

CSE 4/589

Week 11 Recitation

Algorithms in PWP

- Piece downloading
 - Rarest first
 - Keep info from `bitfield` message
 - Update with every received `have` message
- Choking and Optimistic Unchoking
 - At most 4 unchoked (and interested) remote peers
 - Two types of unchoking (RU and OU)
 - Must maintain remote peer download/upload rate
 - Must setup timer to perform periodic check

Timed Signal Handling

- Signals are software generated interrupts sent to process when an event occurs
- When a process receives a signal
 - The process stops what it is doing
 - Control is passed to a signal handler
 - After signal handler finishes, process resumes
- User-defined signal handlers
- Macros for common signals
 - SIGINT (on a *cntrl+c*)
 - SIGCHLD (to parent on child stop or exit)
 - SIGILL (on illegal instruction encountered)
 - ...
 - ...
 - **SIGALRM (when timer goes off)**

Timed Signal Handling

- Setup timed alarm for certain intervals and write signal handlers for when alarm(s) go off
- Signal handlers to change state variables
- Stick to one signal handler
 - Handling multiple alarms is tricky

Timed Signal Handling

- `#include <signal.h>`
- To setup a signal handler for some signal (e.g. SIGALRM)
`void (*signal (int sig, void (*func)(int)))(int)`

```
#include <signal.h>

void handle_signal(int sigNum)
{
    /* do something */
    /* reset signal handler each time (IMPORTANT!) */
    signal(sigNum, handle_signal);
}

signal (SIGALRM, handle_signal);
```

Reference:

http://publications.gbdirect.co.uk/c_book/chapter9/signal_handling.html

Timed Signal Handling

```
void sigproc(int n)
{
    signal(SIGINT, sigproc);
    printf("you have pressed ctrl-c \n");
}

void quitproc(int n)
{
    printf("ctrl- pressed to quit\n");
    exit(0); /* normal exit status */
}

main()
{
    signal(SIGINT, sigproc);
    signal(SIGQUIT, quitproc);

    printf("ctrl-c disabled use ctrl- to quit");
    for(;;); /* infinite loop */
}
```

Reference:

<http://www.cs.cf.ac.uk/Dave/C/node24.html>

Setting up an Alarm

- `#include <unistd.h>`

- To setup an alarm

```
unsigned alarm(unsigned seconds)
```

```
unsigned int interval = 10;
```

```
/* setup signal handler for SIGALRM */
```

```
alarm (interval) /* generate SIGALRM after interval seconds */
```

- If interval is 0, pending alarm request is cancelled
- Subsequent alarms overwrite currently set alarm

Reference:

<http://www.opengroup.org/onlinepubs/000095399/functions/alarm.html>