativo.
   - **La struttura pcb viene eliminata solo quando il padre invoca il comando wait** (o una system call waitpid che fa la stessa cosa) per attendere la terminazione del figlio. A quel punto il processo figlio sparisce dalle tabelle e il suo pid può essere riutilizzato per altri processi.
2. **Se invece il processo padre termina senza fare la wait per il figlio, il processo figlio diventa orfano**. Quando un processo orfano termina, oppure quando uno zombie diventa orfano perché termina il padre, **l'orfano viene adottato dal processo init**, quello con pid 0 che ha originato tutti i processi al boot. Gli orfani diventano figli di init. **Ogni tanto il processo init chiama la wait sui propri figli**, anche quelli adottati ovviamente, **e così fa rilasciare i pcb degli orfani**.
   - Quindi, **morto il padre, i figli che terminano** vengono eliminati (**no zombie**)

Processi Zombie, Processi Orfani, processo init (2)

Per quale motivo un processo padre non fa una wait per un figlio?

1. o per sbaglio, perché il programmatore se ne è dimenticato,

2. oppure appositamente, per impedire che un altro suo nuovo figlio prenda dal sistema operativo proprio il pid del processo figlio terminato.

In entrambi i casi, si rischia di esaurire i pid disponibili nel sistema operativo.

Processi Zombie e signal SIGCHLD

Se un processo padre vuole rilasciare i figli zombie senza bloccarsi a fare una wait, può scrivere una funzione che faccia automaticamente la waitpid quando un figlio termina.
Infatti, alla terminazione, ogni processo invia un segnale detto SIGCHLD al proprio padre per informarlo che il figlio è terminato.
Il processo padre può ordinare al sistema operativo di eseguire una funzione del padre quando al padre arriva questo segnale. La funzione viene eseguita dal padre. Il programmatore in questa funzione chiama la waitpid per far rilasciare il pcb del figlio terminato.

```
void dezombizzaFigli(int signum) {
    pid_t pid;    int saved_errno = errno;
    while( waitpid((pid_t)(-1),0,WNOHANG)>0)
        { ; }
    errno = saved_errno;
}
void register_SIGCHLD_sighandler(void) {
    struct sigaction sa;
    sa.sa_handler = &dezombizzaFigli;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_NOCLDSTOP;
    if (sigaction ( SIGCHLD, &sa, 0) == -1)
      { perror("sigaction"); exit(1); }
}
```

```
int main( void )
{
    pid_t pid;  int i;
    /* REGISTER signal handler */
    register_SIGCHLD_sighandler();
    while( 1 ) {
        pid=fork();
        if ( pid==0 ) {
            funcFiglio(); exit(0);
        } else
            sleep(1);
    }
}
```

Esempio di SIGUSR1

vedere codice nella pagina di sistop