NAME

mpxio − multiplexed I/O

DESCRIPTION

Data transfers on *mpx* files (see *mpx* (2)) are multiplexed by imposing a record structure on the I/O stream. Each record represents data from/to a particular channel or a control or status message associated with a particular channel.

The prototypical data record read from an *mpx* file is as follows

```
struct input_record {
        short   index;
        short   count;
        short   ccount;
        char    data[];
};
```

where *index* identifies the channel, and *count* specifies the number of characters in *data*. If *count* is zero, *ccount* gives the size of *data*, and the record is a control or status message. Although *count* or *ccount* might be odd, the operating system aligns records on short (i.e. 16-bit) boundaries by skipping bytes when necessary.

Data written to an *mpx* file must be formatted as an array of record structures defined as follows:

```
struct output_record {
        short   index;
        short   count;
        short   ccount;
        char    *data;
};
```

where the data portion of the record is referred to indirectly and the other cells have the same interpretation as in *input_record*.

The control messages listed below may be read from a multiplexed file descriptor. They are presented as two 16-bit integers: the first number is the message code (defined in *<sys/mx.h>*), the second is an optional parameter meaningful only with M_WATCH.

- M_WATCH a process 'wants to attach' on this channel. The second parameter is the 16-bit user-id of the process that executed the open.
- M_CLOSE the channel is closed. This message is generated when the last file descriptor referencing a channel is closed. The *detach* command (see *mpx* (2) should be used in response to this message.
- M_EOT indicates logical end of file on a channel. If the channel is joined to a typewriter, EOT (control-d) will cause the M_EOT message under the conditions specified in *tty* (4) for end of file. If the channel is attached to a process, M_EOT will be generated whenever the process writes zero bytes on the channel.
- M_UBLK is generated for a channel when the internal queues have drained below a threshold.
- M_SIG is generated instead of a normal asynchronous signal on channels that are joined to typewriters. The parameter is the signal number.

Two other messages may be generated by the kernel. As with other messages, the first 16-bit quantity is the message code.

M_OPEN is generated in conjunction with 'listener' mode (see *mpx*(2)). The uid of the calling process follows the message code as with M_WATCH. This is followed by a null-terminated string which is the name of the file being opened.

M_IOCTL is generated for a channel connected to a process when that process executes the *ioctl (fd, cmd, &vec)* call on the channel file descriptor. The M_IOCTL code is followed by the *cmd* argument given to *ioctl* followed by the contents of the structure *vec*. It is assumed, not needing a better compromise at this time, that the length of *vec* is determined by *sizeof (struct sgttyb)* as declared in <*ioctl.h*>.

Two control messages are understood by the operating system. M_EOT may be sent through an *mpx* file to a channel. It is equivalent to propagating a zero-length record through the channel; i.e. the channel is allowed to drain and the process or device at the other end receives a zero-length transfer before data starts flowing through the channel again. M_IOCTL can also be sent through a channel. The format is identical to that described above.

FILES

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

SEE ALSO

ioctl(2), mpx(2), tty(4)