Temporizador en la llamada bloqueante recvfrom

Elaborado por: Ukranio Coronilla

Una llamada a `recvfrom` se va a bloquear hasta que no llegue un mensaje, que normalmente se ha solicitado antes mediante una llamada `sendto` previa. En ciertas aplicaciones conviene sacar al proceso del bloqueo después de cierto tiempo transcurrido (timeout) donde no se ha recibido respuesta.

Utilizaremos las señales POSIX (véase el capítulo 11 *Señales y temporización* del manual "Programación de Sistemas Linux") para activar el temporizador que saque del bloqueo al proceso que ha mandado llamar a `sendto`.

Ejercicio 1: Añada una variable privada `struct timeval` y tres métodos a la clase `SocketDatagrama` denominados:

```
setTimeout(time_t segundos, suseconds_t microsegundos);
```

para que un socket tenga asociado un temporizador. También se debe añadir una variable privada `bool timeout` cuyo valor será 1 si está activado el temporizador y 0 en caso contrario.

Observe que en este caso sólo se podrá hacer la temporización de tiempos pequeños menores a un segundo mediante la función `ualarm()`, o de tiempos múltiplos enteros de un segundo mediante `alarm()`.

```
unsetTimeout( );
```

el cual deshabilitará el timeout dejando el socket en su estado inicial (véase man 7 socket).

```
recibeTimeout(PaqueteDatagrama &p);
```

que va a contener la función `recvfrom` y va a evaluar su valor de retorno. Si este valor es negativo significa que la señal SIGALRM la ha sacado del bloqueo y deberá imprimir el mensaje "Tiempo para recepción transcurrido", junto con el tiempo que se programó para la alarma.

```
Tiempo para recepción transcurrido. Tiempo = 0s 500000us
```

Tenga en cuenta de que en caso de que el mensaje se haya recibido correctamente, deberá desactivar la alarma.

Ejercicio 2: Además de imprimir el mensaje "Tiempo para recepción transcurrido", el método `recibeTimeout` deberá imprimir para el caso de que no haya llegado la señal SIGALRM, el

tiempo que ha tardado desde que se ejecuta la función `sendto()` y hasta que se recibe el mensaje con la función `recvfrom()`, si es que esto ha sucedido. Para ello utilizaremos la función `gettimeofday()` que inicializa su primer parámetro con el número de segundos más la fracción de microsegundos transcurridos desde el 1 de Enero de 1970 hasta este momento.

Un ejemplo de su uso se muestra a continuación:

```
#include <stdio.h>
#include <sys/time.h>

int main(void)
{
    struct timeval tiempo;

    gettimeofday(&tiempo, NULL);
    printf("segundos: %ld\n", tiempo.tv_sec);
    printf("Microsegundos: %ld\n", tiempo.tv_usec);

}
```

Utilice cualquiera de las funciones: `timeradd, timersub, timercmp, timerclear, timerisset`; y lleve a cabo la medición de tiempo que se solicita en este inciso.

Sugerencia: Para verificar si su programa es correcto mida el tiempo que tarda el siguiente par de instrucciones. Siempre deberá devolver muy aproximadamente el mismo valor, sin importar el número de ejecuciones.

```
sleep(1);
usleep(500000);
```

Ejercicio 3: Para probar estos métodos, elabore un cliente **monoproceso** que imprime todas las IP's de los servidores (descritos en el capítulo 9 del manual “Programación de Sistemas Linux”) que se encuentran activos en la subred del profesor. El ruteador comienza a asignar las IPs desde la 192.168.0.100 en adelante.