Part 2: Signals

- **Basic Signal Handling:**

  - **kill** (pid, sig)

    To send a signal sig to a process pid
    *(To list all available signals type: `% kill -l`)*

  - **pause()**

    To wait for a signal

  - **signal** (sig, handler)

    To call the function handler
    when the signal sig occurs.

*Example 6: signal1.c*

```c
main ...
{
    signal(SIGUSR1, handler);
    signal(SIGUSR2, handler);
    for (;;) pause();
}

handler(int sig)
{
    /* Print out what we received */
    psignal(sig, "Received signal");
}
```

*Examples of usage:*

```bash
% kill -l
HUP INT QUIT ILL TRAP ABRT EMT FPE KILL BUS SEGV SYS PIPE ALRM TERM USR1 USR2 CLD PWR WINCH URG POLL STOP TSTP CONT TTIN TTOU VTALRM PROF XCPU XFSZ WAITING
```
LWP FREEZE THAW CANCEL LOST RTMIN RTMIN+1
RTMIN+2 RTMIN+3 RTMAX-3 RTMAX-2
RTMAX-1 RTMAX

% signal &
12345

% kill -USR1 12345
Recived signal: User Signal 1

% kill -16 12345 //signal #16 is USR1
Recived signal: User Signal 1

% kill -KILL 12345
Killed

○ Using Signals for Timeouts:

**alarm** (T)
To send the ALRM signal to the current process after T seconds. alarm (0) will cancel the alarm signal.

**Example 7:** timeout1.c

```c
main ...
{
    signal(SIGALRM, handler);
    alarm(atoi(argv[1]));
    printf("Enter a string: ");
    fgets(buf, sizeof(buf), stdin);
    alarm(0);
}

handler(int sig)
{
    printf("inside intr hand
");
}
```

`timeout1.c`