Мультиплексирование ввода/вывода

Программирование с использованием POSIX thread library

select(3C)

```
#include <sys/time.h>

int select(int nfds,
  fd_set *restrict readfds,
  fd_set *restrict writefds,
  fd_set *restrict errorfds,
  struct timeval *restrict timeout);
```

nfds – это максимальный номер дескриптора во всех наборах

fd_set

- Описывает множество дескрипторов файлов
- Номер дескриптора не может быть больше, чем FD_SETSIZE
- На 32-битных платформах FD_SETSIZE==1024
- На 64-битном Solaris FD_SETSIZE==65536

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

readfds, writefds, errorfds

- Входные и выходные параметры
- readfds
  - дескрипторы, годные для чтения
  - дескрипторы, где достигнут конец файла (e.q. закрытые на другом конце трубы и сокеты)
  - слушающие сокеты (accept)

  ^^^ *это нельзя понять, это можно только запомнить*
- writefds
  - дескрипторы, годные для записи (write не заблокируется)
  - для сокетов гарантируется запись SO_SNDLOWAT байт
  - закрытые на другом конце трубы и сокеты
- errorfds
  - зависит от типа устройства, напр. для регулярных файлов не используется
  - для сокетов TCP/IP – приход внеполосных данных (URGENT)

struct timeval

```
long tv_sec;
long tv_usec;
```

- Если timeout==NULL, ждать бесконечно
- Если timeout->tv_sec/tv_usec==0, работает в режиме опроса (возвращается немедленно)
- В остальных случаях обозначает таймаут (максимальное время ожидания)
- Может модифицироваться при успешном завершении

Пример использования

```
#include    "unp.h"

void str_cli(FILE *fp, int sockfd) {
     int          maxfdp1, stdineof;
     fd_set       rset;
     char         sendline[MAXLINE], recvline[MAXLINE];

     stdineof = 0;
     FD_ZERO(&rset);
     for ( ; ; ) {
      if (stdineof == 0) FD_SET(fileno(fp), &rset);
      FD_SET(sockfd, &rset);
      maxfdp1 = max(fileno(fp), sockfd) + 1;
      Select(maxfdp1, &rset, NULL, NULL, NULL);
      if (FD_ISSET(sockfd, &rset)) {/* socket is readable */
            if (Readline(sockfd, recvline, MAXLINE) == 0) {
                  if (stdineof == 1) return;    /* normal termination */
                  else  err_quit("str_cli: server terminated prematurely");
            }

            Fputs(recvline, stdout);
      }
      if (FD_ISSET(fileno(fp), &rset)) {  /* input is readable */
            if (Fgets(sendline, MAXLINE, fp) == NULL) {
                  stdineof = 1;
                  Shutdown(sockfd, SHUT_WR);    /* send FIN */
                  FD_CLR(fileno(fp), &rset);
                  continue;
            }
            Writen(sockfd, sendline, strlen(sendline));
      }
     }
}
```

poll(2)

```
#include <poll.h>

int poll(struct pollfd fds[],
  nfds_t nfds, int timeout);
```

nfds – это количество дескрипторов в fds

struct pollfd

```
int     fd;         /* file descriptor */
short   events;     /* requested events */
short   revents;    /* returned events */
```

- POLLIN (== POLLRDNORM | POLLRDBAND)
  - для слушающих сокетов означает готовность accept
- POLLOUT
- POLLERR (только в revents)
- POLLHUP (только в revents)
- POLLNVAL (только в revents)

Пример использования

```
#include <poll.h>


    struct pollfd fds[3];
    int ifd1, ifd2, ofd, count;

    fds[0].fd = ifd1;
    fds[0].events = POLLNORM;
    fds[1].fd = ifd2;
    fds[1].events = POLLNORM;
    fds[2].fd = ofd;
    fds[2].events = POLLOUT;
    count = poll(fds, 3, 10000);
    if (count == -1) {
            perror("poll failed");
            exit(1);
    }
    if (count==0)
            printf("No data for reading or writing\n");
    if (fds[0].revents & POLLNORM)
            printf("There is data for reading fd %d\n", fds[0].fd);
    if (fds[1].revents & POLLNORM)
            printf("There is data for reading fd %d\n", fds[1].fd);
    if (fds[2].revents & POLLOUT)
            printf("There is room to write on fd %d\n", fds[2].fd);
```

poll(7d)

- Solaris only (начиная с Solaris 7)

```
#include <sys/devpoll.h>
int fd = open("/dev/poll", O_RDWR);
ssize_t n = write(int fd,
  struct pollfd buf[], int bufsize);
Pollfd.events=POLLREMOVE;
int n = ioctl(int fd, DP_POLL,
        struct dvpoll* arg);
int n = ioctl(int fd, DP_ISPOLLED,
     struct pollfd* pfd);
```

Преимущества poll(7d)

http://developers.sun.com/solaris/articles/polling_efficient.html

Данные для Solaris 7

```
bash-2.03$ a.out 4000 poll
Opened 4000 file descriptor(s)
Starting write / [poll|ioctl] / read loop:
104858 write()/poll()/read()s on 4000 fd(s) took : 34254 msec.
bash-2.03$ a.out 4000 devpoll
Opened 4000 file descriptor(s)
Starting write / [poll|ioctl] / read loop:
104858 write()/ioctl(DP_POLL)/read()s on 4000 fd(s) took : 2179 msec.
```