poll()

```
#include <poll.h>
int poll (struct pollfd *fdarray, unsigned long nfds,
                                  int timeout);
```

- Similar to `select()`
- Provides additional information when dealing with STREAMS devices
- Parameter:
  - `fdarray`: point to the array of `pollfd` structures
  - `nfds`: number of elements in `fdarray`
  - `timeout`: INFTIM(wait forever), 0(return immediately) or >0(wait specified number of milliseconds)
- Return: number of elements have had event, 0 if timeout, -1 if error

pollfd structure

```
struct pollfd {
    int fd;          // the socket descriptor
    short events;    // bitmap of events we're interested in
    short revents;   // when poll() returns, bitmap of events
                     //that occurred
};
```

- events and revents are bitmasks constructed by OR'ing a combination of the following event flags

Constant	events	revents	Description
POLLIN	x	x	Normal or priority data can be read
POLRDNORM	x	x	Normal data can be read
POLLRDBAND	x	x	Priority (OOB) data may be read
POLLPRI	x	x	High-priority data may be read

`poll()` – Event flags(cont.)

Constant	events	revents	Description
POLLOUT	x	x	Normal data may be written
POLLWRNORM	x	x	Equivalent to POLLOUT
POLLWRBAND	x	x	Priority (OOB) data may be written
POLLERR		x	An error has occurred on socket
POLLHUP		x	The hangup state
POLLNVAL		x	Something was wrong with the socket descriptor *fd*

Example

```
struct pollfd ufds[2];
s1 = socket(AF_INET, SOCK_STREAM, 0);
s2 = socket(AF_INET, SOCK_STREAM, 0);
//connect to server…
ufds[0].fd = s1;
ufds[0].events = POLLIN;
ufds[1].fd = s2;
ufds[1].events = POLLOUT;
rv = poll(ufds, 2, 3500);
if (rv == -1) {
    perror("poll"); // error occurred in poll()
} else if (rv == 0) {
    printf("Timeout occurred!  No data after 3.5 seconds.\n");
} else {
    // check for events on s1:
    if (ufds[0].revents & POLLIN)
        recv(s1, buf1, sizeof buf1, 0);

    // check for events on s2:
    if (ufds[1].revents & POLLOUT)
        send(s2, buf2, sizeof buf2, 0);
}
```