Signal Handling

Signals

- Software notification to a process of an event

- Generated by an event

- Delivered when a process receiving the signal takes an action based on that signal

- Signal lifetime
  - Interval between signal generation and delivery
  - There may be considerable time between generation and delivery

- Signal pending if it has been generated but not delivered

- Process state must be running at the time of signal delivery

- Catching a signal
  - Process executes a signal handler upon delivery
  - Process can install a signal handler by calling `sigaction(2)` with the name of a user-written function
  - `sigaction(2)` may also be called with `SIG_DFL` or `SIG_IGN` instead of a handler
    * These two actions are not considered to be catching the signal
  - If signal is ignored, it is thrown away at delivery and has no effect on the process

- Process signal mask
  - Decides the action in addition to the current signal handler for that signal
  - Contains a list of currently blocked signals
    * Blocked signals are not thrown away as ignored signals
    * If a pending signal is blocked, it is delivered when the process unblocks that signal
  - Process blocks a signal by changing its signal mask by using `sigprocmask(2)`

Generating signals

- Each signal has a symbolic name starting with `SIG`

- All signal names defined in `signal.h`

<table>
<thead>
<tr>
<th>No.</th>
<th>Signal</th>
<th>Description</th>
<th>Default Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SIGHUP</td>
<td>Hang-up (death) on controlling terminal</td>
<td>Abnormal termination</td>
</tr>
<tr>
<td>2</td>
<td>SIGINT</td>
<td>Interactive attention signal (usually <code>^C</code>)</td>
<td>Abnormal termination</td>
</tr>
<tr>
<td>3</td>
<td>SIGHUP</td>
<td>Interactive termination: core dump (usually <code>^D</code>)</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>4</td>
<td>SIGILL</td>
<td>Invalid hardware instruction</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>5</td>
<td>SIGTRAP</td>
<td>Trace trap</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>6</td>
<td>SIGABRT</td>
<td>Process abort</td>
<td>Implementation dependent</td>
</tr>
</tbody>
</table>

- Two signals (`SIGUSR1` and `SIGUSR2`) are available for users and do not have a preassigned use

- Some signals such as `SIGFPE` and `SIGSEGV` are generated upon certain errors; others are generated by specific calls such as

- You can generate signals from the shell with the `kill` command
You can get all possible signals by using the command `kill -l`.

The `kill(2)` system call:

- Used to send signals to a process owned by the user from within the process:

```
if ( kill ( 3423, SIGUSR1 ) == -1 )
    perror ( "Failed to send the SIGUSR1 signal" );
```

- A child may kill its parent by:

```
if ( kill ( getpid(), SIGTERM ) == -1 )
    perror ( "Failed to kill parent" );
```