Multiplexed I/O

Motivation

- Consider a process that reads from multiple sources without knowing in advance which source will provide some input first
- Three solutions:
  - alternate non-blocking reads on input sources (wasteful of CPU)
  - fork a process for each input source, and each child can block on one specific input source (can be hard to coordinate/synchronize)
  - use the **`select()`** system call … (*see next slide*)

`select()` (Wang, 12.14)

- Usage:

```
#include <sys/time.h>
#include <sys/types.h>
int select( int nfds,
            fd_set *readfds,
            fd_set *writefds,
            fd_set *exceptfds,
            struct timeval *timeout );
```

- where the three **`fd_set`** variables are file descriptor *masks*
- **`fd_set`** is defined in **`<sys/select.h>`**, which is included by **`<sys/types.h>`**

Details

- The first argument (**`nfds`**) represents the number of bits in the masks that will be processed. Typically, this is 1 + the value of the highest fd
- The three **`fd_set`** arguments are bit masks … their manipulation is discussed on the next slide
- The last argument specifies the amount of time the select call should wait before completing its action and returning:
  - if **`NULL`**, select will wait (*block*) indefinitely until one of the file descriptors is ready for i/o
  - if **`tv_sec`** and **`tv_usec`** are zero, select will return immediately
  - if timeval members are non-zero, the system will wait the specified time *or* until a file descriptor is ready for i/o
- **`select()`** returns the number or file descriptors ready for i/o

"**`FD_`**" macros

- Useful macros defined in **`<sys/select.h>`** to manage the masks:

```
void FD_ZERO( fd_set &fdset );
void FD_SET( int fd, fd_set &fdset );
void FD_CLR( int fd, fd_set &fdset );
int  FD_ISSET( int fd, fd_set &fdset );
```

- Note that each macro is passed the *address* of the file descriptor mask

Example

```
#include <sys/types.h>
fd_set rmask;
int fd;        /* a socket or file descriptor */
FD_ZERO( &rmask );
FD_SET( fd, &rmask ); FD_SET( fileno(stdin), &rmask );
for(;;) {
   select( fd+1, &rmask, NULL, NULL, NULL );
   if( FD_ISSET( fileno(stdin, &rmask ) )
      /* read from stdin */
   if( FD_ISSET( fd, &rmask ) )
      /* read from descriptor fd */
   FD_SET( fd, &rmask); FD_SET( fileno(stdin), &rmask );
}
```