2011

Diseño de Sistemas Operativos

David Guillermo Morales Sáez

[PRÁCTICA 2: ALARMA TEMPORIZADA]

1.- En primer lugar, se ha creado un programa que esté utilizando continuamente la pantalla pero que, cada 4 segundos, lanza un hijo que muestra la hora y la fecha actual por pantalla. Para esto, se ha utilizado el comando date. Además, para poder ver la fecha, se ha añadido una pausa de un segundo tras mostrar la fecha. El código es el siguiente:

```
#include <signal.h>
#include <iostream>
#include <stdio.h>
#include <sys/wait.h>
using namespace std;
bool alarma = false;

void setAlarm(int)
{
	alarma = true;
	alarm(0);
	alarm(4);
}

int main()
{
	alarm(4);
	signal(14, setAlarm);
	while(true)
	{
	  printf(" ");
	  if(alarma)
	  {
		alarma = false;
		if(fork()!=0)
		{
			waitpid(0, NULL, 0);
			sleep(1);
		}
		else
		{
			char* args[] = {""};
			printf("\n");
			execv("/bin/date", args);
		}
	  }
	}
	return 0;
}
```

Si observamos el código, en primer lugar veremos la llamada de una alarma cada 4 segundos con la función alarm(int segundos). Después indicamos que, cuando se reciba la señal 14 (SIGALARM) se ejecute la función setAlarm, que modifica una variable global booleana, estableciendo su valor a Verdadero. Además, reinicia la alarma.

Tras estas declaraciones, se inicia un bucle infinito, donde se imprime un espacio por pantalla y, si la variable global está a Verdadero, se pone a falso y se lanza un hijo, quien ejecuta el comando "/bin/date" mientras el padre espera a su finalización. Una vez ejecutado el comando, el hijo muere y entonces el padre espera un segundo (sleep(1)). Finalmente, vuelve a ejecutarse el bucle. En caso que la variable global esté a falso, sólo imprime por pantalla el espacio.

2.- Para permitir que trabajen a la vez tanto el padre como el hijo, se ha añadido un semáforo que impide que ambos procesos impriman a la vez en pantalla. Este semáforo es el incluido en el lenguaje C ("semaphore.h"). El código es el siguiente:

```
#include <signal.h>
#include <iostream>
#include <stdio.h>
#include <sys/wait.h>
#include <semaphore.h>

using namespace std;
bool alarma = false;
sem_t semaforo;

void setAlarm(int)
{
	alarma = true;
	alarm(0);
	alarm(4);
}

int main()
{
	sem_init(&semaforo, 0, 1);
	alarm(4);
	signal(14, setAlarm);
	while(true)
	{
		sem_wait(&semaforo);
		printf(" ");
		sem_post(&semaforo);
		if(alarma)
		{
			alarma = false;
			if(fork()!=0)
			{
				waitpid(0, NULL, WNOHANG);
				printf("Soy el padre");
				sleep(1);
			}
			else
```

```
                {
                        char* args[] = {""};
                        sem_wait(&semaforo);
                        printf("\n Soy el hijo: ");
                        sem_post(&semaforo);
                        execv("/bin/date", args);
                }
        }
    }
    return 0;
}
```

Como podemos comprobar, se ha puesto como opción al waitpid el flag de WNOHANG, que permite al padre trabajar a la par que el hijo. Por otro lado, la estructura del semáforo se inicializa al principio del código y, cada vez que alguien va a imprimir, solicita el semáforo y, tras imprimir lo devuelve. Hay que comentar que la ejecución del comando date ha de hacerse fuera de la sección controlada por los semáforos, ya que el execv sustituye el programa por, en este caso, date.