Today

- Signals
- Real-Time Signals
Signals

What are they?
Notifications sent from the OS or programs

Why are they useful?
Listen and catch signals to do something (alarms)

What should we do with them?
Send, listen, ignore, handle
### Some signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Value</th>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGHUP</td>
<td>1</td>
<td>Term</td>
<td>Hangup detected on controlling terminal or death of controlling process</td>
</tr>
<tr>
<td>SIGINT</td>
<td>2</td>
<td>Term</td>
<td>Interrupt from keyboard</td>
</tr>
<tr>
<td>SIGQUIT</td>
<td>3</td>
<td>Core</td>
<td>Quit from keyboard</td>
</tr>
<tr>
<td>SIGILL</td>
<td>4</td>
<td>Core</td>
<td>Illegal Instruction</td>
</tr>
<tr>
<td>SIGABRT</td>
<td>6</td>
<td>Core</td>
<td>Abort signal from abort(3)</td>
</tr>
<tr>
<td>SIGFPE</td>
<td>8</td>
<td>Core</td>
<td>Floating point exception</td>
</tr>
<tr>
<td>SIGKILL</td>
<td>9</td>
<td>Term</td>
<td>Kill signal (cannot be caught, blocked, or ignored)</td>
</tr>
<tr>
<td>SIGSEGV</td>
<td>11</td>
<td>Core</td>
<td>Invalid memory reference</td>
</tr>
<tr>
<td>SIGPIPE</td>
<td>13</td>
<td>Term</td>
<td>Broken pipe: write to pipe with no readers</td>
</tr>
<tr>
<td>SIGALRM</td>
<td>14</td>
<td>Term</td>
<td>Timer signal from alarm(2)</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>15</td>
<td>Term</td>
<td>Termination signal</td>
</tr>
<tr>
<td>SIGUSR1</td>
<td>30,10,16</td>
<td>Term</td>
<td>User-defined signal 1</td>
</tr>
<tr>
<td>SIGUSR2</td>
<td>31,12,17</td>
<td>Term</td>
<td>User-defined signal 2</td>
</tr>
</tbody>
</table>

...
Handle a signal

First, define a handler function to execute when the signal arrives:

```c
void handler(int signum)
{
    printf("Signal has arrived!\n")
    ...
}
```

Once a signal handler function is defined, you can install a signal handler using the sigaction call.
Problems with signals

- Multiple signals are coalesced when blocked
  - Only one pending signal is delivered
  - Lose multiple signals
- No order of signal delivery
Realtime Signals

• Allow queuing of signals
  – Multiple signals of the same type can be delivered
• Ordering of signal delivery
  – Queued signals in FIFO order
Using Realtime Signals

• Sender
  – Can enqueue multiple instances of the same signal type
• Receiver
  – Can pick up one signal instance at time
Realtime handlers

Void func(int signo, siginfo_t *info, void *context)

• Parameters
  – Signo: Signal number
  – Info: Contains information about signal data
  – Context: Undefined
Realtime handlers

Void func(int signo, siginfo_t *info, void *context)

- siginfo_t
  - si_signo: Signal number (same as signo)
  - si_code: How signal was generated
    - By a user process using kill
    - Using sigqueue
  - si_value: Data generated with signal
Exercise

Two codes:
- signalQueue.c
- sigqueuehandler.c

Run sigqueuehandler and signalQueue and observe what happens:
- Test with signal 10 and 30
Questions?