NAME

mpx, join, chan, extract, attach, detach, connect, npgrp, ckill, mpxcall − create and manipulate multiplexed files

SYNOPSIS

mpx (name, access)
char *name;

join (fd, xd)

chan (xd)

extract (i, xd)

attach (i, xd)

detach (i, xd)

connect (fd, cd, end)

npgrp (i, xd, pgrp)

ckill (i, xd, signal)

#include <sys/mx.h>
mpxcall (cmd, vec)
int *vec;

DESCRIPTION

Mpxcall (cmd, vec) is the system call shared by the library routines described below. *Cmd* selects a command using values defined in **<sys/mx.h>**. *Vec* is the address of a structure containing the arguments for the command:

mpx (name, access)

Mpx creates and opens the file *name* with access permission *access* (see *creat* (2)) and returns a file descriptor available for reading and writing. A −1 is returned if the file cannot be created, if *name* already exists, or if the file table or other operating system data structures are full. The file descriptor is required for use with other routines.

If *name* designates a null string, a file descriptor is returned as described, but no entry is created in the file system.

Once created, an mpx file may be opened (see *open* (2)) by any process. This provides a form of interprocess communication whereby a process B can "call" process A by opening an mpx file created by A. To B, the file is ordinary with one exception: the *connect* primitive could be applied to it. Otherwise, the functions described below are used only by process A and descendants that inherit the open mpx file.

When a process opens an mpx file, the owner of the file receives a control message when the file is next read. The method for "answering" this kind of call involves using *attach* and *detach* as described in more detail below.

Once B has opened A's mpx file, it is said to have a *channel* to A. A channel is a pair of data streams: in this case, one from B to A and the other from A to B. Several processes may open the same mpx file, yielding multiple channels within the one mpx file. By accessing the appropriate channel, A can communicate with B and any others. When A reads (see *read* (2)) from the mpx file, data written to A by the other processes appears in A's buffer using a record format described in *mpxio* (5). When A writes (see *write* (2)) on its mpx file, the data must be formatted in a similar way.

The following commands are used to manipulate mpx files and channels.

join − adds a new channel on an mpx file to an open file F. I/O on the new channel is

I/O on F.
chan − creates a new channel.
extract − file descriptor maintenance.
connect − similar to join except that the open file F is connected to an existing channel.
attach and *detach* − used with call protocol.
npgrp − manipulates process group numbers so that a channel can act as a control terminal (see *tty*(4)).
ckill − send signal (see *signal*(2)) to process group through channel.

A maximum of 15 channels may be connected to an mpx file. They are numbered 0 through 14. *Join* may be used to make one mpx file appear as a channel on another mpx file. A hierarchy or tree of mpx files may be set up in this way. In this case, one of the mpx files must be the root of a tree where the other mpx files are interior nodes. The maximum depth of such a tree is 4.

An *index* is a 16-bit value that denotes a location in an mpx tree other than the root: the path through mpx "nodes" from the root to the location is expressed as a sequence of 4-bit nibbles. The branch taken at the root is represented by the low-order 4-bits of an index. Each succeeding branch is specified by the next higher-order nibble. If the length of a path to be expressed is less than 4, then the illegal channel number, 15, must be used to terminate the sequence. This is not strictly necessary for the simple case of a tree consisting of only a root node; its channels can be expressed by the numbers 0 through 14. An index *i* and file descriptor *xd* for the root of an mpx tree are required as arguments to most of the commands described below. Indices also serve as channel identifiers in the record formats given in *mpxio*(5). Since −1 is not a valid index, it can be returned as a error indication by subroutines that normally return indices.

The operating system informs the process managing an mpx file of changes in the status of channels attached to the file by generating messages that are read along with data from the channels. The form and content of these messages is described in *mpxio*(5).

join (fd, xd) establishes a connection (channel) between an mpx file and another object. *Fd* is an open file descriptor for a character device or an mpx file and *xd* is the file descriptor of an mpx file. *Join* returns the index for the new channel if the operation succeeds and −1 if it does not.

Following join, *fd* may still be used in any system call that would have been meaningful before the join operation. Thus, a process can read and write directly to *fd* as well as access it via *xd*. If the number of channels required for a tree of mpx files exceeds the number of open files permitted a process by the operating system, some of the file descriptors can be released using the standard *close*(2) call. Following a close on an active file descriptor for a channel or internal mpx node, that object may still be accessed through the root of the tree.

chan (xd) allocates a channel and connects one end of it to the mpx file represented by file descriptor *xd*. *Chan* returns the index of the new channel or a −1 indicating failure. The *extract* primitive can be used to get a non-multiplexed file descriptor for the free end of a channel created by *chan*.

Both *chan* and *join* operate on the mpx file specified by *xd*. File descriptors for interior nodes of an mpx tree must be preserved or reconstructed with *extract* for use with *join* or *chan*. For the remaining commands described here, *xd* denotes the file descriptor for the root of an mpx tree.

Extract (i, xd) returns a file descriptor for the object with index *i* on the mpx tree with root file descriptor *xd*. A −1 is returned by extract if a file descriptor is not available or if the arguments do not refer to an existing channel and mpx file.

attach (i, xd)
detach (i, xd). If a process A has created an mpx file represented by file descriptor *xd*, then a

process B can open (see *open*(2)) the mpx file. The purpose is to establish a channel between A and B through the mpx file. *Attach* and *Detach* are used by A to respond to such opens.

An open request by B fails immediately if a new channel cannot be allocated on the mpx file, if the mpx file does not exist, or if it does exist but there is no process (A) with a multiplexed file descriptor for the mpx file (i.e. *xd* as returned by *mpx*(2)). Otherwise, a channel with index number *i* is allocated. The next time A reads on file descriptor *xd*, the WATCH control message (see *mpxio*(5)) will be delivered on channel *i*. A responds to this message with *attach* or *detach*. The former causes the open to complete and return a file descriptor to B. The latter deallocates channel *i* and causes the open to fail.

One mpx file may be placed in 'listener' mode. This is done by writing *ioctl(xd, MXLSTN, 0)* where *xd* is an mpx file descriptor and MXLSTN is defined in **/usr/include/mx.h**. The semantics of listener mode are that all file names discovered by *open*(2) to have the syntax **system!pathname** (see *uucp*(1C)) are treated as opens on the mpx file. The operating system sends the listener process an OPEN message (see *mpxio*(5)) which includes the file name being opened. *Attach* and *detach* then apply as described above.

Detach has two other uses: it closes and releases the resources of any active channel it is applied to, and should be used to respond to a CLOSE message (see *mpxio*(5)) on a channel so the channel may be reused.

connect(fd, cd, end). *Fd* is a character file descriptor and *cd* is a file descriptor for a channel, such as might be obtained via *extract(chan(xd), xd)* or by *open*(2) followed by *attach*. *Connect* splices the two streams together. If *end* is negative, only the output of *fd* is spliced to the input of *cd*. If *end* is positive, the output of *cd* is spliced to the input of *fd*. If *end* is zero, then both splices are made.

npgrp(i, xd, pgrp). If *xd* is negative, *npgrp* applies to the process executing it, otherwise *i* and *xd* are interpreted as a channel index and mpx file descriptor, and *npgrp* is applied to the process on the non-multiplexed end of the channel. If *pgrp* is zero, the process group number of the indicated process is set to the process number of that process, otherwise the value of *pgrp* is used as the process group number.

Npgrp normally returns the new process group number. If *i* and *xd* specify a nonexistant channel, *npgrp* returns −1.

ckill(i, xd, signal) sends the specified signal (see *signal*(2)) through the channel specified by *i* and *xd*. If the channel is connected to anything other than a process, *ckill* is a null operation. If there is a process at the other end of the channel, the process group will be interrupted (see *signal*(2), *kill*(2)). *Ckill* normally returns *signal*. If *ch* and *xd* specify a nonexistent channel, *ckill* returns −1.

FILES

/usr/include/sys/mx.h
/usr/include/sys/ioctl.h

SEE ALSO

ioctl(2), mpxio(5)

BUGS

Mpx files are an experimental part of the operating system more subject to change and prone to bugs than other parts. Maintenance programs, e.g. *icheck*(1M), diagnose mpx files as an illegal mode. Channels may only be connected to objects in the operating system that are accessible through the line discipline mechanism. Higher performace line disciplines are needed. A non-destructive *disconnect* primitive (inverse of *connect*) is not provided. A non-blocking flow control strategy based on messages defined in *mpxio*(5) should not be attempted by novices; the enabling *ioctl* command should be protected. The *join* operation could be subsumed by *connect*. A mechanism is needed for moving a channel from one location in an mpx tree to another.

Attempts to read when there are less than 14 bytes left in the channel buffer may cause an incorrect length to be returned on the read. There are problems with splicing channels and redirecting them.