**PBRead** Read data from a device driver

#include <Devices.h> **Device Manager**

```
OSErr       PBRead(pb, async);
ParmBlkPtr  pb ;      address of a 50-byte IOParam structure
Boolean     async ;   0=await completion; 1=immediate return
            returns   Error Code; 0=no error
```

**PBRead** is used by the **File Manager** and the **Device Manager**.

**File Manager Usage**

**PBRead** reads a specified number of bytes from an open file (data or resource fork) and transfers them to your buffer. It updates the file mark, in preparation for the next sequential read.

*pb* is the address of a 50-byte IOParam structure. The following fields are relevant:

| Out-In | Name | Type | Size | Offset | Description |
|---|---|---|---|---|---|
| –> | ioCompletion | ProcPtr | 4 | 12 | Completion routine address (if *async* =TRUE ) |
| –> | ioRefNum | short | 2 | 24 | File reference number |
| –> | ioBuffer | Ptr | 4 | 32 | Address of I/O buffer |
| –> | ioReqCount | long | 4 | 36 | I/O transfer size, in bytes |
| –> | ioPosMode | short | 2 | 44 | Positioning Mode (1=absolute, 2=from EOF, et.al)<br>Bit 7=newline mode; high byte=newline delimiter |
| –> | ioPosOffset | long | 4 | 46 | Positioning delta (bytes from start, EOF, ...)<br>Return: none (error in IM, see Tech Note #187) |
| <- | ioResult | OSErr | 2 | 16 | Error Code (0=no error, 1=not done yet) |
| <- | ioActCount | long | 4 | 40 | Actual number of bytes transferred |

*async* is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

**File Mgr Returns**: an operating system Error Code. It will be one of:

| | | |
|---|---|---|
| noErr | (0) | No error |
| eofErr | (-39) | End of file encountered |
| extFSErr | (-58) | External file system |
| fnOpnErr | (-38) | File not open |
| ioErr | (-36) | I/O error |
| posErr | (-40) | Can't position to before start of file |
| rfNumErr | (-51) | Bad ioRefNum |

---

File Manager Notes:

**PBRead** is similar to **FSRead**, but with additional options. You can perform the I/O asynchronously (if *async* =TRUE), you can specify the position to start reading in the call, and you can use the "newline" mode (see below).

If you attempt to read past the logical end-of-file, all bytes up to the EOF (if any) are transferred to your buffer, the file mark is set to the end of file, the actual number of bytes read is returned in ioActCount, and eofErr is returned.

**Note**: This differs from other operating systems which return an error

only when the file mark is at the EOF when you start the read.

The following example reads the 1004-th 74-byte record from a file. It assumes that *pb* is an ioParam structure as returned from a previous call to **PBOpen**, **PBHOpen**, **PBOpenRF**, etc.:

```
MyStruct  theBuf;                    /* a fictitious 74-byte structure */

pb.ioReqCount = 74;                  /* assume a 74-byte record */
pb.ioPosMode = fsFromStart;          /* absolute positioning */
pb.ioPosOffset = 74 * 1003;          /* point to 1004-th record */
pb.ioBuffer = theBuf;                /* prepare for the read */
err=PBRead( &pb, FALSE );            /* read the record */
if (err != 0) { /* . . . handle errors . . . */ }
```

## Newline Mode

The File Manager provides a method to read a file as a series of lines of text. The so-called 'newline mode' reads from the specified file position up to the next occurrence of a specified delimiter, such as a carriage return.

To use this feature, you must set bit 7 of ioPosMode and store the desired delimiter as the high byte of that field. The ioPosOffset field is ignored. If no delimiter is encountered, then ioReqCount bytes are read. In either case, ioActCount returns with the number of bytes read from the file.

The following example is a function which reads a carriage return-delimited line from a file and stores it in the caller's buffer. Its return value is the length of the line or a (negative) system Error Code. The function can be called repeatedly to read lines sequentially.

## Example

```
#include <Files.h>

#define EOL 13              // line delimiter is CR (ASCII 13)
#define MAXLINE 256         // caller's buffer size; max line len

/*
    ReadLine reads a line of text from the file represented by pbp.
    It returns -1 if the file is not delimited, or a file manager
    result code (i.e. eofErr.) It places the null terminated string
    in the buffer pointed to by lineBuf.
*/
short ReadLine(ioParam *pbp, char *lineBuf)
{
    IOParam   pb;
    short   len, rc;

    pb = *pbp;                              // make copy of caller's param block

    pb.ioPosMode = fsAtMark | 0x80 | (256*EOL);
    pb.ioReqCount = MAXLINE;                // max line size
```

```
    pb.ioBuffer = lineBuf;                  // transfer to this address

    rc=PBRead( &pb, FALSE );                // read one line

    if (rc==eofErr && pb.ioActCount==0)
        return ( eofErr );                  // end of file reached

    if ((rc==noErr) || (rc==eofErr)) {
        len = pb.ioActCount;
        if (len==MAXLINE) return(-1);       // not a delimited file
        if (lineBuf[len-1] != EOL) len++;   // last line has no EOL
        lineBuf[len-1]=0;                   // convert to ASCIIZ
    }
    return (rc);
}
```

---

**Device Manager Usage**

**PBRead** reads a specified number of bytes (ioReqCount) from a device driver with the specified ioRefNum and transfers them to your ioBuffer. If there is a drive number, it is read from ioVRefNum. IOPosOffset reveals the device driver's position. IOActCount is the number of bytes actually read.

*pb* is the address of a 50-byte IOParam structure. The following fields are relevant:

| Out-In | Name | Type | Size | Offset | Description |
|---|---|---|---|---|---|
| –> | ioCompletion | ProcPtr | 4 | 12 | Completion routine address (if *async* =TRUE) |
| –> | ioRefNum | short | 2 | 24 | Device reference number |
| –> | ioBuffer | Ptr | 4 | 32 | Address of I/O buffer |
| –> | ioReqCount | long | 4 | 36 | I/O transfer size, in bytes |
| –> | ioPosMode | short | 2 | 44 | Positioning Mode (0=current position) |
| –> | ioPosOffset | long | 4 | 46 | Positioning delta (bytes from ioPosMode position) |
| <- | ioResult | OSErr | 2 | 16 | Error Code (0=no error, 1=not done yet) |
| <- | ioActCount | long | 4 | 40 | Actual number of bytes transferred |
| –> | ioVRefNum | short | 2 | 22 | Drive identification |

*async* is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

**Device Mgr Returns**: an operating system Error Code. It will be one of:

| | | |
|---|---|---|
| noErr | (0) | No error |
| badUnitErr | (-21) | *refNum* doesn't match unit table |
| unitEmptyErr | (-22) | *refNum* specifies NIL handle in unit table |
| notOpenErr | (-28) | Driver closed |
| readErr | (-19) | Driver can't respond to Read |

---

Device Manager Notes:

**PBRead** should be performed right after a PBWrite to make sure that data written to a block matches the data in memory. Read-verify has the same parameters as a standard Read, except that it includes the constant rdVerify

= 64 in the positioning mode.