CSCI 4061
Introduction to Operating Systems

Instructor: Abhishek Chandra

Outline

- Realtime Signals
  - Properties
  - Usage

Problems with Signals

- Multiple signals are coalesced when blocked
  - Only one pending signal is delivered
  - Lose multiple signals
- Cannot pass data to signal handlers
  - Cannot distinguish between multiple signals
  - No information about the signal-raising event
- Very few user-defined signals
  - SIGUSR1 and SIGUSR2
- No order of signal delivery

Realtime Signals

- Allow queuing of signals
  - Multiple signals of same type can be delivered
- Allow passing data with the signal
  - Can pass the context of signal to a process
  - Data can be received by signal handler
    - E.g.: file descriptor on which data arrived
- Large number of new user-defined signals
  - SIGRTMIN...SIGRTMAX
- Ordering of signal delivery
  - Queued signals in FIFO order
  - Priority order between RT signals
Using Realtime Signals

- **Sender:**
  - Can enqueue multiple instances of the same signal type
  - Can send data with the signal
  - Can send different RT signals with different priority
- **Receiver:**
  - Can pick up one signal instance at a time
  - Can pick up data: catch signals using a different signal handler, or different signal waiting call

Queuing Signals

```c
int sigqueue(pid_t pid, int signo,
union sigval value);
```

- Queues a signal to another process
- Also depends on recipient's action for signal
- Also sends data with the signal
- **Parameters**
  - `pid`: Signal-receiving process
  - Sender should have appropriate permissions
  - `signo`: Signal number
  - `value`: Data passed with the signal
  - Union of int and (void *)

Revisiting Signal Handlers

- `struct sigaction`
  - `sa_handler`: Signal handler
  - `sa_mask`: Additional signals to be blocked in the signal handler
  - `sa_flags`: Special flags and options
  - `sa_sigaction`: Realtime signal handler
  - **For realtime behavior:**
    - `sa_flags` should be set to `SA_SIGINFO`
    - `sa_sigaction` should be set to the handler

Realtime Signal Handlers

```c
void func(int signo, siginfo_t *info,
void *context)
```

- **Parameters:**
  - `signo`: Signal number
  - `info`: Contains information about signal data
  - `context`: Undefined
Realtime Signal 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
    - Timer, asynchronous I/O, etc.
  - si_value: Data generated with the signal
    - Union of int and (void *)

Realtime Signal Example: Sender

int pid;
union sigval value;

/* Set value to send with signal */
value.sival_int = 1;

/* Send signal to be queued */
sigqueue(pid, SIGRTMIN, value);

Realtime Signal Example: Receiver

/* Signal handler */
void myhandler(int signo, siginfo_t *info, void *context)
{
  int val = info->si_value.sival_int;
  printf("Signal: %d, value: %d\n", signo, val);
}

struct sigaction act;
act.sa_sigaction = myhandler; /* Set RT sig handler */
sigemptyset(&act.sa_mask);
act.sa_flags = SA_SIGINFO; /* RT sigs flag */

/* Install the signal handler for SIGRTMIN */
sigaction(SIGRTMIN, &act, NULL);

Waiting for RT Signals: sigwaitinfo

int sigwaitinfo(sigset_t *set, siginfo_t *info);

- Similar to sigwait
  - Returns when one of the signals becomes pending
  - Removes it from set of pending signals
- Parameters
  - set: Set of signals to wait for
  - info: Contains info about signal data
**Realtime Signals Usage Scenarios**

- Lightweight IPC: Processes can pass int values
- Timer interrupts could be sent more efficiently
  - Multiple timer interrupts could be queued
- I/O multiplexing easier
  - File descriptors could be sent with I/O ready signals
- Asynchronous I/O could be signaled easily
  - Data pointer could be returned with the signal

**Realtime Signals Summary**

- Allow queuing of signals
- Allow passing data with the signal
- Can pass the context of signal to a process
- Can do asynchronous operations easily