FileParam structure

#include <Files.h>

typedef struct **FileParam {**		Size	Offset	Description
ParamBlockHeader		24	0	common fields of ParamBlock types
short	ioFRefNum;	2	24	File reference number
SignedByte	ioFVersNum;	1	26	Version (use 0 for HFS)
SignedByte	filler1;	1	27	(unused)
short	ioFDirIndex;	2	28	Index
unsigned char	ioFlAttrib;	1	30	File Attribute bits (locked, directory, etc)
unsigned char	ioFlVersNum;	1	31	File version (always set to 0)
FInfo	ioFlFndrInfo;	16	32	File type, creator, flags, etc. (see FInfo)
unsigned long	ioFlNum;	4	48	File number
unsigned short	ioFlStBlk;	2	52	First allocation block of data fork
long	ioFlLgLen;	4	54	Logical end-of-file of data fork
long	ioFlPyLen;	4	68	Physical end-of-file of data fork
unsigned short	ioFlRStBlk;	2	62	First allocation block of resource fork
long	ioFlRLgLen;	4	64	Logical end-of-file of resource fork
long	ioFlRPyLen;	4	68	Physical end-of-file of resource fork
unsigned long	ioFlCrDat;	4	72	Date/time of creation (seconds since 1/1/04)
unsigned long	ioFlMdDat;	4	76	Date/Time of last modification
} **FileParam**;		80		

Notes: This structure is used in **PB**xxx calls which typically operate on unopened files:

PBCreate	**PBGetFInfo**	**PBSetFInfo**
PBDelete	**PBRstFLock**	**PBSetFLock**

Functions vary as to which fields are required on entry and which fields are defined upon return. Some fields take on different meanings or even data types in certain cases. Refer to the function in question for additional information on fields.

The most common way to use this structure is to allocate a union which is an aggregate. Then create and initialize a pointer to the desired data type. See ParamBlockRec for an example.

The HFileParam structure is similar but has been modernized for use with HFS-specific calls (**PBH**xxx).