

Signal Handling (1)

- Signals are used for communication between hardware and processes, or between processes
- When a signal is received:
 - ★ the process stops whatever its doing
 - ★ a special signal handler function is called
 - ★ after this function is finished, the process continues

Signal Handling (2)

- You can install your own signal handler functions.
- Signals can be used to abort system calls
 - ★ like receiving a packet !

Signal Handling (3)

- For CN, you must be able to handle lost packets
 - ★ they cause `ip_receive` to wait forever!!
- You can solve this by using an **ALARM** signal
- Alarms are generated by the clock of the machine

Signal Handling (4)

```
#include <signal.h>

typedef void (*sighandler_t)(int);

sighandler_t signal(int signum, sighandler_t handler);

unsigned int alarm(unsigned int seconds);
```

- You can set your own alarms using **alarm(...)**
- You need to install a signal handler function using **signal(...)**
- This function must be of type **sighandler_t**

Signal Handling (5)

- Signal handler function must fit prototype **sig_handler_t**
 - ★ return void and have an int parameter.

```
#include <signal.h>
static void tcp_alarm_handler(int signal_number)
{
    /* This function has a the shape of
       void (*sig_handler_t)(int);
    */
}
```


Signal Handling (7)

- Then use this function to set a timeout on a receive.

```
#include <ip.h>
#include <signal.h>

typedef void (*sighandler_t)(int);

#define TIMEOUT 1

static void tcp_alarm_handler(int signal_number) { ... }
static void set_timeout(unsigned int n_seconds) { ... }

void my_receive(...) {
    int result;
    set_timeout(TIMEOUT);
    result = receive(...);
    set_timeout(0);

    /* timeout if result < 0 */
    /* received something result >= 0 */
}
```

Set Timeout Function (1)

```
static void set_timeout(unsigned int n_seconds)
{
    static int          timeout_on = 0;
    static unsigned int old_seconds;
    static sighandler_t old_handler;

    if (n_seconds == 0) {
        assert(timeout_on);
        alarm(0);
        old_handler = signal(SIGALRM, old_handler);
        assert(old_handler != SIG_ERR);
        alarm(old_seconds);
        timeout_on = 0;
    } else {
        assert(!timeout_on);
        old_seconds = alarm(0);
        old_handler = signal(SIGALRM, tcp_alarm_handler);
        assert(old_handler != SIG_ERR);
        alarm(n_seconds);
        timeout_on = 1;
    }
}
```

Alarm Handler Function (1)

- Alarm handler should look like this:

```
static void tcp_alarm_handler(int signal_number)
{
    if (signal_number == SIGALRM) {
        alarm(1);    /* CASCADE */
    }
}
```