## 第3-2课 讲稿（I/O多路复用-poll()函数）

在本实例中，用poll()函数替换select()函数，实现如第3-1课讲稿里面实例的功能：

poll()函数的语法格式如表3所示。

|  |  |
| --- | --- |
| 表3 poll()函数的语法格式 | |
| 所需头文件 | #include <sys/types.h>  #include <poll.h> |
| 函数原型 | int poll(struct pollfd \*fds, int numfds, int timeout) |
| 函数传入值 | fds：struct pollfd 结构的指针，用于描述需要对哪些文件的哪种类型的操作进行监控。  struct pollfd  {  int fd; /\* 需要监听的文件描述符 \*/  short events; /\* 需要监听的事件 \*/  short revents; /\* 已发生的事件 \*/  }  events成员描述需要监听哪些类型的事件，可以用以下几种标志来描述。  POLLIN：文件中有数据可读，用1表示，实验中使用到了这个标志  POLLPRI:：文件中有紧急数据可读，用2表示  POLLOUT：可以向文件写入数据，用4表示  POLLERR：文件中出现错误，只限于输出，用8表示  POLLHUP：与文件的连接被断开了，只限于输出，用16表示  POLLNVAL：文件描述符不合法，即它并没有指向一个成功打开的文件，用32表示 |
| numfds：需要监听的文件个数，即第一个参数所指向的数组中的元素数目 |
| timeout：表示poll阻塞的超时时间（毫秒）。如果该值小于等于0，则表示无限等待 |
| 函数返回值 | 成功：返回大于0 的值，表示事件发生的pollfd结构的个数  0：超时；  -1：出错 |

【程序代码】

|  |
| --- |
| #include <fcntl.h>  #include <stdio.h>  #include <unistd.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <errno.h>  #include <poll.h>  #define MAX\_BUFFER\_SIZE 1024 /\* 缓冲区大小/  #define IN\_FILES 3 /\* 多路复用输入文件数目 \*/  #define TIME\_DELAY 60 /\* 超时时间秒数 \*/  #define MAX(a, b) ((a > b)?(a):(b))  int main(void)  {  struct pollfd fds[IN\_FILES];  char buf[MAX\_BUFFER\_SIZE];  int i, res, real\_read, maxfd;  fds[0].fd = 0;  if((fds[1].fd = open ("in1", O\_RDONLY|O\_NONBLOCK) ) < 0)  {  printf("Open in1 error\n");  return 1;  }  if((fds[2].fd = open ("in2", O\_RDONLY|O\_NONBLOCK) ) < 0)  {  printf("Open in2 error\n");  return 1;  }  /\*监听三个文件的读事件/  for (i = 0; i < IN\_FILES; i++)  {  fds[i].events = POLLIN;  }  /\*判断三个文件的读事件是否有效，循环监听三个文件/  while(fds[0].events || fds[1].events || fds[2].events)  {  if (poll(fds, IN\_FILES, 0) < 0)  {  printf("Poll error\n");  return 1;  }  for (i = 0; i< IN\_FILES; i++)  {  if (fds[i].revents)  {  memset(buf, 0, MAX\_BUFFER\_SIZE);  real\_read = read(fds[i].fd, buf, MAX\_BUFFER\_SIZE);  if (real\_read < 0)  {  if (errno != EAGAIN)  {  return 1;  }  }  else if (!real\_read)  {  close(fds[i].fd);  fds[i].events = 0;  }  else  {  if (i == 0)  {  if ((buf[0] == 'q') || (buf[0] == 'Q'))  {  return 1;  }  }  else  {  buf[real\_read] = '\0';  printf("%s", buf);  }  } /\* end of if real\_read\*/  } /\* end of if revents \*/  } /\* end of for \*/  } /\*end of while \*/  exit(0);  } |