Unix Signals
Section 6.2 of Donahoo

Signals

- A notification to a process that an event has occurred.
- AKA a software interrupt
- Usually handled asynchronously
- Can be sent
  - Process to process
  - Kernel to process

Types of signals

- SIGALRM: expiration of the alarm timer
- SIGCHLD: Child process exits
- SIGINT: Interrupt (CNT-C)
- SIGIO : Socket ready for I/O
- SIGTERM or SIGKILL : Terminate a process

Unix kill : send a signal

```
NAME
     kill - send signals to processes, or list signals. The command kill sends
      the specified signal to the specified process or
       process group.  If no signal is specified, the  TERM  signal  is  sent.
       The  TERM  signal  will  kill processes which do not catch this signal.
       For other processes, it may be necessary to use the  KILL  (9)  signal,
       since this signal cannot be caught.


SYNOPSIS
     kill [-s SIGNAL | -SIGNAL] PID...
     kill -l [SIGNAL]...
     kill -t [SIGNAL]...

DESCRIPTION
     Send signals to processes, or list signals.

     Mandatory arguments to long options are mandatory for  short
     options too.

     -s, --signal=SIGNAL, -SIGNAL

          specify the name or number of the signal to be sent

     -l, --list
          list signal names,  or  convert  signal  names  to/from
          numbers


EXAMPLE:
     >>ps –aux
      >>kill -9 pid       %Kill is a 9
```

Signals

- Each has a disposition or an action.
  - Catching the signal
  - Ignore the signal
  - Set to the default disposition (usually terminate or ignore)

Two ways to set the disposition

- Signal function

```
signal (SIGINT, clientCNTCCode);
```

- Sigaction function

Sigaction …. SIGALARM

```
Struct sigaction myaction;

Myaction.sa_handler = CatchAlarm;
If (sigfillset(&myaction.sa_mask) < 0)  //blocks all sigs during this sig
  exit

Myaction.sa_flags = 0;
If (sigaction(SIGALRM,&myaction,0) < 0)
  exit
```

Socket timeout example

```
alarm(2) //set the timeout for 2 seconds
rc = recv()                          //do a socket read
If (rc == -1)
  if (errno == EINTR)           //If no data is available, it unblocks with error
                                     //2 seconds
        timeouts++;

alarm(0);                            //turn off alarm

//If you are going to retry an operation, you must reset the alarm!!!
```

Socket timeout example

```
void CatchAlarm(int ignored);

CatchAlarm (unsigned int unused)
{
    //Can do something in this routine if needed,  or can just return
    return;
}
```

Cnt-c example

```
void clientCNTCCode();

Main {
 signal (SIGINT, clientCNTCCode);

….

}
void clientCNTCCode() {
    printf("UDPEchoClient:  CNT-C Interrupt,  exiting....\n");
}
```