Fundamentos de Sistemas de Operação

Unix Windows NT Netware MacOS DOS/VS Vax/VMS Linux Solaris HP/UX AIX Mach Chorus

Programação Concorrente

Interacção por Sinais

(continuação)

Sinais durante Chamadas ao SO

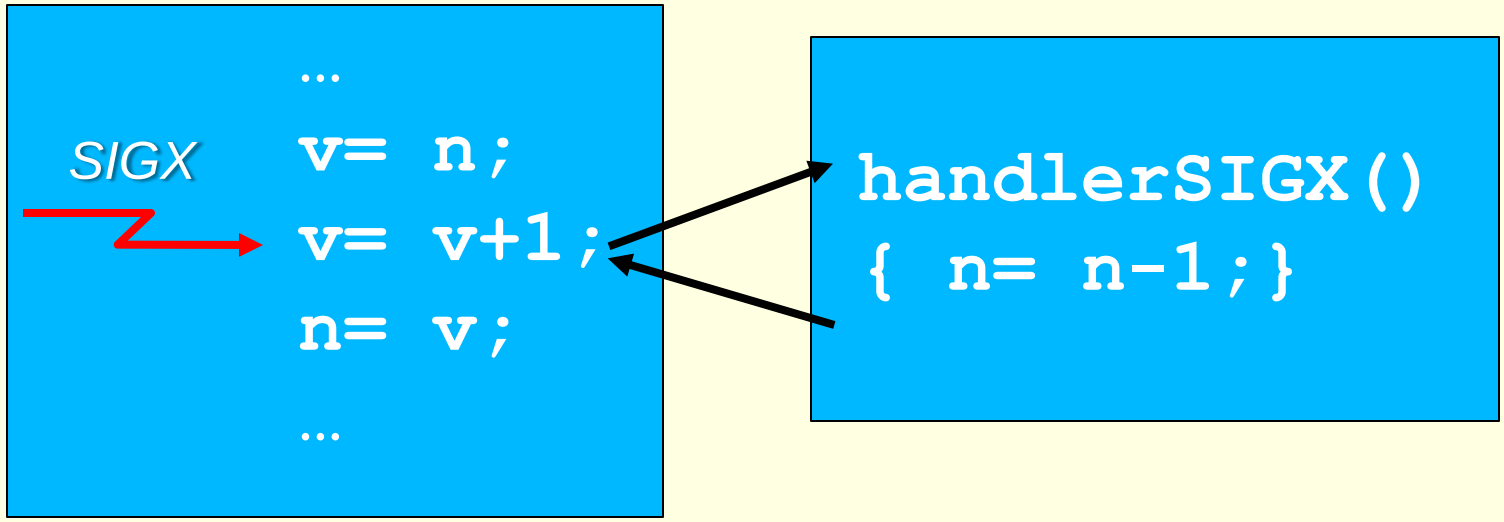
- O que acontece quando é gerado um sinal destinado a um processo que invocou uma chamada ao SO e ainda não retornou ao programa utilizador?
  - Para chamadas não bloqueantes o SO pode deferir a entrega do sinal para um instante posterior, em que a chamada já esteja concluída.
  - Mas se a chamada for bloqueante, por ex. um read/write num pipe, um wait(), … a chamada é interrompida, o handler é executado; depois,
    - A execução da chamada é reexecutada [esta opção pode ser especificada na estrutura **`sigaction`**], e completa-se correctamente, ou,
    - A execução da chamada não é retomada e retorna com erro **`EINTR`**.

Sinais e fork/exec

- *Que acontece quando é executado um* **`fork()`** *ou* **`exec()`***?*
  - **`fork()`***: o processo filho herda as definições dos sinais do pai (sinais a tratar e respectivos handlers). Se o pai programou um* **`alarm()`***, o seu disparo não é recebido pelo filho.*
  - **`exec()`***: todos os tratamentos específicos (i.e., não default) programados são convertidos nas acções por omissão.*

Cuidados na programação com sinais

- *Que acontece quando o handler e o restante programa partilham variáveis?*



A situação é equivalente à partilha entre threads ou processos!

Exemplos de chamadas que usam sinais

- **`int alarm(int seg)`**
  - Permite ao processo programar um "disparo" de um sinal (**`SIGALRM`**) ao fim de **`seg`** segundos.
  - O utilizador deverá definir um handler que será executado se o sinal for recebido
  - A chamada retorna um valor positivo para indicar que há outro alarme activo, sendo que esse valor representa os segundos que faltam para esse alarme disparar; ou zero, para indicar que o alarme foi programado (e não há nenhum outro activo)
  - Para "desactivar" o alarme o utilizador executa **`alarm(0)`**

(continua)

Exemplos de chamadas que usam sinais

- **`int alarm(int seg)`** (continuação)
  - Exemplo de uso: pode ser usado para programar acções que são executadas ciclicamente em intervalos de tempo pré-definidos.
  - Exemplo de uso: pode ser usado para desbloquear situações por timeout – se uma chamada fica “demasiado” tempo bloqueada, é interrompida pelo alarme.

Exemplos de chamadas que usam sinais

- **`int raise(int signo)`**
  - Envia o sinal **`signo`** ao próprio processo.
- **`int pause(void)`**
  - Espera bloqueado por um sinal; quando o recebe, executa o handler. Se este não terminar o processo, a chamada retorna -1 e **`EINTR`**.

Fundamentos de Sistemas de Operação

Unix Windows NT Netware MacOS DOS/VS Vax/VMS
Linux Solaris HP/UX AIX Mach
Chorus

Exemplos de utilização de Sinais

Exemplo com **`alarm()`**

- *Objectivo: definir uma função,* **`mySleep()`**, *que tem o mesmo efeito que a função* **`sleep()`**

```
void alarmHandler(int signo)
{ return; }

void my_Sleep( int s )
{  signal(SIGALRM, alarmHandler);
   alarm(s);
   pause();
   signal(SIGALRM, SIG_DFL);
}
```

Exemplo com `sigaction()` (1)

- *Objectivo: fazer um programa, que recebe os caracteres introduzidos no teclado tratando os sinais que são gerados quando se primem as teclas*

```
// dados e rotina de tratamento de um sinal de I/O
int gotdata=0;
void sighandler(int signo)
{
    if (signo==SIGIO)
        gotdata++;
    return;
}
```

Exemplo com `sigaction()` (2)


```
char buffer[4096];

int main(int argc, char **argv)
{
  int count;
  struct sigaction action;

  memset(&action, 0, sizeof(action));
  action.sa_handler = sighandler;
  sigaction(SIGIO, &action, NULL);

  // programar o stdin para funcionar em assíncrono com sinal de I/O
  fcntl(STDIN_FILENO, F_SETOWN, getpid());
  fcntl(STDIN_FILENO, F_SETFL, fcntl(STDIN_FILENO, F_GETFL) | FASYNC);
```

(continua)

Exemplo com `sigaction()` (3)


```
    ...

    while(1) {
        pause();        // Bloqueia até chegar um sinal
        if (!gotdata)   // se este não foi causado por premir uma tecla
            continue;   // continuar em pause()

        count=read(0, buffer, 4096);  // aqui, houve tecla premida
        write(1,buffer,count);        // escrever no écrã
        gotdata=0;
    }
}
```