CInfoPBRec union

#include <Files.h>

typedef union **CInfoPBRec {**		Size	Description
HFileInfo	hFileInfo;	108	Use for files
DirInfo	dirInfo;	104	Use for directories
} CInfoPBRec;		108	(size of aggregate)

typedef CInfoPBRec ***CInfoPBPtr**;

Notes: This union (or either of the HFileInfo or DirInfo structures) is used in calls to **PBGetCatInfo** and **PBSetCatInfo**. Note that the return value of the ioFlAttrib field (of either member structure) identifies which type of information is returned (see **PBGetCatInfo** for details).

A handy technique is to allocate the CInfoPBRec union and create pointers which refer to each data type:

```
CInfoPBRec     cipb;                          /* allocate a union */
HFileInfo   *hfipb=(HFileInfo *)&cipb; /* and separate struc ptrs */
DirInfo        *dipb=(DirInfo *)&cipb;     /* pointing same address */

cipb.hFileInfo.vRefNum = 2;       /* as a union member field */
cipb.dirInfo.ioDrFndrInfo.frLocation.h = 100;

hfipb->vRefNum = 2 ;                    /* or as a structure field */
dipb->ioDrFndrInfo.frLocation.h = 100;
```

You can also perform ad hoc type coercion:

```
unsigned char pb[108];      /* big enough to hold either struct */
short theVRef;

theVRef = ((HFileInfo *)pb)->ioVRefNum;  /* fetch 1 field */

((HFileParam *)pb)->ioFlLgLen = 1000L; /* change some fields */
((HFileInfo *)pb)->ioFlClpSiz = 2048;
GetDateTime(&((HFileInfo *)pb)->ioFlMdDat);
                                        /* access chars of a long */
printf("File type is '%c%c%c%c'\n", pb[32], pb[33], pb[34],pb[35]);
```