Hierarchical File System Unload Utility

Author: Bruce R. Wells

z/OS Security Server RACF

brwells@us.ibm.com

Last updated: 07/15/2013

Change Date	Change Description
10/20/2000	Introduction of irrhfsu
07/20/2001	Changed source file extension from '.c' to '.txt'
06/15/2001	Updated (by Shozab Naqvi) to support Access Control Lists (ACLs)
05/01/2003	Fixed the package description and setup instructions
03/29/2004	Updated to unload the security label (SECLABEL)
04/17/2006	Clarified applicability to tfs/zfs (it applies!) and fixed typo
09/18/2008	- Changed UID/GID fields of ACL definitions in DB2 samples from character to integer - Changed table space name in samples from IRRDBU00 to IRRHFSU - Removed sequence numbers from the RACHFSTB.txt file - Provide an executable version of the utility at the z/OS V1R7 level - Made editorial changes to the README
10/19/2012	- Unload the containing file system data set name - Unload the contents of a symbolic and external link - Remove obsolete conditional compiler directives from source - Fix return code error - Support new type 0904 record for each mounted file system
07/15/2013	- Fix doc error regarding use of the –f option, and the STDOUT DD name in BPXBATCH - Describe how to run the executable from an MVS library - Add FSACCESS requirement to authorization section - Fixed the code to allow compile on an R12 system

Disclaimers, etc.

This program contains code made available by IBM Corporation on an AS IS basis. Any one receiving this program is considered to be licensed under IBM copyrights to use the IBM-provided source code in any way he or she deems fit, including copying it, compiling it, modifying it, and redistributing it, with or without modifications, except that it may be neither sold nor incorporated within a product that is sold. No license under any IBM patents or patent applications is to be implied from this copyright license.

The software is provided "as-is", and IBM disclaims all warranties, express or implied, including but not limited to implied warranties of merchantibility or fitness for a particular purpose. IBM shall not be liable for any direct, indirect, incidental, special or consequential damages arising out of this agreement or the use or operation of the software.

A user of this program should understand that IBM cannot provide technical support for the program and will not be responsible for any consequences of use of the program.

Background

RACF currently provides the IRRDBU00 utility to unload the contents of the RACF database into a flat file suitable for viewing or loading into a relational database for querying. No such capability exists within the RACF product for the security data contained within the z/OS UNIX file system. This data is managed by RACF through a set of SAF callable services in a data area known as the File Security Packet (FSP). Some examples of the data contained within the FSP are: file permission bits, owning UID and GID, owner- and auditor-specified logging options, etc. This data, as well as additional data (see below) can be unloaded using the HFS Unload Utility!

Note: HFS Unload uses standard UNIX interfaces and works equally well on HFS, TFS, or z/FS file systems, all of which are collectively referred to as HFS in this document.

RACF also provides the IRRRID00 utility to remove user and group references from the RACF database. IRRRID00 can be used to delete references to specific user and groups, or can be run to locate references to users and groups which no longer exist. The UNIX find command can be used to locate files which are owned by a specific user or group, and can be used to locate files whose owner cannot be mapped to a RACF user or group. Shell command substitution can be used to issue a command, such as chown or rm, against the files located by find. Similarly, find can locate files with access control lists (ACLs) containing entries for a specific user or group, and command substitution can be used to remove these references using the setfacl command. The find command can also locate files with ACLs containing "orphaned" ACL references; that is, entries for UIDs ad GIDs which can not be mapped to RACF user or group profiles. However, the find output is not useful for removing these references, because the UID or GID is not reported as part of the output. In order to provide coverage for this feature of IRRRID00, the HFS Unload Utility can be invoked with a parameter which results in deletion of orphaned ACL entries.

Package Contents

This package contains

- This README file
- irrhfsu.txt - the source code for the utility
- irrhfsu.o - an executable version of the utility compiled at the z/OS V1R7 level
- RACHFSTB - sample DB2 table definition statements
- RACHFSLD - sample DB2 load statements

Installation Instructions

You have the choice of modifying the source to your liking and compiling it, or, if you don't have access to a C compiler, you can use the executable version provided.

Using the executable

Simply download the executable from the web, and transfer it in binary mode to the HFS directory of your choice. Feel free to rename it. For example, remove the ".o" file extension:

mv irrhfsu.o irrhfsu

Using the source code

Once the package has been downloaded from the web, you must transfer the source code to the HFS in the directory of your choice and compile it. (Before compiling, rename the file from irrhfsu.txt to irrhfsu.c.) In the following examples, we use /u/mydir/tools. The compiler options you specify will depend on what release level you are on. From the shell, in current working directory /u/mydir/tools:

c89 -o irrhfsu irrhfsu.c

If you are compiling on one system, but plan to execute the utility on another, make sure you are compiling correctly for the target system. This is a consideration when compiling on a higher release and executing on a lower release, on which a certain function of this utility may not be supported, as function is added over time. To accomplish this, you must specify the target release when you invoke the compiler. For example, the HFMFS_MUID field in the type 0904 record is only supported starting with z/OS V1R13. If you are compiling irrhfsu.c on R13 for execution on an R12 system, you would compile it as follws:

c89 -Wc,TARGET\(zosv1r12\) -o irrhfsu irrhfsu.c

The z/OS XL C/C++ User's Guide documents the valid target values.

(Note that you sometimes can't compile on a lower release for execution on a higher release. For example, the converse of the preceding example may not work, because the header files on your downlevel system may not have some necessary updates, and your compile would fail.)

If you want the executable to reside in an MVS library, you can simply copy the executable generated as shown above into an MVS library using the shell cp command:

cp irrhfsu "//'SYS1.MYLIB(IRRHFSU)'"

Or, you can bind directly into the library using the c89 command:

c89 -o "//'SYS1.MYLIB(IRRHFSU)'" irrhfsu.c

See example 8 below for a sample invocation of this executable.

Note: Do not use to OGET/OGETX commands to copy the executable. There seems to be some magic in the cp command.

Now you have created the irrhfsu command. Set the permission bits for irrhfsu as appropriate.

Note: when transferring the file from your PC to the host, do so in text mode, not binary mode. If you use IBM eNetwork Personal Communications for the file transfer, make sure the code page is set to "1047 United States", or the file may not compile on the host. In general, using ftp is probably the simplest approach.

irrhfsu - the HFS Unload Utility

The irrhfsu utility will unload HFS file data in a manner which is complimentary to IRRDBU00. It can report on files residing within the currently mounted file system structure. It runs in the shell and creates a record for every file/directory in the HFS sub-tree(s) which is passed into the utility as an argument. It will unload the FSP data, as mentioned above, plus additional data provided by the C stat() routine (for example: creation date, last access date, inode, number of links, etc), as well as the contents of any ACLs which may exist for the file or directory. See below for specification of record format. The utility comes with sample load and table definitions for use with DB2.

The irrhfsu utility can optionally create a record for each mounted file system, containing security information about the file system (for example, the data set name and type, whether it was mounted R/O or R/W, whether setuid bits re being honored, etc). You can choose to unload *only* these types of records, or you can choose to unload them in addition to the file/directory records.

The irrhfsu utility can also be used to delete ACL entries containing UIDs and GIDs which cannot be mapped to RACF user or group profiles ("orphaned" ACL entries). This ability corresponds to the ability of RACF's IRRRID00 utility to delete references to users and groups which no longer exist in the RACF database.

Irrhfsu can be executed from within an interactive shell environment, or from JCL using the BPXBATCH utility. The executable can also be run from an MVS library using EXEC PGM=.

Authorization Required

The invoker must have read and search permission to each directory containing the files to unload. Thus, a general user can use irrhfsu against her own files.

In order to run against files you do not own, you will require either

- ✓ UID 0
- ✓ READ access to the BPX.SUPERUSER profile in the FACILITY class so you can switch to superuser mode via the "su" command before running the tool.
- ✓ READ access to the SUPERUSER.FILESYS profile in the UNIXPRIV class
- ✓ The RACF AUDITOR attribute
- ✓ READ access to SUPERUSER.FILESYS.CHANGEPERMS if you are using irrhfsu to delete orphaned ACL entries.

If the FSACCESS class is implemented, and you do not have the AUDITOR attribute, you will, in addition, require UPDATE access to the FSACCESS profile protecting a given file system data set in order to unload files within it.

You will also require write access to the file or data set you are using for output.

Syntax

The syntax of irrhfsu is as follows:

irrhfsu [-c] [-m | -M] [-f *outputfile*] *dir1* [*dir2* ...]

Where

irrhfsu	- The name of the HFS Unload Utility
-c	- Indicates to clean up (delete) orphaned ACL entries for the specified files/directories. When -c is specified, irrhfsu unloads each orphaned ACL entry before deleting it.
-m	- Indicates to create a type 0904 record for each mounted file system, in addition to the type 0900-0903 records created for file system objects.
-M	- Indicates to create *only* type 0904 records for mounted file systems. When –M is specified dir1 [dir2 …] are ignored if specified.
-f *outputfile*	- The name of the file you wish to contain the utility's output. Either an HFS file or an MVS data set can be specified. To specify an MVS data set, prefix it with two slashes (//) and fully qualify the data set name. Do not enclose the data set name in single quotes, and do not enclose the entire path name in double quotes.
Dir1 [dir2 ...]	- The name of the directory, or directories, whose contents you wish to unload. You can also specify individual files.

If -f *outputfile* is not specified, then irrhfsu will write output to stdout by default. Thus, it is not necessary to specify -f when output is directed to an HFS file (see Example 1 below). When irrhfsu is invoked from the shell, the -f option is only required if you want output directed to an MVS data set, or when you wish to append the output to an existing UNIX file. Note that when invoked from BPXBATCH, the STDOUT DD statement can be used in lieu of the –f option and can name either a UNIX file or an MVS data set as the output file.

When irrhfsu opens the output file specified with -f, it does so in append mode. So, you can specify the MVS data set you use for IRRDBU00 output and the data from both utilities will be combined and ready for use.

The output file is opened with recfm=vb and lrecl=4096. If the output file does not exist, it will be created. If the output file is an MVS data set, and it has already been pre-allocated and catalogued, then the data set attributes must be consistent with recfm=vb and lrecl=4096 or an fopen() error will occur.

Examples

In the following examples, it is assumed that the irrhfsu utility exists in the invoker's current working directory, when invoked from the shell, and in the user's home directory when invoked from batch. You can install it anywhere you wish.

Example 1 - From the shell, unload the contents of the entire file system to HfsuOutFile

```
irrhfsu / > HfsuOutFile
```

Using the redirection operator (>) stdout is directed to the file HfsuOutFile in the user's current working directory. Note that when using ">" to redirect stdout to a file, if the file exists its current contents will be overwritten. Use -f if you want to append output to an existing file.

Example 2 - From the shell, remove orphaned ACL entries from the entire file system

```
irrhfsu -c / > /dev/null
```

Note that when using irrhfsu to delete orphaned ACL entries, it will still perform the unload of security data. If you don't care about the unload output, then redirect stdout to the 'bit bucket', which is accomplished in UNIX by writing to the special file called /dev/null.

Example 3 - From the shell, unload two directories to an MVS data set

```
irrhfsu -f //BRWELLS.HFSU.OUTPUT /u/brwells/dir1 dir2/subdir
```

In this example, the first directory to be unloaded is specified as an absolute path name, and the second directory is specified relative to the current working directory. Note that when specifying a relative path name, the file name will be output as a relative path name. Thus, subsequent queries may not be very helpful in identifying the files for which you are looking. Relative path names may be sufficient for casual browsing of the output (such as in the following example), but you probably want to use absolute path names if you intend to run relational queries against the output.

Example 4 - From the shell, unload a single file to the display

```
irrhfsu myfile
```

The security data for a single file will be output to the display, which is the default for stdout.

Example 5 - From batch, unload the entire file system to the IRRDBU00 output data set

```
//BRWELLSL JOB '577018,B0011038','B.R.WELLS',
// CLASS=2,NOTIFY=BRWELLS,MSGLEVEL=(1,1),
//  MSGCLASS=H
//*****************************************************
//HFSUNLD   EXEC PGM=BPXBATCH,
//  PARM='PGM irrhfsu -f //SYS1.IRRDBU00.OUTPUT   /'
//STDERR  DD  PATH='/u/brwells/hfsuerr',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
```

The irrhfsu utility may create error messages. This example directs stderr to a file in the user's home directory.

Note that this example represents just one way in which to combine the output of IRRDBU00 with that of irrhfsu. Another method would be to write irrhfsu output to an HFS file, and then copy it to an MVS data set in another job step. You should implement the JCL in whatever manner best fits with your current procedure.

Example 6 - From the shell, unload the contents of the entire file system, and information about each mounted file system, to HfsuOutFile

```
Irrhfsu -m / > HfsuOutFile
```

This example supplements Example 1 with mount table information.

Example 7 - From the shell, unload only the mount table information to HfsuOutFile

```
irrhfsu -M > HfsuOutFile
```

This does not unload information about individual file system objects, and so no path names are required.

Example 8 – Execute from an MVS library in batch

```
//BRWELLSL JOB '577018,B0011038','B.R.WELLS',
// CLASS=7,NOTIFY=BRWELLS,MSGLEVEL=1,
//  MSGCLASS=H
//*--------------------------------------
//HFSUNLD  EXEC  PGM=IRRHFSU,PARM=('/-m /')
//STEPLIB  DD  DISP=SHR,DSN=SYS1.MYLIB
//SYSPRINT DD  DISP=SHR,DSN= BRWELLS.DOWNLOAD.HFSU.OUTPUT
```

This executes the program from an MVS library. See "**Using the source code**" above to see how to get the executable into a library. This can be helpful in a Sysplex when your libraries are on shared DASD, but you have not implemented shared HFS.

Notes:

1. The leading "/" in the PARM= string is a signal to LE that the LE parameters have are being terminated, and in this case, there are no LE parameters. The normal input to irrhfsu follows the forward slash.
2. When executed from batch in this fashion, the SYSPRINT DD name must be used instead of STDOUT.
3. You can specify the –f option to specify a UNIX file. You can also use it to specify an MVS data set name in the format "//HLQ.DATASET.NAME", where the fully qualified data set name is specified in double quotes.

UID and GID Name Mapping

For your convenience, each output record contains the UID and GID of the file owner, as well as the RACF user ID and group name to which they map. The same is true for the contents of ACL entries. The C routines getpwuid() and getgrgid() are used for this purpose. The irrhfsu program will locally cache id/name mappings to reduce calls to RACF. However, if your HFS contains UID/GID values which are unknown to RACF, then you should implement the UNIXMAP class, or application identity mapping (AIM), in order to eliminate lengthy searches of the RACF database. See z/OS Security Server RACF Security Administrator's Guide for instructions on setting up the UNIXMAP class. You can use AIM instead of the UNIXMAP class. This involves running a RACF utility program named IRRIRA00 to define database index aliases for UIDs and GIDs. This method is preferred over the UNIXMAP class. See z/OS Security Server RACF System Programmer's Guide for details.

If neither UNIXMAP nor AIM is implemented, the mapping service invoked by getgrgid() and getpwuid() will scan USER and GROUP profiles until a match on the id is found, or the name space is exhausted. (This information will be cached so that subsequent requests for this id can bypass the RACF search. However, the caching algorithm by default only remembers the previous 10 ids encountered in the HFS, so the unknown id can fall out of the cache, and the exhaustive search will need to be performed again if that id is subsequently encountered. Note that you can modify the "maxcache" variable in irrhfsu to increase the number of id/name pairings maintained in the cache.) On the other hand, if UNIXMAP or AIM is active, RACF will stop immediately after checking for existence of the appropriate profile, and if not found, irrhfsu will leave blanks in the associated user ID or group name field.

As an alternative to implementing the UNIXMAP class or AIM, you can remove the mapping code from the C source file and recompile it yourself.

Record Type 0900 - HFS File Basic Data record

The HFS File Basic Data record defines the basic information about an HFS file or directory within the currently mounted file system structure. There is one record per file. This table is consistent with the IRRDBU00 record formats documented in z/OS Security Server RACF Macros and Interfaces.

Field Name	Type	Start	End	Comments
HFSBD_RECORD_TYPE	Int	1	4	Record type of the HFS Basic Data record (0900)
HFSBD_NAME	Char	6	1,028	Path name of file or directory
HFSBD_INODE	Int	1,030	1,039	Inode (file serial number)
HFSBD_FILE_TYPE	Char	1,041	1,048	What type of file is this? Valid values are FILE, DIR, SOCKET, EXTLINK, SYMLINK, FIFO, BLOCK, and CHAR.
HFSBD_OWN_UID	Int	1,050	1,059	The owner's z/OS UNIX user identifier (UID) associated with the file.
HFSBD_OWN_UNAME	Char	1,061	1,068	The owner's RACF user ID
HFSBD_OWN_GID	Int	1,070	1,079	The owner z/OS UNIX group identifier (GID) associated with the file.
HFSBD_OWN_GNAME	Char	1,081	1,088	The RACF group name corresponding to this GID
HFSBD_S_ISUID	Yes/No	1,090	1,093	Is the S_ISUID (set-uid) bit on for this file?
HFSBD_S_ISGID	Yes/No	1,095	1,098	Is the S_ISGID (set-gid) bit on for this file?
HFSBD_S_ISVTX	Yes/No	1,100	1,103	Is the S_ISVTX (sticky) bit on for this file?
HFSBD_OWN_READ	Yes/No	1,105	1,108	Is the owner read bit on for this file?
HFSBD_OWN_WRITE	Yes/No	1,110	1,113	Is the owner write bit on for this file?
HFSBD_OWN_EXEC	Yes/No	1,115	1,118	Is the owner execute bit on for this file?
HFSBD_GRP_READ	Yes/No	1,120	1,123	Is the group read bit on for this file?
HFSBD_GRP_WRITE	Yes/No	1,125	1,128	Is the group write bit on for this file?
HFSBD_GRP_EXEC	Yes/No	1,130	1,133	Is the group execute bit on for this file?
HFSBD_OTH_READ	Yes/No	1,135	1,138	Is the other read bit on for this file?
HFSBD_OTH_WRITE	Yes/No	1,140	1,143	Is the other write bit on for this file?
HFSBD_OTH_EXEC	Yes/No	1,145	1,148	Is the other execute bit on for this file?
HFSBD_APF	Yes/No	1,150	1,153	Is the APF bit on for this file?
HFSBD_PROGRAM	Yes/No	1,155	1,158	Is the program-control bit on for this file?
HFSBD_SHAREAS	Yes/No	1,160	1,163	Is the SHAREAS bit on for this file?
HFSBD_AAUD_READ	Char	1,165	1,172	What are the auditor audit options for READ actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_AAUD_WRITE	Char	1,174	1,181	What are the auditor audit options for WRITE actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_AAUD_EXEC	Char	1,183	1,190	What are the auditor audit options for EXECUTE actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_UAUD_READ	Char	1,192	1,199	What are the user audit options for READ actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_UAUD_WRITE	Char	1,201	1,208	What are the user audit options for WRITE actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_UAUD_EXEC	Char	1,210	1,217	What are the user audit options for EXECUTE actions? Valid values are ALL, SUCCESS, FAIL, and NONE.
HFSBD_AUDIT_ID	Char	1,219	1,250	RACF audit id
HFSBD_FID	Char	1,252	1,267	FID
HFSBD_CREATE_DATE	Date	1,269	1,278	Date the file was created.

HFSBD_CREATE_TIME	Time	1,280	1,287	Time the file was created.
HFSBD_LASTREF_DATE	Date	1,289	1,298	Date of last access
HFSBD_LASTREF_TIME	Time	1,300	1,307	Time of last access
HFSBD_LASTCHG_DATE	Date	1,309	1,318	Date of last file status change
HFSBD_LASTCHG_TIME	Time	1,320	1,327	Time of last file status change
HFSBD_LASTDAT_DATE	Date	1,329	1,338	Date of last data modification
HFSBD_LASTDAT_TIME	Time	1,340	1,347	Time of last data modification
HFSBD_NUMBER_LINKS	Int	1,349	1,358	Number of links
HFSBD_SHARELIB	Yes/No	1,360	1,363	Is the shared library extended attribute bit on for this file?
HFSBD_ACCESS_ACL	Yes/No	1,365	1,368	Does an access ACL exist for this file or directory?
HFSBD_FILEMOD_ACL	Yes/No	1,370	1,373	Does a file default ACL exist for this directory?
HFSBD_DIRMOD_ACL	Yes/No	1,375	1,378	Does a directory default ACL exist for this directory?
HFSBD_SECLABEL	Char	1,380	1,387	The security label (SECLABEL)
HFSBD_DSNAME	Char	1,389	1,432	The name of the data set containing the file system in which this object resides
HFSBD_LINK	Char	1,434	2,456	For a symbolic link or external link, the contents of the link

Record Type 0901 - HFS File Access record

The HFS File Access record defines the users or groups who have specific access to HFS files via an access ACL. There is one record per file/authorization combination.

Field Name	Type	Start	End	Comments
HFACC_RECORD_TYPE	Int	1	4	Record type of the HFS File Access record (0901)
HFACC_NAME	Char	6	1,028	Path name of file or directory
HFACC_INODE	Int	1,030	1,039	Inode (file serial number)
HFACC_TYPE	Char	1,041	1,048	'USER' or 'GROUP'
HFACC_ID	Int	1,050	1,059	UID or GID
HFACC_ID_NAME	Char	1,061	1,068	RACF user ID or group name
HFACC_READ	Yes/No	1,070	1,073	Does the user or group have read access to this file?
HFACC_WRITE	Yes/No	1,075	1,078	Does the user or group have write access to this file?
HFACC_EXEC	Yes/No	1,080	1,083	Does the user or group have search/execute access to this file?

Record Type 0902 - HFS File Default Access record

The HFS File Default Access record defines the users or groups who are defined in a file default ACL, if one exists for a directory. There is one record per file/authorization combination. The mapping is the same as for record type 0901, except that the field name prefix is "HFACF_".

Record Type 0903 - HFS Directory Default Access record

The HFS Directory Default Access record defines the users or groups who are defined in a directory default ACL, if one exists for a directory. There is one record per directory/authorization combination. The mapping is the

same as for record type 0901, except that the field name prefix is “HFACD_”.

Record Type 0904 – Mounted File System record

The Mounted File System record defines the basic information pertaining to a mounted file system.

Field Name	Type	Start	End	Comments
HFMFS_RECORD_TYPE	Int	1	4	Record type of the Mounted File System record (0904)
HFMFS_DSNAME	Char	6	49	MVS data set containing this file system
HFMFS_TYPE	Char	51	58	File system type (HFS, ZFS, etc) from the FILESYSTYPE statement in BPXPRMxx.
HFMFS_MODE	Char	60	69	Mount mode (READONLY or READWRITE)
HFMFS_SECURITY	Char	71	80	Are security checks performed? (SECURITY or NOSECURITY)
HFMFS_SETUID	Char	82	91	Are setuid bits honored? (SETUID or NOSETUID)
HFMFS_MUID	Int	93	102	The effective z/OS UNIX user identifier (UID) of the user who mounted the file system. This will be blanks when compiled for a release lower than z/OS R13.
HFMFS_MUSER	Char	104	111	The user ID who mounted the file system (obtained by mapping the UID). This will be blanks when compiled for a release lower than z/OS R13.
HFMFS_MOUNTPOINT	Char	113	1135	The name of the directory where the file system is mounted.

Messages Created by irrhfsu

The following error messages can be issued by irrhfsu. They will be directed to stderr.

IRR67700I

IRR67700I fopen() error on output file: *message-text*

Explanation: The irrhfsu utility was unable to open the output file specified in the -f option.

System Action: The utility stops processing.

User Response: Use the *message-text* to perform problem determination. If your output file is an MVS data set, and the message text says "An I/O abend was trapped", make sure the user running the utility has RACF access to the output data set.

IRR67701I

IRR67701I ftw() error

Explanation: The irrhfsu utility encountered an error using the C function ftw().

System Action: The utility stops processing.

User Response: Look up the error code.

IRR67702I

IRR67702I stat() could not be executed on *file*. Possible search error on parent directory.

System Action: The utility continues processing the next file.

User Response: If you wish irrhfsu to report on the failed file, then contact the directory owner to grant you search (execute) access, and rerun the utility against the directory.

IRR67703I

IRR67703I Unable to read directory *directory*

System Action: The utility continues processing the next directory.

User Response: If you wish irrhfsu to report on the failed directory, then contact the directory owner to grant you read access, and rerun the utility against the directory.

IRR67704I

IRR67704I fprintf() error while writing to output file

System Action: The utility terminates. The output file will contain records for files which were processed prior to this error.

User Response: Check the system console for message IEC031I indicating an abend D37 with reason code 04. This means you have run out of space in your output file. You need to either allocate a larger output file, or run irrhfsu against a smaller portion of the file system. Also look for message IEC030I indicating an abend B37 with reason code 04. If you've preallocated your own output file, this may indicate insufficient secondary space allocated.

For errors other than these, look up the number of the error message which is displayed after this one, and check the operator console for other indicators.

IRR67705I

IRR67705I acl_get_file() error: *message-text*

Explanation: The irrhfsu utility encountered an error using the C function acl_get_error(). The UNIX error message is displayed in *message-text*.

System Action: The utility continues with the next file.

User Response: An IRR67706I message will immediately follow this one. See the documentation for IRR67706I.

IRR67706I

IRR67706I Error received from acl_get_file while retrieving the *type* ACL for the file *file*

Explanation: The irrhfsu utility encountered an unexpected error trying to retrieve an ACL from the file system using the acl_get_file function. The error code is displayed in the IRR67705I message immediately preceding this one. The ACL *type* can be access ACL, file default ACL, or directory default ACL. The file name is displayed in *file*.

System Action: The utility continues with the next ACL, if another one exists, and then on to the next file. The ACL will not be unloaded for this file, nor will orphan ACL entries be deleted from it if the -c option was specified.

User Response: Try using the UNIX getfacl command to display the ACL in question. If an error is encountered, follow the instructions documented for that error message. The *message-text* in the IRR67705I message immediately preceding this one may also be of help.

IRR67707I

IRR67707I error opening /dev/tty

Explanation: The -c option was specified to remove orphan ACL entries. The irrhfsu utility attempted to issue message IRR67708I to the terminal, but encountered an error opening the terminal. This is probably because irrhfsu was invoked using the BPXBATCH utility. This is not a problem.

System Action: The utility writes the IRR67708I message to stderr instead of to the terminal.

User Response: Look in stderr to see the IRR67708I message.

IRR67708I

IRR67708I There were *number* extended ACL entries deleted as a result of specifying the -c option

Explanation: The -c option was specified to remove orphan ACL entries. This informational message simply reports the *number* of ACL entries which were actually deleted.

System Action: The utility completes successfully.

User Response: Bask in the warm glow resulting from your successful use of irrhfsu.

IRR67709I

IRR67709I w_getmntent() error: *message-text*

Explanation: The irrhfsu utility encountered an error using the C function w_getmntent(). The UNIX error message is displayed in *message-text*.

System Action: The data set name field (HFBD_DSNAME) is set to blanks and the utility continues.

User Response: Look up the error code.

IRR67710I

IRR67710I readlink() error: *message-text*

Explanation: The irrhfsu utility encountered an error using the C function readlink(). The UNIX error message is displayed in *message-text*.

System Action: The link contents field (HFSBD_LINK) is set to blanks and the utility continues.

User Response: Look up the error code.