ParamBlockRec union

#include <Files.h>

typedef union **ParamBlockRec{**

		Size	Description
IOParam	ioParam ;	50	Generally used in I/O for open files
FileParam	fileParam ;	80	Used for unopened files
VolumeParam	volumeParam ;	64	Used in volume-specific functions
CntrlParam	cntrlParam;	50	
SlotDevParam	slotDevParam;	36	
MultiDevParam	multiDevParam;	38	
} ParamBlockRec ;		80	(size of aggregate)

typedef ParamBlockRec ***ParmBlkPtr**; Note the spelling convention

---

Notes: All six structures on this union share the same names for the first eight fields (the first 24 bytes). These common fields are defined in a macro called the ParamBlockHeader.

In lieu of Pascal's system of records and variants, C programmers can use predefined unions to access the various parts of the file system parameter blocks. There are several options, but a common way to access the data is by allocating a union (ie, storage for the largest of the union-member structures) and creating pointers which refer to the relevant structure data types:

```
ParamBlockRec   pb;                     // allocate a union
ioParam         *ipb=(IOParam *)&pb;    // and structure ptrs
fileParam       *fpb=(FileParam *)&pb;  // all pointing same addr
volumeParam     *vpb=(VolumeParam *)&pb;

pb.ioParam.ioVRefNum = 2;               // as union member field
pb.fileParam.ioFlFndrInfo.fdType = 'TEXT';
pb.volumeParam.ioVolIndex = 0;

ipb->ioVRefNum = 2;                     // or as a structure field
fpb->ioFlFndrInfo.fdType = 'TEXT';
vpb->ioVolIndex = 0;
```

You can also do ad hoc type coercion:

```
unsigned char pb[80];   // big enough to hold a FileParam or IOParam
short theVRef;

theVRef = ((IOParam *)pb)->ioVRefNum;

((FileParam *)pb)->ioFlLgLen = 1000L;

printf("File type is '%c%c%c%c'\n",pb[32],pb[33],pb[34],pb[35]);
```