HParamBlockRec union

#include <Files.h>

typedef union **HParamBlockRec {**		Size	Description
HIOParam	ioParam;	50	Generally used in I/O for open files
HFileParam	fileParam;	80	Used for unopened files
HVolumeParam	volumeParam;	122	Used in volume-specific functions
AccessParam	accessParam;	48	Used in shared environment calls
ObjParam	objParam;	44	Used in shared environment calls
CopyParam	copyParam;	52	Used in shared environment calls
WDParam	wdParam;	52	Used in shared environment calls
FIDParam	fidParam;	58	Used in calls which manipulate File ID's
CSParam	csParam;	76	Used by **PBCatSearch**
ForeignPrivParam	foreignPrivParam;	68	Used for communication with foreign file systems
} HParamBlockRec;		122	(size of aggregate - largest struct in union structure)

typedef HParamBlockRec ***HParmBlkPtr**; Note **Parm / Param** spelling convention

---

Notes: All ten structures on this union share the same names for the first eight fields (the first 24 bytes). These fields are also the same in the flat file system version ParamBlockRec. These common fields are defined in a macro as the ParamBlockHeader.

This is the HFS variation of the ParamBlockRec. It should be used in HFS-specific calls (**PBH**xxx). A typical technique for using parameter blocks is to allocate the HParamBlockRec union and create pointers that refer to the relevant structure data types:

```
HParamBlockRec  hpb;                    /* allocate a union */
HIOParam  *hipb=(HIOParam *)&hpb;       /* and struc ptrs */
HFileParam     *hfpb=(HFileParam *)&hpb;     /* all point same addr */
HVolumeParam     *hvpb=(HVolumeParam *)&hpb;
AccessParam   *hapb=(AccessParam *)&hpb;
ObjParam  *hopb=(ObjParam *)&hpb;
CopyParam      *hcpb=(CopyParam *)&hpb;
WDParam  *hwpb=(WDParam *)&hpb;

hpb.ioParam.ioVRefNum = 2;              /* as a union member */
hpb.fileParam.ioFlFndrInfo.fdType = 'TEXT';
hpb.volumeParam.ioVolIndex = 0;
hpb.accessParam.ioDenyModes = 0;
hpb.objParam.ioObjType = 2;
hpb.copyParam.ioDstVRefNum = 1;
hpb.wdParam.ioWDIndex = 1;

hipb->ioVRefNum = 2;                    /* or as a structure field */
hfpb->ioFlFndrInfo.fdType = 'TEXT';
```

```
hvpb->ioVolIndex = 0;
hapb->ioDenyModes = 0;
hopb->ioObjType = 2;
hcpb->ioDstVRefNum = 1;
hwpb->ioWDIndex = 1;
```

You can also use ad hoc type coercion:

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

theVRef = ((HIOParam *)pb)->ioVRefNum; /* fetch a field */

((HFileParam *)pb)->ioFlLgLen = 1000L; /* store a field */

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