


```

CCCCCCCC RRRRRRRR EEEEEEEEE EEEEEEEEE TTTTTTTTT EEEEEEEEE
CCCCCCCC RRRRRRRR EEEEEEEEE AAAAAA AAAAAA TTTTTTTTT EEEEEEEEE
CC        RR      RR EE          AA      AA TTT      EE
CC        RR      RR EE          AA      AA TT       EE
CC        RR      RR EE          AA      AA TT       EE
CC        RR      RR EE          AA      AA TT       EE
CC        RRRR?RRR EEEEEEEEE AA      AA TT       EEEEEEEEE
CC        RRRRRRRR EEEEEEEEE AA      AA TT       EEEEEEEEE
CC        RR  RR   EE          AAAAAAAAAA TT       EE
CC        RR  RR   EE          AAAAAAAAAA TT       EE
CC        RR      RR EE          AA      AA TT       EE
CC        RR      RR EE          AA      AA TT       EE
CCCCCCCC RR      RR EEEEEEEEE AA      AA TT       EEEEEEEEE
CCCCCCCC RR      RR EEEEEEEEE AA      AA TT       EEEEEEEEE

```

```

LL        IIIIII SSSSSSSS
LL        IIIIII SSSSSSSS
LL        II     SS
LL        II     SS
LL        II     SS
LL        II     SS
LL        II     SSSSSS
LL        II     SSSSSS
LL        II     SS
LL        II     SS
LL        II     SS
LL        II     SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```

.....

```

1 0001 0 MODULE CREATE (
2 0002 0
3 0003 0     LANGUAGE (BLISS32),
4-1 0004 0     IDENT = 'V04-002'
5 0005 1 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1
9 0009 1
10 0010 1 *
11 0011 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 *   ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 *   TRANSFERRED.
21 0021 1 *
22 0022 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 *   CORPORATION.
25 0025 1 *
26 0026 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *
30 0030 1
31 0031 1 **
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1     This module processes the create function. It creates a file with the
38 0038 1     attributes requested, enters it in a directory if desired, and
39 0039 1     accesses it if requested.
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1
43 0043 1     STARLET operating system, including privileged system services
44 0044 1     and internal exec routines.
45 0045 1
46 0046 1 --
47 0047 1
48 0048 1
49 0049 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 28-Mar-1977 15:05
50 0050 1
51 0051 1 MODIFIED BY:
52 0052 1
53 0053 1     V04-002 CDS0007 Christian D. Saether 2-Jan-1985
54 0054 1     Set the CLF HDRNOTCHG flag in PROPAGATE ATTR when
55 0055 1     calling COPY_INFO from secondary context.
56 0056 1
57 0057 1     V04-001 CDS0006 Christian D. Saether 12-Sep-1984

```

54	0058	1	Modify test for re-reading file header after ENTER (CDS0004).
55	0059	1	
56	0060	1	
57	0061	1	V03-042 CDS0005 Christian D. Saether 31-Aug-1984
58	0062	1	Defer building of ACL's until after initial extend
59	0063	1	takes place so that the map pointer for a contiguous
60	0064	1	file is in the primary header.
61	0065	1	
62	0066	1	V03-041 CDS0004 Christian D. Saether 30-Aug-1984
63	0067	1	Reread newly created header after ENTER because
64	0068	1	it may have been flushed from the cache by a multi
65	0069	1	header directory file.
66	0070	1	
67	0071	1	V03-040 CDS0013 Christian D. Saether 14-Aug-1984
68	0072	1	Modify creation of extension fcb chain, if necessary.
69	0073	1	
70	0074	1	V03-039 LMP0298 L. Mark Pilant, 7-Aug-1984 16:22
71	0075	1	Add the necessary protection checks for create-if.
72	0076	1	
73	0077	1	V03-038 ACG0438 Andrew C. Goldstein, 1-Aug-1984 21:23
74	0078	1	Fix link truncation error; release any existing
75	0079	1	serialization lock before starting create
76	0080	1	
77	0081	1	V03-037 LMP0288 L. Mark Pilant, 29-Jul-1984 13:56
78	0082	1	Make sure that the ACL queue head of the new file is properly
79	0083	1	initialized when copying the ACL from a prior version (this
80	0084	1	bug introduced in LMP0284.)
81	0085	1	
82	0086	1	V03-036 LMP0284 L. Mark Pilant, 26-Jul-1984 12:14
83	0087	1	Fix call to ACL_INIT_QUEUE, since it was moved to ACLSUBR.
84	0088	1	
85	0089	1	V03-035 ACG0440 Andrew C. Goldstein, 25-Jul-1984 14:27
86	0090	1	Move setup of default access ACE to after attributes are written
87	0091	1	
88	0092	1	V03-034 LMP0275 L. Mark Pilant, 23-Jul-1984 14:40
89	0093	1	Don't try to propagate an ACL if there isn't one.
90	0094	1	
91	0095	1	V03-033 ACG0437 Andrew C. Goldstein, 13-Jul-1984 15:27
92	0096	1	Corrections to alternate file ownership: fix interface to
93	0097	1	CHANGE_OWNER so that next version propagation works and
94	0098	1	so that space charging is done correctly. Also add an
95	0099	1	ACL entry for the creator to guarantee access.
96	0100	1	
97	0101	1	V03-032 CDS0012 Christian D. Saether 29-Jun-1984
98	0102	1	Add another call to read_header after copying info
99	0103	1	in propagate_attr because primary header may have
100	0104	1	been flushed from the cache.
101	0105	1	
102	0106	1	V03-031 CDS0011 Christian D. Saether 22-Apr-1984
103	0107	1	Modify access arbitration.
104	0108	1	
105	0109	1	V03-030 CDS0010 Christian D. Saether 11-Apr-1984
106	0110	1	Remove call to allocation_unlock after create_header
107	0111	1	call because that routine does it now.
108	0112	1	
109	0113	1	V03-029 CDS0009 Christian D. Saether 1-Apr-1984
110	0114	1	Call ALLOCATION_UNLOCK prior to deleting previous file

111	0115	1	version in supersede operations to eliminate possible
112	0116	1	deadlock condition if the previous version is being
113	0117	1	extended at the same time.
114	0118	1	Also call ALLOCATION_UNLOCK after an ENTER because it
115	0119	1	may have extended the directory and thus be holding the
116	0120	1	allocation lock, also causing potential deadlock further
117	0121	1	on in a number of ways.
118	0122	1	
119	0123	1	V03-028 ACG0412 Andrew C. Goldstein, 22-Mar-1984 18:19
120	0124	1	Implement agent access mode support; add access mode to
121	0125	1	check protection call; make attribute propagation to self
122	0126	1	a NOP (when a file is entered as a new version of itself).
123	0127	1	
124	0128	1	V03-027 ACG0408 Andrew C. Goldstein, 20-Mar-1984 17:54
125	0129	1	Make APPLY_RVN and DEFAULT_RVN macros;
126	0130	1	Make rest of global storage based.
127	0131	1	
128	0132	1	V03-026 ACG0405 Andrew C. Goldstein, 16-Mar-1984 15:12
129	0133	1	Fix handling of file headers in CHANGE_OWNER
130	0134	1	
131	0135	1	V03-025 CDS0008 Christian D. Saether 9-Mar-1984
132	0136	1	Remember CURR_LCK:NDX from primary context and set
133	0137	1	it in secondary after OPEN FILE so that copy_info
134	0138	1	has the right lock basis when writing acl's to the
135	0139	1	primary file's header.
136	0140	1	
137	0141	1	V03-024 LMP0203 L. Mark Pilant, 29-Feb-1984 10:34
138	0142	1	Add support for FIBSV_PROPAGATE. This allow the propagation
139	0143	1	rules to apply on an enter operation as well as a create
140	0144	1	operation.
141	0145	1	
142	0146	1	V03-023 LMP0189 L. Mark Pilant, 6-Feb-1984 13:54
143	0147	1	Add support for FIBSV_DIRACL. This allows the ACL of a
144	0148	1	directory file parent to be copied directly to the
145	0149	1	children (with the exception of NOPROPAGATE ACEs).
146	0150	1	
147	0151	1	V03-022 LMP0188 L. Mark Pilant, 3-Feb-1984 16:08
148	0152	1	Add support for a classification block.
149	0153	1	
150	0154	1	V03-021 CDS0007 Christian D. Saether 17-Jan-1984
151	0155	1	Modify interface to DEFAULT_RVN.
152	0156	1	
153	0157	1	V03-020 CDS0006 Christian D. Saether 27-Dec-1983
154	0158	1	Use BIND_COMMON macro.
155	0159	1	
156	0160	1	V03-019 LMP0174 L. Mark Pilant, 1-Dec-1983 14:01
157	0161	1	Change routine name for default ACE propagation. Also,
158	0162	1	Add a call to a routine to do general propagation.
159	0163	1	
160	0164	1	V03-018 CDS0005 Christian D. Saether 14-Sep-1983
161	0165	1	Modify interface to SERIAL_FILE routine.
162	0166	1	
163	0167	1	V03-017 ACG5691c Andrew C. Goldstein, 21-Jun-1983 18:25
164	0168	1	Use central routine for date management
165	0169	1	
166	0170	1	V03-016 LMP0156 L. Mark Pilant, 19-Sep-1983 15:43
167	0171	1	Files not entered into a directory now get the process

168	0172	1	default protection.
169	0173	1	
170	0174	1	V03-015 LMP0149 L. Mark Pilant, 13-Sep-1983 11:25
171	0175	1	Correct a logic problem that caused problems during the
172	0176	1	protection check of a write attribute operation.
173	0177	1	
174	0178	1	V03-014 LMP0148 L. Mark Pilant, 31-Aug-1983 13:29
175	0179	1	Make sure propagated attributes make it to the header.
176	0180	1	
177	0181	1	V03-013 CDS0004 Christian D. Saether 16-May-1983
178	0182	1	Release allocation lock after newly allocated file
179	0183	1	header is locked.
180	0184	1	
181	0185	1	V03-012 CDS0003 Christian D. Saether 4-May-1983
182	0186	1	Add call to SERIAL_FILE routine to interlock file
183	0187	1	processing.
184	0188	1	
185	0189	1	V03-011 CDS0002 Christian D. Saether 9-Apr-1983
186	0190	1	Reflect change to ACCESS_LOCK interface.
187	0191	1	
188	0192	1	V03-010 ACG0323 Andrew C. Goldstein, 25-Mar-1983 15:51
189	0193	1	Simplify backlink handling to track RENAME changes
190	0194	1	
191	0195	1	V03-009 ACG53759 Andrew C. Goldstein, 24-Mar-1983 15:10
192	0196	1	Update revision date & count & expiration on ENTER
193	0197	1	
194	0198	1	V03-008 LMP0091 L. Mark Pilant, 18-Mar-1983 16:14
195	0199	1	Add a condition handler to the attribute propagation to
196	0200	1	catch non-existent files. Also, copy the entire file name
197	0201	1	when creating a long file named file.
198	0202	1	
199	0203	1	V03-007 LMP0080 L. Mark Pilant, 14-Feb-1983 16:16
200	0204	1	Add a new routine that is called to propagate the attributes
201	0205	1	from either the previous version of the file or the parent
202	0206	1	directory as necessary.
203	0207	1	
204	0208	1	V03-006 ACG53050 Andrew C. Goldstein, 31-Jan-1983 13:59
205	0209	1	Remove RVN check from check for dummy file ID
206	0210	1	
207	0211	1	V03-005 CDS0001 Christian D. Saether 12-Jan-1983
208	0212	1	Call routine to take out file access lock.
209	0213	1	
210	0214	1	V03-004 LMP0059 L. Mark Pilant, 21-Dec-1982 11:17
211	0215	1	Always create an FCB when accessing a file header. This
212	0216	1	eliminates a lot of special casing in FCB handling.
213	0217	1	
214	0218	1	V03-003 LMP0047 L. Mark Pilant, 29-Sep-1982 12:05
215	0219	1	Put back in the volume protection check deleted by LMP0036.
216	0220	1	
217	0221	1	V03-002 LMP0036 L. Mark Pilant, 5-Aug-1982 13:50
218	0222	1	Shuffle the order that the protection checks are done to
219	0223	1	allow for ACL's.
220	0224	1	
221	0225	1	V03-001 LMP0016 L. Mark Pilant, 25-Mar-1982 13:18
222	0226	1	Remove diddling of the COMPLETE bit in the window segments.
223	0227	1	
224	0228	1	V02-021 ACG0265 Andrew C. Goldstein, 15-Feb-1982 9:50

```

: 225      0229 1  Fix order of expiration date handling
: 226      0230 1
: 227      0231 1  V02-020 ACG0258 Andrew C. Goldstein, 26-Jan-1982 16:57
: 228      0232 1  Fix reference to RVN 1 in expiration date processing
: 229      0233 1
: 230      0234 1  V02-019 ACG0230 Andrew C. Goldstein, 23-Dec-1981 22:59
: 231      0235 1  Add expiration date support
: 232      0236 1
: 233      0237 1  V02-018 ACG0247 Andrew C. Goldstein, 23-Dec-1981 20:44
: 234      0238 1  Set revision date to creation date
: 235      0239 1
: 236      0240 1  V02-017 ACG0245 Andrew C. Goldstein, 23-Dec-1981 20:40
: 237      0241 1  Don't write back link if file is a spool file
: 238      0242 1
: 239      0243 1  V02-016 LMP0003 L. Mark Pilant, 8-Dec-1981 10:20
: 240      0244 1  Added byte limit quota check on window creation.
: 241      0245 1
: 242      0246 1  V02-015 ACG0238 Andrew C. Goldstein, 11-Dec-1981 23:30
: 243      0247 1  Allow creation of dummy directory entries
: 244      0248 1
: 245      0249 1  V02-014 ACG0208 Andrew C. Goldstein, 17-Nov-1981 15:16
: 246      0250 1  Add segmented directory record support
: 247      0251 1
: 248      0252 1  V02-013 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:25
: 249      0253 1  Previous revision history moved to f11B.REV
: 250      0254 1  **
: 251      0255 1
: 252      0256 1
: 253      0257 1  LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 254      0258 1  REQUIRE 'SRC$:FCPDEF.B32';
: 255      1249 1
: 256      1250 1
: 257      1251 1  FORWARD ROUTINE
: 258      1252 1  CREATE : L_NORM, : CREATE function routine
: 259      1253 1  PROPAGATE_ATTR : L_NORM, : Propagate file attributes
: 260      1254 1  PROPAGATE_HANDLER, : condition handler for above
: 261      1255 1  COPY_INFO : L_NORM; : Copy info from old to new file

```

```

263 1256 1 GLOBAL ROUTINE CREATE : L_NORM =
264 1257 1
265 1258 1 ++
266 1259 1
267 1260 1 FUNCTIONAL DESCRIPTION:
268 1261 1
269 1262 1     This routine processes the CREATE function. It creates a file with the
270 1263 1     attributes requested, enters it in a directory if desired, and
271 1264 1     accesses the file if requested.
272 1265 1
273 1266 1 CALLING SEQUENCE:
274 1267 1     CREATE ()
275 1268 1
276 1269 1 INPUT PARAMETERS:
277 1270 1     NONE
278 1271 1
279 1272 1 IMPLICIT INPUTS:
280 1273 1     CURRENT_VCB: VCB of volume
281 1274 1     IO_PACKET: packet of this I/O request
282 1275 1
283 1276 1 OUTPUT PARAMETERS:
284 1277 1     NONE
285 1278 1
286 1279 1 IMPLICIT OUTPUTS:
287 1280 1     PRIMARY_FCB: FCB of file if accessed
288 1281 1     CURRENT_WINDOW: window of file if accessed
289 1282 1     USER_STATUS: I/O status block of user
290 1283 1
291 1284 1 ROUTINE VALUE:
292 1285 1     1 if successful
293 1286 1     0 if error
294 1287 1
295 1288 1 SIDE EFFECTS:
296 1289 1     File created, blocks allocated, directory modified, file accessed, etc.
297 1290 1
298 1291 1 --
299 1292 1
300 1293 2 BEGIN
301 1294 2
302 1295 2 LITERAL
303 1296 2     ACE_LENGTH      = $BYTEOFFSET (ACESL_KEY) + 4;
304 1297 2
305 1298 2 LOCAL
306 1299 2     STATUS,          | general return status
307 1300 2     K,               | miscellaneous constant
308 1301 2     FCB_CREATED,    | flag indicating new FCB created
309 1302 2     PACKET          : REF BBLOCK,   | address of I/O packet
310 1303 2     ABD             : REF BBLOCKVECTOR [ABD$C_LENGTH],
311 1304 2                   | buffer descriptors
312 1305 2     FIB            : REF BBLOCK,   | file identification block
313 1306 2     RESULT_LENGTH, : VECTOR [FILENAME_LENGTH+6, BYTE],
314 1307 2     RESULT         : VECTOR [FILENAME_LENGTH+6, BYTE],
315 1308 2                   | result string from ENTER
316 1309 2     LINK_DID       : BBLOCK [FID$C_LENGTH], ! header back link
317 1310 2     IDENT_AREA     : REF BBLOCK,   | pointer to file header ident area
318 1311 2     PCB            : REF BBLOCK,   | requestor PCB address
319 1312 2     ARB            : REF BBLOCK,   | access rights block of caller

```



```

: 320      1313  2      MAP_AREA      : REF BBLOCK,      : file header map area
: 321      1314  2      IDX_FCB       : REF BBLOCK,      : FCB of index file
: 322      1315  2      FCB           : REF BBLOCK,      : FCB address
: 323      1316  2      UCB           : REF BBLOCK,      : UCB pointer for RVN 1
: 324      1317  2      PRIMARY_VCB   : REF BBLOCK,      : VCB of root volume
: 325      1318  2      HEADER        : REF BBLOCK,      : address of file header
: 326      1319  2      NEW_HEADER    : REF BBLOCK,      : Address of extension header
: 327      1320  2      ACL_CONTEXT,   : REF BBLOCK,      : dummy ACL context longword
: 328      1321  2      ACE           : BBLOCK [ACE_LENGTH], : buffer for ACE for file creator
: 329      1322  2      FUNCTION      : BLOCK [1];       : function code qualifiers
: 330      1323  2
: 331      1324  2      EXTERNAL
: 332      1325  2      ACP$GB_WRITBACK : BITVECTOR ADDRESSING_MODE (ABSOLUTE),
: 333      1326  2      : ACP write back cache enable
: 334      1327  2      SCH$GL_PCBVEC   : REF VECTOR ADDRESSING_MODE (ABSOLUTE), : PCB vector
: 335      1328  2      EXE$GL_DYNAMIC_FLAGS : ADDRESSING_MODE (ABSOLUTE);
: 336      1329  2      : Dynamic SY$GEN flags
: 337      1330  2
: 338      1331  2      EXTERNAL LITERAL
: 339      1332  2      EXE$V_CLASS_PROT; : set if doing non-discretionary checks
: 340      1333  2
: 341      1334  2      BIND_COMMON;
: 342      1335  2
: 343      1336  2      EXTERNAL ROUTINE
: 344      1337  2      ACL_DELETEACL  : ADDRESSING_MODE (GENERAL), : delete acls
: 345      1338  2      UPDATE_FCB    : L_NORM,           : rebuild fcb from header
: 346      1339  2      REBLD_PRIM_FCB : L_NORM NOVALUE,  : rebuild primary fcb from header
: 347      1340  2      BUILD_EXT_FCBS : L_NORM NOVALUE,  : build extension fcb chain
: 348      1341  2      RELEASE_SERIAL_LOCK : L_NORM,          : release file synchronization lock
: 349      1342  2      ALLOCATION_UNLOCK : L_NORM,          : synchronize allocation/deallocation
: 350      1343  2      ARBITRATE_ACCESS : L JSB_2ARGS,     : establish file access.
: 351      1344  2      CONV_ACCLOCK   : L_NORM,          : convert/dequeue access lock.
: 352      1345  2      SERIAL_FILE    : L_NORM,          : interlock file processing.
: 353      1346  2      GET_FIB        : L_NORM,          : get FIB for operation
: 354      1347  2      GET_LOC_ATTR   : L_NORM,          : get placement data form attribute list
: 355      1348  2      GET_LOC       : L_NORM,          : get placement data
: 356      1349  2      SWITCH_VOLUME  : L_NORM,          : switch context to specified volume
: 357      1350  2      SELECT_VOLUME  : L_NORM,          : find volume in volume set for create
: 358      1351  2      CHECK_PROTECT  : L_NORM,          : check file protection
: 359      1352  2      CHARGE_QUOTA   : L_NORM,          : charge blocks to user's disk quota
: 360      1353  2      CREATE_HEADER  : L_NORM,          : create a file ID and header
: 361      1354  2      CHECKSUM      : L_NORM,          : compute header checksum
: 362      1355  2      MARK_DIRTY    : L_NORM,          : mark buffer for write-back
: 363      1356  2      ACL_INIT_QUEUE  : ADDRESSING_MODE (GENERAL), : Initialize ACL queue
: 364      1357  2      ACL_ADDENTRY   : ADDRESSING_MODE (GENERAL), : add entry to ACL
: 365      1358  2      ACL_BUILDACL  : ADDRESSING_MODE (GENERAL) L_NORM, : build ACL into file headers
: 366      1359  2      READ_HEADER   : L_NORM,          : read file header
: 367      1360  2      ENTER        : L_NORM,          : enter file in directory
: 368      1361  2      COPY_NAME     : L_NORM,          : copy file name to result string
: 369      1362  2      SET_REVISION  : L_NORM,          : set file revision and exp dates
: 370      1363  2      CREATE_FCB    : L_NORM,          : create an FCB
: 371      1364  2      CREATE_WINDOW  : L_NORM,          : create a window
: 372      1365  2      SET_EXPIRE    : L_NORM,          : enable expiration date recording
: 373      1366  2      MAKE_ACCESS   : L_NORM,          : complete the access
: 374      1367  2      MARKDEL_FCB   : L_NORM,          : mark FCB for delete
: 375      1368  2      WRITE_ATTRIB  : L_NORM,          : write attributes
: 376      1369  2      EXTEND        : L_NORM,          : extend the file

```

```

377      1370      2          SAVE CONTEXT      : L_NORM,      : save reentrant context area
378      1371      2          RESTORE CONTEXT : L_NORM,      : restore reentrant context area
379      1372      2          MARK DELETE    : L_NORM,      : mark file for delete
380      1373      2          REMAP FILE     : L_NORM,      : remap the file completely
381      1374      2          SEARCH_FCB    : L_NORM ADDRESSING_MODE (GENERAL); ! Search FCB list
382      1375      2
383      1376      2
384      1377      2      ! Enable the deaccess cleanup if an access is taking place.
385      1378      2      !
386      1379      2
387      1380      2      PACKET = .IO PACKET;
388      1381      2      FUNCTION = .PACKET[IRPSW_FUNC];
389      1382      2      IF .FUNCTION[IOSV_ACCESS]
390      1383      2      THEN
391      1384      2          BEGIN
392      1385      2              CLEANUP_FLAGS[CLF_ZCHANNEL] = 1;
393      1386      2              CLEANUP_FLAGS[CLF_DELWINDOW] = 1;
394      1387      2          END;
395      1388      2
396      1389      2      ! Set up pointers to interesting control blocks.
397      1390      2      !
398      1391      2
399      1392      2      PCB = .SCH$GL_PCBVEC[(IO_PACKET[IRPSL_PID])<0,16>];
400      1393      2      ABD = .BBLOCK[.PACKET[IRPSL_SVAPE], AIB$L_DESCRIPTOR];
401      1394      2          ! pointer to buffer descriptors
402      1395      2      FIB = GET_FIB (.ABD);          ! pointer to FIB
403      1396      2
404      1397      2      IF .FIB[FIBSV_TRUNC]
405      1398      2      OR .FIB[FIBSV_VERLIMIT] GTRU 32767
406      1399      2      OR (.FUNCTION[IOSV_DELETE] AND NOT .FUNCTION[IOSV_ACCESS])
407      1400      3      OR (NOT .FUNCTION[IOSV_CREATE]
408      1401      4          AND (.FIB[FIBSV_EXTEND]
409      1402      4              OR .PACKET[IRPSW_BCNT] GTR ABD$C_ATTRIB
410      1403      4              OR .FUNCTION[IOSV_ACCESS]
411      1404      4          )
412      1405      3      )
413      1406      2      THEN ERR_EXIT (SS$_BADPARAM);
414      1407      2
415      1408      2      IF .CURRENT_VCB[VCBSV_NOALLOC]
416      1409      2      THEN ERR_EXIT (SS$_WRITLCK);
417      1410      2
418      1411      2      ! Do the create if requested. Start by allocating a file number from the
419      1412      2      ! index file bitmap and reading in the initial file header.
420      1413      2      !
421      1414      2
422      1415      2      IF .FUNCTION[IOSV_CREATE]
423      1416      2      THEN
424      1417      2          BEGIN
425      1418      2
426      1419      2      ! Deal with special cases related to create-if. Release any serialization
427      1420      2      ! lock we are holding, and force supersede mode to dispose of bad
428      1421      2      ! directory entries.
429      1422      2      !
430      1423      2
431      1424      2          IF .PACKET[IRPSV_FCODE] EQL IOSV_ACCESS
432      1425      2          THEN
433      1426      4              BEGIN

```

```

434      1427 4      IF .PRIM_LCKINDX NEQ 0
435      1428 4      THEN
436      1429 5          BEGIN
437      1430 5              RELEASE SERIAL_LOCK (.PRIM_LCKINDX);
438      1431 5              PRIM_LCKINDX = 0;
439      1432 4              END;
440      1433 4          FIB[FIB$V_SUPERSEDE] = 1;
441      1434 4
442      1435 4      ! Finally, the protection check if the directory has been accessed. This
443      1436 4      ! is because the protection check is not done in DIR_ACCESS (via ENTER) if
444      1437 4      ! the directory file has already been accessed.
445      1438 4
446      1439 4
447      1440 4          IF .DIR_FCB NEQ 0
448      1441 4          AND .CLEANUP_FLAGS[CLF_DIRECTORY]
449      1442 4          AND NOT .CLEANUP_FLAGS[CLF_SPOOLFILE]
450      1443 4          THEN
451      1444 5              BEGIN
452      1445 5                  STATUS = CHECK_PROTECT (WRITE_ACCESS, 0, .DIR_FCB, 0,
453      1446 5                  (IF .BBLOCK [FIB[FIB$C_ALT_ACCESS], ARMSV_DELETE]
454      1447 5                  THEN ARMSM_WRITE ELSE 0),
455      1448 5                  .FIB[FIB$V_ALT_REQ]);
456      1449 5
457      1450 5                  IF .STATUS EQL SSS_NOTALLPRIV
458      1451 5                  THEN FIB[FIB$V_ALT_GRANTED] = 0;
459      1452 4                  END;
460      1453 4          END;
461      1454 4      ! Handle any placement specified and find a suitable volume for the
462      1455 4      ! file in a volume set.
463      1456 4
464      1457 4
465      1458 4          FIB[FIB$V_PROPAGATE] = 0;          ! Since propagation is implied
466      1459 4          IF .FIB[FIB$V_ALLOCATR]
467      1460 4          THEN GET_LOC_ATTR (.ABD, .FIB);
468      1461 4          GET_LOC (.FIB, LOC_RVN, LOC_LBN);
469      1462 4          IF .LOC_RVN NEQ 0
470      1463 4          AND .FIB[FIB$V_EXACT]
471      1464 4          THEN
472      1465 4              SWITCH_VOLUME (.LOC_RVN)
473      1466 4          ELSE
474      1467 4              SELECT_VOLUME (.FIB, (IF .FIB[FIB$V_EXTEND]
475      1468 4              THEN .FIB[FIB$L_EXSZ]
476      1469 4              ELSE 0));
477      1470 4
478      1471 4          CHECK_PROTECT (CREATE_ACCESS, 0, 0, 0);          ! Check volume protection
479      1472 4          IF .BBLOCK [CURRENT_UCB[UCB$L_DEVCHAR], DEV$V_SWL]
480      1473 4          OR .CURRENT_VCB[VCB$V_NOALLOC]
481      1474 4          THEN ERR_EXIT (SS$WRITLCK);
482      1475 4
483      1476 4          HEADER = CREATE_HEADER (FIB[FIB$W_FID]);
484      1477 4
485      1478 4      ! Now build an initialized file header in the buffer.
486      1479 4
487      1480 4
488      1481 4          ARB = .PACKET[IRP$L_ARB];
489      1482 4
490      1483 4          IF .EXE$GL_DYNAMIC_FLAGS<EXE$V_CLASS_PROT,1>

```

```

491 1484 3 THEN HEADER[FH2$B_IDOFFSET] = FH2$C_FULL_LENGTH / 2
492 1485 ELSE HEADER[FH2$B_IDOFFSET] = FH2$C_LENGTH / 2;
493 1486 HEADER[FH2$B_MPOFFSET] = .HEADER[FH2$B_IDOFFSET] + F12$C_LENGTH / 2;
494 1487 HEADER[FH2$B_ACOFFSET] = ($BYTEOFFSET (FH2$W_CHECKSUM)) / 2;
495 1488 HEADER[FH2$B_RSOFFSET] = ($BYTEOFFSET (FH2$W_CHECKSUM)) / 2;
496 1489 HEADER[FH2$W_SEG_NUM] = 0;
497 1490 HEADER[FH2$W_STRUCLEV] = FH2$C_LEVEL2 + 1;
498 1491
499 1492 CH$FILL (0, 512 - $BYTEOFFSET(FH2$W_EXT_FID), HEADER[FH2$W_EXT_FID]);
500 1493 HEADER[FH2$L_FILEOWNER] = .ARB[ARB$C_UI];
501 1494 HEADER[FH2$W_FILEPROT] = .PCB[PCB$L_DEFPROT];
502 1495
503 1496 IF .FUNCTION[IOSV_DELETE]
504 1497 THEN HEADER[FH2$V_MARKDEL] = 1;
505 1498
506 1499 IF .CLEANUP_FLAGS[CLF_SPOOLFILE]
507 1500 THEN HEADER[FH2$V_SPOOL] = 1;
508 1501
509 1502 $ASSUME (ARB$S_CLASS EQL FH2$S_CLASS_PROT);
510 1503
511 1504 IF .EXE$GL_DYNAMIC_FLAGS<EXE$V_CLASS_PROT,1>
512 1505 THEN CH$MOVE (ARB$S_CLASS, ARB[ARB$R_CLASS], HEADER[FH2$R_CLASS_PROT]);
513 1506
514 1507 NEW FID = 0; ! new file ID is no longer unrecorded
515 1508 CLEANUP_FLAGS[CLF_DELFIL] = 1;
516 1509 CLEANUP_FLAGS[CLF_HDRNOTCHG] = 1;
517 1510 FILE HEADER = .HEADER; ! record header address for cleanup
518 1511 CHECKSUM (.HEADER);
519 1512
520 1513 ! At this point build the necessary FCB, even if the file is not accessed.
521 1514 ! This is necessary to allow the ACL to be built.
522 1515
523 1516
524 1517 FCB = KERNEL_CALL (CREATE_FCB, .HEADER);
525 1518 PRIMARY_FCB = .FCB;
526 1519 END;
527 1520
528 1521 ! If a non-zero directory ID is supplied, enter the file in the directory.
529 1522 ! Otherwise, just copy down the name string (if any) into the result string.
530 1523 ! Note that directory operations are also nooped on spool files operations.
531 1524
532 1525
533 1526 IF .CLEANUP_FLAGS[CLF_DIRECTORY] AND NOT .CLEANUP_FLAGS[CLF_SPOOLFILE]
534 1527 THEN
535 1528 BEGIN
536 1529 CH$FILL (0, FID$C_LENGTH, OLD_VERSION FID);
537 1530 ENTER (.ABD, .FIB, RESULT_LENGTH, RESULT);
538 1531
539 1532 ! Always attempt to release the allocation lock here. We will be holding
540 1533 ! it if the directory was extended. It might make more sense to release
541 1534 ! it in the directory extension, but the call is relatively cheap.
542 1535
543 1536
544 1537 ALLOCATION_UNLOCK ();
545 1538
546 1539 ! ENTER may have flushed the new buffer from the cache if either the
547 1540 ! directory file header(s) and quota file header(s) were accessed and

```

```

548 1541 3 | there were multiple headers. Make sure FILE_HEADER is what we think
549 1542 3 | it is.
550 1543 3 |
551 1544 3 |
552 1545 3 | IF .FUNCTION [IOSV_CREATE]
553 1546 3 | THEN
554 1547 3 | FILE_HEADER = READ_HEADER (0, .FCB);
555 1548 3 |
556 1549 3 | IF .FUNCTION[IOSV_CREATE] OR .FIB[FIBSV_PROPAGATE]
557 1550 3 | THEN
558 1551 4 | BEGIN
559 1552 4 |
560 1553 4 | ! If the CREATE modifier was not specified, then this must be a directory
561 1554 4 | entry operation. In which case it is necessary to actually access the
562 1555 4 | file being entered, so that an FCB will exist for the propagation to
563 1556 4 | occur.
564 1557 4 |
565 1558 4 | IF NOT .FUNCTION[IOSV_CREATE]
566 1559 4 | THEN
567 1560 5 | BEGIN
568 1561 5 |
569 1562 5 | ! Switch context to the volume of the specified RVN.
570 1563 5 |
571 1564 5 |
572 1565 5 | SWITCH_VOLUME (.FIB[FIBSW_FID_RVN]);
573 1566 5 |
574 1567 5 | ! Synchronize further processing on this file.
575 1568 5 |
576 1569 5 |
577 1570 5 | PRIM_LCKINDX = SERIAL_FILE (FIB [FIBSW_FID]);
578 1571 5 |
579 1572 5 | ! Find the FCB of the file, if one exists. then read the file
580 1573 5 | header. If there is no FCB, create one.
581 1574 5 |
582 1575 5 |
583 1576 5 | FCB = SEARCH_FCB (FIB[FIBSW_FID]);
584 1577 5 | HEADER = READ_HEADER (FIB[FIBSW_FID], .FCB);
585 1578 5 | FCB_CREATED = 0;
586 1579 5 |
587 1580 5 | IF .FCB EQL 0
588 1581 5 | THEN
589 1582 6 | BEGIN
590 1583 6 | FCB_CREATED = 1;
591 1584 6 | FCB = KERNEL_CALL (CREATE_FCB, .HEADER);
592 1585 5 | END;
593 1586 5 | PRIMARY_FCB = .FCB; ! record FCB for external use
594 1587 5 |
595 1588 5 | ! If the file is multi-header, read the extension headers and create
596 1589 5 | extension FCB's as necessary. Finally read back the primary header.
597 1590 5 |
598 1591 5 |
599 1592 5 | IF .FCB_CREATED
600 1593 5 | THEN
601 1594 5 | BUILD_EXT_FCBS (.HEADER)
602 1595 5 | ELSE
603 1596 5 | IF .FCB [FCBSV_STALE]
604 1597 5 | THEN

```

```

605      1598      6
606      1599
607      1600      6
608      1601      5
609      1602      5
610      1603      5 : Wipe out any acs that may have existed, because they are going
611      1604      5 : to be propagated.
612      1605      5
613      1606      5
614      1607      5      IF .BBLOCK [FCB [FCBSR_ORB], ORBSV_ACL_QUEUE]
615      1608      5      THEN
616      1609      5          ACL_DELETEACL (FCB [FCBSL_ACLFL], 0);
617      1610      5
618      1611      4      END;
619      1612      4
620      1613      4 ! Now propagate the file attributes to the file just entered.
621      1614      4
622      1615      4      STATUS = PROPAGATE_ATTR (.FIB);
623      1616      4      IF NOT .STATUS THEN ERR_EXIT (.STATUS);
624      1617      4      HEADER = .FILE_HEADER;
625      1618      4      HEADER[FH2SL_FILEOWNER] = .PRIMARY_FCB[FCBSL_FILEOWNER];
626      1619      4      HEADER[FH2SW_FILEPROT] = .PRIMARY_FCB[FCBSW_FILEPROT];
627      1620      4      CHECKSUM (.HEADER);
628      1621      4      MARK_DIRTY (.HEADER);
629      1622      4      END;
630      1623      3      END
631      1624      2      ELSE
632      1625      2      BEGIN
633      1626      2      KERNEL_CALL (COPY_NAME, .ABD);
634      1627      2      RESULT_LENGTH = MINU (.ABD[ABDSC_NAME, ABD$W_COUNT], FI2SS_FILENAME+FI2SS_FILENAMEEXT);
635      1628      2      CHSMOVE (.RESULT_LENGTH,
636      1629      2      .ABD[ABDSC_NAME, ABD$W_TEXT] + ABD[ABDSC_NAME, ABD$W_TEXT] + 1, RESULT);
637      1630      2      END;
638      1631      2
639      1632      2 ! Read the file header, regardless of the operation. Do a protection check
640      1633      2 ! on the directory pointed to by the present back link. If it is not valid,
641      1634      2 ! or if write access is allowed, then overwrite the back link with the new
642      1635      2 ! directory ID. Copy the file string into the header ident area. Then write
643      1636      2 ! attributes as specified.
644      1637      2
645      1638      2
646      1639      2      IF .FIB[FIB$W_FID_NUM] NEQ 65535
647      1640      2      OR .FIB[FIB$W_FID_SEQ] NEQ 65535
648      1641      2      OR .FIB[FIB$B_FID_NMX] NEQ 255
649      1642      2      THEN
650      1643      2      BEGIN
651      1644      2      PRIMARY_VCB = .CURRENT_VCB;
652      1645      2      IF .PRIMARY_VCB[VCB$W_RVN] NEQ 0
653      1646      2      THEN
654      1647      2      BEGIN
655      1648      2      UCB = .VECTOR [CURRENT_RVT[RVT$S_UCBLST], 0];
656      1649      2      IF .UCB EQL 0
657      1650      2      THEN ERR_EXIT (SS$ DEVNOTMOUNT);
658      1651      2      PRIMARY_VCB = .UCB[UCB$S_VCB];
659      1652      2      END;
660      1653      2
661      1654      2      IF .PRIM_LCKINDX EQL 0

```

```

662 1655 3 THEN
663 1656 3     PRIM_LCKINDX = SERIAL_FILE (FIB [FIB$W_FID]);
664 1657 3
665 1658 3 HEADER = READ_HEADER (FIB[FIB$W_FID], 0);
666 1659 3 IDENT_AREA = .HEADER + .HEADER[FH2$B_IDOFFSET]*2;
667 1660 3
668 1661 3 CHSMOVE (FID$C_LENGTH, HEADER[FH2$W_BACKLINK], PREV_LINK);
669 1662 3 IF .PREV_LINK[FID$W_NUM] EQL 0
670 1663 3 AND .PREV_LINK[FID$W_RVN] EQL 0
671 1664 3 THEN
672 1665 4     BEGIN
673 1666 4     IF NOT .CLEANUP_FLAGS[CLF_SPOOLFILE]
674 1667 4     THEN
675 1668 5         BEGIN
676 1669 5         CHSMOVE (FID$C_LENGTH, FIB[FIB$W_DID], HEADER[FH2$W_BACKLINK]);
677 1670 5         DEFAULT_RVN (HEADER[FH2$W_BK_FIDRVN], .CURRENT_RVN);
678 1671 5         CLEANUP_FLAGS[CLF_FIXLINK] = -1;
679 1672 4         END;
680 1673 4
681 1674 4     CHSMOVE (F12$S_FILENAME, IDENT_AREA[F12$T_FILENAME], PREV_INAME);
682 1675 4     CHSMOVE (F12$S_FILENAMEEXT, IDENT_AREA[F12$T_FILENAMEEXT],
683 1676 4         PREV_INAME[F12$S_FILENAME]);
684 1677 4     CHSCOPY (.RESULT_LENGTH, RESULT, ' ', F12$S_FILENAME, IDENT_AREA[F12$T_FILENAME]);
685 1678 4     IF .HEADER[FH2$B_MPOFFSET] - .HEADER[FH2$B_IDOFFSET]
686 1679 4     GEQU ($BYTEOFFSET (F12$T_FILENAMEEXT) + F12$S_FILENAMEEXT) / 2
687 1680 4     THEN
688 1681 5         BEGIN
689 1682 5         K = MAX (.RESULT_LENGTH - F12$S_FILENAME, 0);
690 1683 5         CHSCOPY (.K, RESULT[F12$S_FILENAME], ' ',
691 1684 5         F12$S_FILENAMEEXT, IDENT_AREA[F12$T_FILENAMEEXT]);
692 1685 4         END;
693 1686 4
694 1687 4 ! Update revision count and date and expiration date as appropriate.
695 1688 4 !
696 1689 4
697 1690 4     SET_REVISION (.HEADER, 3);
698 1691 3     END;
699 1692 3
700 1693 3 ! Set up file dates; then write the attributes.
701 1694 3 !
702 1695 3
703 1696 3 IF .FUNCTION[IOSV_CREATE]
704 1697 3 THEN
705 1698 4     BEGIN
706 1699 4     IDENT_AREA[F12$W_REVISION] = 0;
707 1700 4     CHSMOVE (F12$S_CREDATE, IDENT_AREA[F12$Q_REVDATE], IDENT_AREA[F12$Q_CREDATE]);
708 1701 4
709 1702 4     IF .PACKET[IRPSW_BCNT] GTR ABD$C_ATTRIB
710 1703 4     THEN
711 1704 5         BEGIN
712 1705 5         WRITE_ATTRIB (.HEADER, .ABD, 0);
713 1706 5         HEADER = .FILE_HEALER;
714 1707 4         END;
715 1708 4
716 1709 4 ! If the file is now owned by a UIC other than the creator, add an ACL
717 1710 4 ! entry granting owner's access to the creator. Then write the modified
718 1711 4 ! ACL into the header.

```

```

719      1712  4
720      1713  4      IF .HEADER[FH2$ FILEOWNER] NEQ .ARB[ARB$ UIC]
721      1714  4      AND NOT .CLEANUP_FLAGS[CLF_SYSPRV]
722      1715  4      THEN
723      1716  5          BEGIN
724      1717  5              ACL_INIT_QUEUE (PRIMARY_FCB[FCB$ ORB]);
725      1718  5              ACL_CONTEXT = 0;
726      1719  5              ACE[ACES$ SIZE] = ACE_LENGTH;
727      1720  5              ACE[ACES$ TYPE] = ACES$ KEYID;
728      1721  5              ACE[ACES$ FLAGS] = ACES$ NOPROPAGATE;
729      1722  5              ACE[ACES$ ACCESS] = ACES$ CONTROL OR
730      1723  5                  ( .HEADER[FH2$ FILEPROT]) <4,4> XOR %B'1111';
731      1724  5              ACE[ACES$ KEY] = .ARB[ARB$ UIC];
732      1725  5              ACL_ADDENTRY (PRIMARY_FCB[FCB$ ACLFL], ACL_CONTEXT, ACE_LENGTH, ACE);
733      1726  5              STATUS = ACL_BUILDACL (.PRIMARY_FCB);
734      1727  5              IF NOT .STATUS THEN ERR_EXIT (.STATUS);
735      1728  4          END;
736      1729  4
737      1730  4      CHARGE_QUOTA (.HEADER[FH2$ FILEOWNER], 1, BITLIST (QUOTA_CHECK, QUOTA_CHARGE));
738      1731  4      CLEANUP_FLAGS[CLF_HDRNOTCHG] = 0;
739      1732  4
740      1733  4      ! If access is requested, access the file.
741      1734  4      !
742      1735  4
743      1736  4      IF .FUNCTION[IOSV_ACCESS]
744      1737  4      THEN
745      1738  5          BEGIN
746      1739  5
747      1740  5              IF NOT ARBITRATE_ACCESS (.FIB [FIB$ ACCTL], .FCB)
748      1741  5              THEN
749      1742  5                  BUG_CHECK (XQPERR, 'how can we fail to access a new file?');
750      1743  5
751      1744  5              CURRENT_WINDOW = CREATE_WINDOW (.FIB[FIB$ ACCTL],
752      1745  5                  .FIB[FIB$ WSIZE], .HEADER, .PACKET[IRP$ PID], .FCB);
753      1746  5
754      1747  5              IF .CURRENT_WINDOW EQL 0
755      1748  5              THEN
756      1749  6                  BEGIN
757      1750  6
758      1751  6      ! This will dequeue the access lock we may have taken above (if a cluster
759      1752  6      ! device) because the refcnt will be zero.
760      1753  6      !
761      1754  6
762      1755  6                  CONV_ACCLOCK (0, .FCB);
763      1756  6                  ERR_EXIT (SS$ EXBYTLH);
764      1757  5                  END;
765      1758  5
766      1759  5              MAKE_ACCESS (.FCB, .CURRENT_WINDOW, .ABD);
767      1760  5
768      1761  5              IF .FUNCTION[IOSV_DELETE]
769      1762  5              THEN KERNEL_CALL (MARKDEL_FCB, .FCB);
770      1763  5              IF .(PRIMARY_VCB[VCB$ RETAINMAX]+4) NEQ 0
771      1764  5              THEN KERNEL_CALL (SET_EXPIRE);
772      1765  4          END;
773      1766  4
774      1767  4      ! Now extend the file if requested.
775      1768  4

```



```

: 776      1769  4
: 777      1770  4
: 778      1771  4
: 779      1772  4
: 780      1773  3
: 781      1774  3
: 782      1775  3
: 783      1776  3
: 784      1777  3
: 785      1778  4
: 786      1779  3
: 787      1780  3
: 788      1781  3
: 789      1782  3
: 790      1783  4
: 791      1784  4
: 792      1785  4
: 793      1786  3
: 794      1787  3
: 795      1788  3
: 796      1789  3
: 797      1790  3
: 798      1791  3
: 799      1792  3
: 800      1793  3
: 801      1794  2
: 802      1795  2
: 803      1796  2
: 804      1797  2
: 805      1798  2
: 806      1799  2
: 807      1800  2
: 808      1801  2
: 809      1802  2
: 810      1803  2
: 811      1804  3
: 812      1805  3
: 813      1806  3
: 814      1807  3
: 815      1808  3
: 816      1809  3
: 817      1810  3
: 818      1811  3
: 819      1812  2
: 820      1813  2
: 821      1814  2
: 822      1815  2
: 823      1816  2
: 824      1817  1

```

```

      IF .FIB[FIBSV_EXTEND] THEN EXTEND (.FIB, .HEADER);
      HEADER = .FILE HEADER;
      KERNEL_CALL (UPDATE_FCB, .HEADER);
      END;

      CHECKSUM (.HEADER);
      MARK_DIRTY (.HEADER);

      IF (.FUNCTION[IOSV_CREATE] OR .FIB[FIBSV_PROPAGATE])
      AND .PRIMARY_FCB NEQ 0
      THEN
      IF .BBLOCK[PRIMARY_FCB[FCBSR_ORB], ORBSV_ACL_QUEUE]
      THEN
      BEGIN
      STATUS = ACL_BUILDACL (.PRIMARY_FCB);
      IF NOT .STATUS THEN ERR_EXIT (.STATUS);
      END;

      ! Perform the remap operation if necessary to account for any initial extend.

      IF .FUNCTION[IOSV_ACCESS] AND .FIB[FIBSV_EXTEND]
      THEN IF .CURRENT_WINDOW[WCBV_CATHEDRAL]
      THEN REMAP_FILE (?);
      END;

      ! If this is a supersede operation, delete the file that was removed during
      ! the enter operation above. This must be done last since we cannot undo
      ! a delete in cleaning up from a subsequent error. We first copy the primary
      ! context into the context save area since this is a secondary operation.

      IF .CLEANUP_FLAGS[CLF_SUPERSEDE]
      THEN
      BEGIN
      ALLOCATION_UNLOCK ();
      SAVE_CONTEXT ();
      CHSCOPY (FIDSC_LENGTH, SUPER_FID, 0,
      FIBSC_LENGTH - $BYTEOFFSET (FIBSW_FID), SECOND_FIB[FIBSW_FID]);
      SECOND_FIB[FIBSB_AGENT_MODE] = .FIB[FIBSB_AGENT_MODE];
      MARK_DELETE (SECOND_FIB, 1, 0, 0);
      RESTORE_CONTEXT ();
      END;

      RETURN 1;

      END;

```

! end of routine CREATE

```

.TITLE CREATE
.IDENT \V04-002\

.EXTRN ACPSGB_WRITBACK
.EXTRN SCH$GL_PCBVEC, EXESGL_DYNAMIC_FLAGS
.EXTRN EXESV_CLASS_PROT

```

```
.EXTRN ACL_DELETEACL, UPDATE_FCB
.EXTRN REBCD PRIM_FCB, BUILD_EXT_FCBS
.EXTRN RELEASE_SERIAL_LOCK
.EXTRN ALLOCATION_UNLOCK
.EXTRN ARBITRATE_ACCESS
.EXTRN CONV_ACCLOCK, SERIAL_FILE
.EXTRN GET_FIB, GET_LOC_ATTR
.EXTRN GET_LOC, SWITCH_VOLUME
.EXTRN SELECT_VOLUME, CHECK_PROTECT
.EXTRN CHARGE_QUOTA, CREATE_HEADER
.EXTRN CHECKSUM, MARK_DIRTY
.EXTRN ACL_INIT_QUEUE, ACL_ADDENTRY
.EXTRN ACL_BUILDACL, READ_HEADER
.EXTRN ENTER, COPY_NAME
.EXTRN SET_REVISION, CREATE_FCB
.EXTRN CREATE_WINDOW, SET_EXPIRE
.EXTRN MAKE_ACCESS, MARKDEL_FCB
.EXTRN WRITE_ATTRIB, EXTEND
.EXTRN SAVE_CONTEXT, RESTORE_CONTEXT
.EXTRN MARK_DELETE, REMAP_FILE
.EXTRN SEARCH_FCB, BUGS_XOPERR
```

.PSECT \$CODE\$,NOWRT,2

			OBFC 00000	.ENTRY CREATE, Save R2,R3,R4,R5,R6,R7,R8,R9,R11	: 1256
	SE	80	AE 9E 00002	MOVAB -128(SP), SP	:
		04	AA 9F 00006	PUSHAB 4(BASE)	: 1332
		08	AA 9F 00009	PUSHAB 8(BASE)	:
	59	18	AA 9E 0000C	MOVAB 24(BASE), R9	:
		30	AA 9F 00010	PUSHAB 48(BASE)	:
		01AB	CA 9F 00013	PUSHAB 424(BASE)	:
		0244	CA 9F 00017	PUSHAB 580(BASE)	:
		90	AA DD 0001B	PUSHL -112(BASE)	: 1380
50	6E		20 C1 0001E	ADDL3 #32, PACKET, RO	: 1381
	7E		60 3C 00022	MOVZWL (RO), FUNCTION	:
06	6E		06 E1 00025	BBC #6, FUNCTION, 1\$: 1382
	02	AA	0402 8F AB 00029	BISW2 #1026, 2(BASE)	: 1386
		51	00000000G 9F D0 0002F	1\$: MOVL @#SCH\$GL_PCBVEC, R1	: 1392
		50	90 AA D0 00036	MOVL -112(BASE), RO	:
		50	0C C0 0003A	ADDL2 #12, RO	:
		50	60 3C 0003D	MOVZWL (RO), RO	:
		58	6140 D0 00040	MOVL (R1)(RO), PCB	:
50	04	AE	2C C1 00044	ADDL3 #44, PACKET, RO	: 1393
		56	90 D0 00049	MOVL @(RO)+, ABD	:
		56	DD 0004C	PUSHL ABD	: 1395
	0000G	CF	01 FB 0004E	CALLS #1, GET_FIB	:
		57	50 D0 00053	MOVL RO, FIB	:
		27	17 A7 E8 00056	BLBS 23(FIB), 3\$: 1397
	7FFF	8F	2C A7 B1 0005A	CMPW 44(FIB), #32767	: 1398
			1F 1A 00060	BGTRU 3\$:
		04	01 AE E9 00062	BLBC FUNCTION+1, 2\$: 1399
17		6E	06 E1 00066	BBC #6, FUNCTION, 3\$:
			6E 95 0006A	2\$: TSTB FUNCTION	: 1400
			16 19 0006C	BLSS 4\$:
			16 A7 95 0006E	TSTB 22(FIB)	: 1401
			0E 19 00071	BLSS 3\$:
50	04	AE	32 C1 00073	ADDL3 #50, PACKET, RO	: 1402

		05	60	B1	00078		CMPW	(R0), #5	
			04	1A	0007B		BGTRU	3\$	
	03	6E	06	E1	0007D		BBC	#6, FUNCTION, 4\$	1403
			14	BF	00081	3\$:	CHMU	#20	1406
				04	00083		RET		
		50	98	AA	D0	00084	4\$:	MOVL	-104(BASE), R0
	03	0B	A0	04	E1	00088		BBC	#4, 11(R0), 5\$
				00C2	31	0008D		BRW	16\$
				6E	95	00090	5\$:	TSTB	FUNCTION
				03	19	00092		BLSS	6\$
				015E	31	00094		BRW	23\$
	50	04	AE	20	C1	00097	6\$:	ADDL3	#32, PACKET, R0
32	60	06		00	ED	0009C		CMPZV	#0, #6, (R0), #50
				4F	12	000A1		BNEQ	10\$
				69	D5	000A3		TSTL	(R9)
				09	13	000A5		BEQL	7\$
				69	DD	000A7		PUSHL	(R9)
		0000G	CF	01	FB	000A9		CALLS	#1, RELEASE_SERIAL_LOCK
				69	D4	000AE		CLRL	(R9)
		15	A7	04	88	000B0	7\$:	BISB2	#4, 21(FIB)
			50	CA	D0	000B4		MOVL	208(BASE), R0
				37	13	000B9		BEQL	10\$
	33	6A		06	E1	000BB		BBC	#6, (BASE), 10\$
				6A	95	000BF		TSTB	(BASE)
				2F	19	000C1		BLSS	10\$
7E	38	A7	01	00	EF	000C3		EXTZV	#0, #1, 56(FIB), -(SP)
		04	3C	03	E1	000C9		BBC	#3, 60(FIB), 8\$
				02	DD	000CE		PUSHL	#2
				02	11	000D0		BRB	9\$
				7E	D4	000D2	8\$:	CLRL	-(SP)
				7E	D4	000D4	9\$:	CLRL	-(SP)
				50	DD	000D6		PUSHL	R0
		7E		01	7D	000D8		MOVQ	#1, -(SP)
		0000G	CF	06	FB	000DB		CALLS	#6, CHECK_PROTECT
		24	AE	50	D0	000E0		MOVL	R0, STATUS
		00000681	8F	24	AE	000E4		CMPL	STATUS, #1665
				04	12	000EC		BNEQ	10\$
		38	A7	02	8A	000EE		BICB2	#2, 56(FIB)
		38	A7	08	8A	000F2	10\$:	BICB2	#8, 56(FIB)
08		16	A7	04	E1	000F6		BBC	#4, 22(FIB); 11\$
			7E	56	7D	000FB		MOVQ	ABD, -(SP)
		0000G	CF	02	FB	000FE		CALLS	#2, GET_LOC_ATTR
				20	AA	9F	00103	11\$:	PUSHAB
				1C	AA	9F	00106		PUSHAB
					57	DD	00109		PUSHL
		0000G	CF	03	FB	0010B		CALLS	#3, GET_LOC
				1C	AA	D5	00110		TSTL
					0E	13	00113		BEQL
			0A	20	A7	E9	00115		BLBC
				1C	AA	DD	00119		PUSHL
		0000G	CF	01	FB	0011C		CALLS	#1, SWITCH_VOLUME
					13	11	00121		BRB
				16	A7	95	00123	12\$:	TSTB
					05	18	00126		BGEQ
				18	A7	DD	00128		PUSHL
					02	11	0012B		BRB
					7E	D4	0012D	13\$:	CLRL
									-(SP)

					57 DD 0012F 14\$:	PUSHL FIB		
	0000G	CF			02 FB 00131	CALLS #2, SELECT_VOLUME		
		7E			7E 7C 00136 15\$:	CLRQ -(SP)		1471
	0000G	CF			03 7D 00138	MOVQ #3, -(SP)		
		50	94		04 FB 0013B	CALLS #4, CHECK_PROTECT		
09	3B	A0			01 E0 00140	MOVL -108(BASE), R0		1472
		50	98		01 E0 00144	SBS #1, 59(R0), 16\$		
05	0B	A0			04 E1 00149	MOVL -104(BASE), R0		1473
			025C		04 E1 0014D	BBC #4, 11(R0), 17\$		
					8F BF 00152 16\$:	CHMU #604		1474
					04 04 00156	RET		
	0000G	CF	04		A7 9F 00157 17\$:	PUSHAB 4(FIB)		1476
		58			01 FB 0015A	CALLS #1, CREATE_HEADER		
50	04	AE	00000058		50 D0 0015F	MOVL R0, HEADER		
	1C	AE			8F C1 00162	ADDL3 #88, PACKET, R0		1481
05	00000000G	9F	00000000G		60 D0 0016B	MOVL (R0), ARB		
					8F E1 0016F	BBC #EXE\$V_CLASS_PROT, @#EXE\$GL_DYNAMIC_FLAGS, -18\$		1483
					36 90 0017B	MOVB #54, (HEADER)		1484
					03 11 0017E	BRB 19\$		
01	A8				28 90 00180 18\$:	MOVB #40, (HEADER)		1485
					3C 81 00183 19\$:	ADDB3 #60, (HEADER), 1(HEADER)		1486
	02	A8	FFFF		8F 3C 00188	MOVZWL #65535, 2(HEADER)		1487
	06	A8	0201		8F B0 0018E	MOVW #513, 6(HEADER)		1490
01F2	8F				00 2C 00194	MOVCS #0, (SP), #0, #498, 14(HEADER)		1492
					0E A8 0019B			
	50	1C	AE		38 C1 0019D	ADDL3 #56, ARB, R0		1493
		3C	A8		60 D0 001A2	MOVL (R0), 60(HEADER)		
		40	A8	0114	CB B0 001A6	MOVW 276(PCB), 64(HEADER)		1494
			05	01	AE E9 001AC	BLBC FUNCTION+1, 20\$		1496
				80	8F 88 001B0	BISB2 #128, 53(HEADER)		1497
					6A 95 001B5 20\$:	TSTB (BASE)		1499
					04 18 001B7	BGEQ 21\$		
					10 88 001B9	BISB2 #16, 53(HEADER)		1500
	0A	00000000G	9F	00000000G	8F E1 001BD 21\$:	BBC #EXE\$V_CLASS_PROT, @#EXE\$GL_DYNAMIC_FLAGS, -22\$		1504
					0C C1 001C9	ADDL3 #12, ARB, R11		1505
58	5B	1C	AE		14 28 001CE	MOVCS #20, (R11), 88(HEADER)		
	A8		6B	A8	AA D4 001D3 22\$:	CLRL -88(BASE)		1507
				0820	8F AB 001D5	BISW2 #2080, 2(BASE)		1509
		02	AA		58 D0 001DC	MOVL HEADER, @24(SP)		1510
		18	BE		58 DD 001E0	PUSHL HEADER		1511
					01 FB 001E2	CALLS #1, CHECKSUM		
	0000G	CF			58 DD 001E7	PUSHL HEADER		1517
					01 FB 001E9	CALLS #1, CREATE_FCB		
	0000G	CF			50 D0 001EE	MOVL R0, FCB		
		5B			5B D0 001F1	MOVL FCB, @20(SP)		1518
		14	BE		06 E0 001F5 23\$:	BBS #6, (BASE), 25\$		1526
03		6A			00E7 31 001F9 24\$:	BRW 33\$		
					6A 95 001FC 25\$:	TSTB (BASE)		
					F9 19 001FE	BLSS 24\$		
	06				00 2C 00200	MOVCS #0, (SP), #0, #6, 332(BASE)		1529
				014C	CA 00205			
				44	AE 9F 00208	PUSHAB RESULT		1530
				2C	AE 9F 0020B	PUSHAB RESULT_LENGTH		
					56 7D 0020E	MOVQ ABD, -(SP)		
	0000G	7E			04 FB 00211	CALLS #4, ENTER		
		CF						

	0000G	CF		00	FB	00216	CALLS	#0, ALLOCATION_UNLOCK	:	1537	
				6E	95	00218	TSTB	FUNCTION	:	1545	
				0D	18	0021D	BGEQ	26\$:		
				5B	DD	0021F	PUSHL	FCB	:	1547	
				7E	D4	00221	CLRL	-(SP)	:		
	0000G	CF		02	FB	00223	CALLS	#2, READ_HEADER	:		
	18	BE		50	DD	00228	MOVL	R0, @24(SP)	:		
				6E	95	0022C	TSTB	FUNCTION	:	1549	
				08	19	0022E	BLSS	27\$:		
03	38	A7		03	F0	00230	BBS	#3, 56(FIB), 27\$:		
				00D4	31	00235	BRW	35\$:		
				6E	95	00238	TSTB	FUNCTION	:	1558	
				6F	19	0023A	BLSS	31\$:		
		7E	08	A7	3C	0023C	MOVZWL	8(FIB), -(SP)	:	1565	
	0000G	CF		01	FB	00240	CALLS	#1, SWITCH_VOLUME	:		
				04	A7	9F	PUSHAB	4(FIB)	:	1570	
	0000G	CF		01	FB	00248	CALLS	#1, SERIAL_FILE	:		
		69		50	DD	0024D	MOVL	R0, (R9)	:		
				04	A7	9F	PUSHAB	4(FIB)	:	1576	
	00000000G	00		01	FB	00253	CALLS	#1, SEARCH_FCB	:		
		5B		50	DD	0025A	MOVL	R0, FCB	:		
				5B	DD	0025D	PUSHL	FCB	:	1577	
				04	A7	9F	PUSHAB	4(FIB)	:		
	0000G	CF		02	FB	00262	CALLS	#2, READ_HEADER	:		
		58		50	DD	00267	MOVL	R0, HEADER	:		
				52	D4	0026A	CLRL	FCB_CREATED	:	1578	
				5B	D5	0026C	TSTL	FCB_CREATED	:	1580	
				0D	12	0026E	BNEQ	28\$:		
		52		01	DD	00270	MOVL	#1, FCB_CREATED	:	1583	
				58	DD	00273	PUSHL	HEADER	:	1584	
	0000G	CF		01	FB	00275	CALLS	#1, CREATE_FCB	:		
		5B		50	DD	0027A	MOVL	R0, FCB	:		
	14	BE		5B	DD	0027D	MOVL	FCB, @20(SP)	:	1586	
		OE		52	E8	00281	BLBS	FCB_CREATED, 29\$:	1592	
		11	23	AB	E9	00284	BLBC	35(FCB), 30\$:	1596	
				58	DD	00288	PUSHL	HEADER	:	1599	
				18	BE	DD	0028A	PUSHL	@24(SP)	:	
	0000G	CF		02	FB	0028D	CALLS	#2, REBLD_PRIM_FCB	:		
				58	DD	00292	PUSHL	HEADER	:	1600	
	0000G	CF		01	FB	00294	CALLS	#1, BUILD_EXT_FCBS	:		
0D		63	AB	01	E1	00299	BBC	#1, 99(FCB), 31\$:	1607	
				7E	D4	0029E	CLRL	-(SP)	:	1609	
			0080	CB	9F	002A0	PUSHAB	128(FCB)	:		
	00000000G	00		02	FB	002A4	CALLS	#2, ACL_DELETEACL	:		
				57	DD	002AB	PUSHL	FIB	:	1614	
	0000V	CF		01	FB	002AD	CALLS	#1, PROPAGATE_ATTR	:		
		AE		50	DD	002B2	MOVL	R0, STATUS	:		
		03	24	AE	E8	002B6	BLBS	STATUS, 32\$:	1616	
				027E	31	002BA	BRW	55\$:		
		58	18	BE	DD	002BD	MOVL	@24(SP), HEADER	:	1617	
		50	14	BE	DD	002C1	MOVL	@20(SP), R0	:	1618	
	3C	AB	58	A0	DD	002C5	MOVL	88(R0), 60(HEADER)	:		
		50	14	BE	DD	002CA	MOVL	@20(SP), R0	:	1619	
	40	AB	7C	A0	BD	002CE	MOVW	112(R0), 64(HEADER)	:		
				58	DD	002D3	PUSHL	HEADER	:	1620	
	0000G	CF		01	FB	002D5	CALLS	#1, CHECKSUM	:		
				58	DD	002DA	PUSHL	HEADER	:	1621	

		0000G	CF		01	FB	002DC		CALLS	#1, MARK_DIRTY		
					29	11	002E1		BRB	35\$		1526
					56	DD	002E3	33\$:	PUSHL	ABD		1626
		0000G	CF		01	FB	002E5		CALLS	#1, COPY_NAME		
			50		A6	3C	002EA		MOVZWL	18(ABD), -R0		1627
		0056	8F		50	B1	002EE		CMPW	R0, #86		
					04	1B	002F3		BLEQU	34\$		
			50		56	8F	9A	002F5	MOVZBL	#86, R0		
			AE		50	DD	002F9	34\$:	MOVL	R0, RESULT_LENGTH		
					10	A6	9E	002FD	MOVAB	16(ABD), RT		1629
					50	61	3C	00301	MOVZWL	(R1), R0		
44	AE	01	A140		28	AE	28	00304	MOVCS	RESULT_LENGTH, 1(R1)[R0], RESULT		1628
		FFFF	8F		04	A7	B1	0030C	35\$:	CMPW	4(FIB), #65535	1639
					12	12	00312		BNEQ	36\$		
		FFFF	8F		06	A7	B1	00314	CMPW	6(FIB), #65535		1640
					0A	12	0031A		BNEQ	36\$		
					09	A7	91	0031C	CMPB	9(FIB), #255		1641
					03	12	00321		BNEQ	36\$		
					0230	31	00323		BRW	57\$		
		20	AE		98	AA	DD	00326	36\$:	MOVL	-104(BASE), PRIMARY_VCB	1644
		50	AE			0E	C1	0032B	ADDL3	#14, PRIMARY_VCB, R0		1645
					60	B5	00330		TSTW	(R0)		
					14	13	00332		BEQL	38\$		
			50		9C	AA	DD	00334	MOVL	-100(BASE), R0		1648
			51		44	A0	DD	00338	MOVL	68(R0), UCB		
					05	12	0033C		BNEQ	37\$		1649
				007C	8F	BF	0033E		CHMU	#124		1650
						04	0C342		RET			
		20	AE		34	A1	DD	00343	37\$:	MOVL	52(UCB), PRIMARY_VCB	1651
					69	D5	00348	38\$:	TSTL	(R9)		1654
					0B	12	0034A		BNEQ	39\$		
					04	A7	9F	0034C	PUSHAB	4(FIB)		1656
		0000G	CF		01	FB	0034F		CALLS	#1, SERIAL_FILE		
			69		50	DD	00354		MOVL	R0, (R9)		
					7E	D4	00357	39\$:	CLRL	-(SP)		1658
					04	A7	9F	00359	PUSHAB	4(FIB)		
		0000G	CF		02	FB	0035C		CALLS	#2, READ HEADER		
			58		50	DD	00361		MOVL	R0, HEADER		
			50		68	9A	00364		MOVZBL	(HEADER), R0		1659
			59		6840	3E	00367		MOVAB	(HEADER)[R0], IDENT_AREA		
10	BE	42	AB		06	28	0036B		MOVCS	#6, 66(HEADER), @16(SP)		1661
					10	BE	B5	00371	TSTW	@16(SP)		1662
					68	12	00374		BNEQ	44\$		
			50		10	AE			ADDL3	#4, 16(SP), R0		1663
					60	B5	0037B		TSTW	(R0)		
					5F	12	0037D		BNEQ	44\$		
					6A	95	0037F		TSTB	(BASE)		1666
					17	19	00381		BLSS	41\$		
			42	AB	0A	A7			MOVCS	#6, 10(FIB), 64(HEADER)		1669
A0	AA	46	AB		00	ED	00389		CMPZV	#0, #8, 70(HEADER), -96(BASE)		1670
					03	12	00390		BNEQ	40\$		
					46	AB	94	00392	CLRB	70(HEADER)		
					40	8F	88	00395	40\$:	BISB2	#64, 3(BASE)	1671
			OC	BE	69				41\$:	MOVCS	#20, (IDENT_AREA), @12(SP)	1674
				7E	OC	AE			ADDL3	#20, 12(SP), -(SP)		1676
				9E	36	A9			MOVCS	#66, 54(IDENT_AREA), @1(SP)+		
		14	AE		0042	28	AE	2C	003AB	MOVCS	RESULT_LENGTH, RESULT, #32, #20, -	1677

					69				003B2	(IDENT AREA)				
					50	01	A8	9A	003B3	MOVZBL	1(HEADER), R0	1678		
					51		68	9A	003B7	MOVZBL	(HEADER), R1			
					50		51	C2	003BA	SUBL2	R1, R0			
					3C		50	D1	003BD	CMPL	R0, #60	1679		
							13	1F	003C0	BLSSU	43\$			
				50	28	AE	14	C3	003C2	SUBL3	#20, RESULT_LENGTH, R0	1682		
							02	18	003C7	BGEQ	42\$			
							50	D4	003C9	CLRL	R0			
0042	8F			20	58	AE	50	2C	003CB	42\$:	MOVCS	K, RESULT+20, #32, #66, 54(IDENT_AREA)	1684	
							36	A9	003D3					
								03	DD	003D5	43\$:	PUSHL	#3	1690
								58	DD	003D7		PUSHL	HEADER	
					0000G	CF	02	FB	003D9	CALLS	#2, SET REVISION			
							6E	95	003DE	44\$:	TSTB	FUNCTION	1696	
							03	19	003E0	BLSS	45\$			
							01	31	003E2	BRW	53\$			
							14	A9	B4	003E5	45\$:	CLRW	20(IDENT AREA)	1699
16	A9	1E	A9				08	28	003E8	MOVCS	#8, 30(IDENT_AREA), 22(IDENT_AREA)	1700		
	50	04	AE				32	C1	003EE	ADDL3	#50, PACKET, -R0	1702		
			05				60	B1	003F3	CMPL	(R0), #5			
							0F	1B	003F6	BLEQU	46\$			
							7E	D4	003F8	CLRL	-(SP)	1705		
							56	DD	003FA	PUSHL	ABD			
							58	DD	003FC	PUSHL	HEADER			
					0000G	CF	03	FB	003FE	CALLS	#3, WRITE ATTRIB			
							58	BE	00403	18	MOVL	@24(SP), HEADER	1706	
				50	1C	AE	38	C1	00407	46\$:	ADDL3	#56, ARB, R0	1713	
							60	A8	0040C	CMPL	60(HEADER), (R0)			
							63	13	00410	BEQL	47\$			
							01	AA	E8	00412	BLBS	1(BASE), 47\$	1714	
							7E	8F	C1	00416	ADDL3	#88, @20(SP), -(SP)	1717	
					00000000G	00	01	FB	0041F	CALLS	#1, ACL_INIT_QUEUE			
							2C	AE	D4	00426	CLRL	ACL_CONTEXT -	1718	
							8F	DO	00429	MOVL	#13217996, ACE	1719		
							04	EF	00431	EXTZV	#4, #4, 64(HEADER), R0	1723		
							0F	CC	00437	XORL2	#15, R0			
							10	C9	0043A	BISL3	#16, R0, ACE+4	1722		
							38	C1	0043F	ADDL3	#56, ARB, R0	1724		
							60	DO	00444	MOVL	(R0), ACE+8			
							30	AE	9F	00448	PUSHAB	ACE	1725	
							0C	DD	0044B	PUSHL	#12			
							34	AE	9F	0044D	PUSHAB	ACL_CONTEXT		
							7E	8F	C1	00450	ADDL3	#128, @32(SP), -(SP)		
					00000000G	00	04	FB	00459	CALLS	#4, ACL_ADDENTRY			
							14	BE	DD	00460	PUSHL	@20(SP)	1726	
					00000000G	00	01	FB	00463	CALLS	#1, ACL_BUILDACL			
							50	DO	0046A	MOVL	R0, STATUS			
							24	AE	E8	0046E	BLBS	STATUS, 47\$	1727	
							00	C6	31	00472	BRW	55\$		
							03	DD	00475	47\$:	PUSHL	#3	1730	
							01	DD	00477	PUSHL	#1			
							3C	A8	DD	00479	PUSHL	60(HEADER)		
					0000G	CF	03	FB	0047C	CALLS	#3, CHARGE QUOTA			
							08	8A	00481	BICB2	#8, 3(BASET	1731		
							06	E1	00485	BBC	#6, FUNC*ION, 51\$	1736		
63							58	DO	00489	MOVL	FCB, R1	1740		

		50		67	DO	0048C		MOVL	(FIB), R0	
				0000G	30	0048F		BSBW	ARBITRATE_ACCESS	
		04		50	EB	00492		BLBS	R0, 48\$	
					FEFF	00495		BUGW		1742
					0000*	00497		.WORD	<BUG\$_XQPERR!4>	
				5B	DD	00499	48\$:	PUSHL	FCB	1745
2	08	AE		0C	C1	0049B		ADDL3	#12, PACKET, R2	
				62	DD	004A0		PUSHL	(R2)	
				5B	DD	004A2		PUSHL	HEADER	
		7E	03	A7	98	004A4		CVTBL	3(FIB), -(SP)	
				67	DD	004A8		PUSHL	(FIB)	1744
		0000G	CF	05	FB	004AA		CALLS	#5, CREATE_WINDOW	
		0C	AA	50	DO	004AF		MOVL	R0, 12(BASE)	
				0E	12	004B3		BNEQ	49\$	1747
				5B	DD	004B5		PUSHL	FCB	1755
				7E	D4	004B7		(LRL	-(SP)	
		0000G	CF	02	FB	004B9		CALLS	#2, CONV_ACCLOCK	
				2A14	8F	004BE		CHMU	#10772	1756
					04	004C2		RET		
				56	DD	004C3	49\$:	PUSHL	ABD	1759
			0C	AA	DD	004C5		PUSHL	12(BASE)	
				5B	DD	004C8		PUSHL	FCB	
		0000G	CF	03	FB	004CA		CALLS	#3, MAKE_ACCESS	
			07	01	AE	E9	004CF	BLBC	FUNCTION#1, 50\$	1761
				5B	DD	004D3		PUSHL	FCB	1762
50		0000G	CF	01	FB	004D5		CALLS	#1, MARKDEL_FCB	
		20	AE	00000078	8F	C1	004DA	50\$:	ADDL3	#120, PRIMARY_VCB, R0
				60	D5	004E3		TSTL	(R0)	1763
				05	13	004E5		BFQL	51\$	
		0000G	CF	00	FB	004E7		CALLS	#0, SET_EXPIRE	1764
				16	A7	95	004EC	51\$:	TSTB	22(FIB)-
					08	18	004EF		BGEQ	52\$
			7E	57	7D	004F1		MOVQ	FIB, -(SP)	
		0000G	CF	02	FB	004F4		CALLS	#2, EXTEND	
			58	18	BE	DO	004F9	52\$:	MOVL	@24(SP), HEADER
				5B	DD	004FD		PUSHL	HEADER	1771
		0000G	CF	01	FB	004FF		CALLS	#1, UPDATE_FCB	1772
				58	DD	00504	53\$:	PUSHL	HEADER	
		0000G	CF	01	FB	00506		CALLS	#1, CHECKSUM	1775
				58	DD	0050B		PUSHL	HEADER	1776
		0000G	CF	01	FB	0050D		CALLS	#1, MARK_DIRTY	
				6E	95	00512		TSTB	FUNCTION-	1778
				05	19	00514		BLSS	54\$	
24		38	A7	03	E1	00516		BBC	#3, 56(FIB), 56\$	
				14	BE	D5	0051B	54\$:	TSTL	@20(SP)
					1F	13	0051E		BEQ	56\$
			50	14	BE	DO	00520		MOVL	@20(SP), R0
16		63	A0	01	E1	00524		BBC	#1, 99(R0), 56\$	1781
				14	BE	DD	00529		PUSHL	@20(SP)
		00000000G	00	01	FB	0052C		CALLS	#1, ACL_BUILDACL	1784
		24	AE	50	DO	00533		MOVL	R0, STATUS	
			04	24	AE	E8	00537		BLBS	STATUS, 56\$
				24	AE	BF	0053B	55\$:	CHMU	STATUS
					04	0053E		RET		
13		6E		06	E1	0053F	56\$:	BBC	#6, FUNCTION, 57\$	1791
				16	A7	95	00543		TSTB	22(FIB)
					0E	18	00546		BGEQ	57\$

CREATE
V04-002

H 5
8-Jan-1985 17:42:14 VAX-11 Bliss-32 V4.0-742
2-Oct-1984 12:43:26 [F11X.BUGSRC]CREATE.B32;1

Page 23
(2)

		50	OC	AA	DO	00548	MOVL	12(BASE), R0	: 1792
05	0B	A0		06	E1	C054C	BBC	#6, 11(R0), 57\$: 1793
	0000G	CF		00	FB	00551	CALLS	#0, REMAP FILE	: 1802
31		6A		05	E1	00556	BBC	#5, (BASE), 58\$: 1805
	0000G	CF		00	FB	0055A	CALLS	#0, ALLOCATION_UNLOCK	: 1806
	0000G	CF		00	FB	0055F	CALLS	#0, SAVE_CONTEXT	: 1808
56	0B	AE		04	C1	00564	ADDL3	#4, 8(SPT), R6	: 1809
3C	00	01FE	CA	06	2C	00569	MOVCS	#6, 510(BASE), #0, #60, (R6)	: 1810
				66		00570			: 1811
50	0B	AE	2E	2E	C1	00571	ADDL3	#46, 8(SP), R0	: 1815
		60		A7	90	00576	MOVB	46(FIB), (R0)	: 1817
				7E	7C	0057A	CLRQ	-(SP)	: 1811
			14	01	DD	0057C	PUSHL	#1	: 1815
	0000G	CF		AE	DD	0057E	PUSHL	20(SP)	: 1817
	0000G	CF		04	FB	00581	CALLS	#4, MARK_DELETE	: 1811
		50		00	FB	00586	CALLS	#0, RESTORE_CONTEXT	: 1815
				01	DO	0058B	MOVL	#1, R0	: 1817
				04	0058E		RET		: 1817

; Routine Size: 1423 bytes, Routine Base: \$CODE\$ + 0000

```

826 1818 1 ROUTINE PROPAGATE_ATTR (FIB) : L_NORM =
827 1819 1
828 1820 1 **
829 1821 1
830 1822 1 FUNCTIONAL DESCRIPTION:
831 1823 1
832 1824 1 This routine is called to propagate the file attributes from one
833 1825 1 file to another. This may be from one version of a file to another
834 1826 1 version of the file (either higher or lower) or from the parent
835 1827 1 directory to the newly created file. The following attributes are
836 1828 1 currently copied:
837 1829 1 1) File owner UIC
838 1830 1 2) File Access Control List (ACL)
839 1831 1 3) File protection (With some twiddling)
840 1832 1
841 1833 1 CALLING SEQUENCE:
842 1834 1 PROPAGATE_ATTR (ARG1)
843 1835 1
844 1836 1 INPUT PARAMETERS:
845 1837 1 ARG1: address of the supplied FIB
846 1838 1
847 1839 1 IMPLICIT INPUTS:
848 1840 1 PRIMARY_FCB: address of the new file's FCB
849 1841 1 DIR_FCB: address of the directory file's FCB
850 1842 1 OLD_VERSION_FID: FID of the old version of the file
851 1843 1
852 1844 1 OUTPUT PARAMETERS:
853 1845 1 none
854 1846 1
855 1847 1 IMPLICIT OUTPUTS:
856 1848 1 none
857 1849 1
858 1850 1 ROUTINE VALUE:
859 1851 1 1 if success
860 1852 1 error code otherwise
861 1853 1
862 1854 1 SIDE EFFECTS:
863 1855 1 The attributes in the file header of the new file are modified
864 1856 1 according to the attribute of the old version or parent directory.
865 1857 1
866 1858 1 --
867 1859 1
868 1860 2 BEGIN
869 1861 2
870 1862 2 MAP
871 1863 2 FIB : REF BBLOCK; ! Address of the FIB
872 1864 2
873 1865 2 LOCAL
874 1866 2 STATUS, ! Routine exit status
875 1867 2 WINDOW : REF BBLOCK, ! Address of created window
876 1868 2 FILE_FCB : REF BBLOCK, ! FCB for newly created file
877 1869 2 ?CB : REF BBLOCK; ! Address of FCB from window
878 1870 2
879 1871 2 BIND_COMMON;
880 1872 2
881 1873 2 EXTERNAL ROUTINE
882 1874 2 READ_HEADER : L_NORM, ! read file header

```

```

883      1875      2          SAVE_CONTEXT : L_NORM,          : Save reentrant context area
884      1876      2          RESTORE_CONTEXT : L_NORM,         : Restore reentrant context area
885      1877      2          OPEN_FILE : L_NORM,             : Open a file
886      1878      2          CLOSE_FILE : L_NORM,            : Close a file
887      1879      2          CHECK_PROTECT : L_NORM;         : Perform a protection check
888      1880
889      1881      2          ENABLE_PROPAGATE_HANDLER;
890      1882
891      1883
892      1884      2          : What we do depends on whether there is an old version present.
893      1885      2          : If it exists, we copy attributes from it. If not, we copy attributes
894      1886      2          : from the directory. If the old version is the same as the file being
895      1887      2          : entered, we do nothing, because the net effect would be a NOP anyway,
896      1888      2          : and we can't open the same file in both primary and secondary context.
897      1889
898      1890
899      1891      2          IF CHSEQL (FID$C_LENGTH, OLD_VERSION_FID,
900      1892      2          FID$C_LENGTH, PRIMARY_FCB[FID$W_FID])
901      1893      2          THEN RETURN 1;
902      1894
903      1895      2          IF .OLD_VERSION_FID[FID$W_NUM] NEQ 0
904      1896      2          OR .OLD_VERSION_FID[FID$B_NMX] NEQ 0
905      1897      2          THEN
906      1898      2              BEGIN
907      1899      2              LOCAL SAVCURRINDX;
908      1900      2              SAVE_STATUS = .USER_STATUS;
909      1901      2              FILE_FCB = .PRIMARY_FCB;          : Save created file FCB address
910      1902      2              SAVCURRINDX = .CURR_LCKINDX;
911      1903      2              SAVE_CONTEXT ();
912      1904      2              WINDOW = OPEN_FILE (OLD_VERSION_FID, 2);
913      1905      2              IF .WINDOW NEQ 0
914      1906      2              THEN
915      1907      2                  BEGIN
916      1908      2                  FCB = .WINDOW[WCBS$L_FCB];
917      1909      2                  IF CHECK_PROTECT (RDATT_ACCESS, 0, .PRIMARY_FCB,
918      1910      2                  MAXU 7.IO_PACKET[IRPSV_MODE], .FIB[FIB$B_AGENT_MODE]))
919      1911      2                  THEN
920      1912      2                      BEGIN
921      1913      2
922      1914      2          : Restore the current lock index we had from primary context.
923      1915      2          : COPY_INFO may need to read the primary file's headers.
924      1916      2          : Also set the HDRNOTCHG flag to avoid bungling quotas when
925      1917      2          : file owner is different from process uic.
926      1918      2
927      1919      2              CURR_LCKINDX = .SAVCURRINDX;
928      1920      2              CLEARUP_FLAGS [CLF HDRNOTCHG] = 1;
929      1921      2              STATUS = KERNEL_CACL (COPY_INFO, .FCB, .FILE_FCB, .FIB, 0);
930      1922      2              CLOSE_FILE (.WINDOW);
931      1923      2              RESTORE_CONTEXT ();
932      1924      2              READ_HEADER (CURRENT_FIB[FIB$W_FID], .PRIMARY_FCB);
933      1925      2              RETURN .STATUS;
934      1926      2              END;
935      1927      2          END;
936      1928      2          RESTORE_CONTEXT ();
937      1929      2          USER_STATUS = .SAVE_STATUS;
938      1930      2          READ_HEADER (CURRENT_FIB[FIB$W_FID], .PRIMARY_FCB);
939      1931      2

```

```

: 937      1932 2      END;
: 938      1933 2
: 939      1934 2      ! If we make it this far, it means that: 1) there was no previous version of
: 940      1935 2      ! the file; 2) the previous version of the file is not accessible; or 3) the
: 941      1936 2      ! current process does not have access to the previous version. In any of
: 942      1937 2      ! these cases, propagate as a newly created file.
: 943      1938 2
: 944      1939 2      STATUS = KERNEL_CALL (COPY_INFO, .DIR_FCB, .PRIMARY_FCB, .FIB, 1);
: 945      1940 2
: 946      1941 2      RETURN .STATUS;
: 947      1942 2
: 948      1943 1      END;

```

! End of routine PROPAGATE_ATTR

.EXTRN OPEN_FILE, CLOSE_FILE

007C 00000 PROPAGATE ATTR:

					WORD	Save R2,R3,R4,R5,R6	1818				
	55	08	AA	9E	00002	MOVAB	8(BASE), R5	1869			
	54	014C	CA	9E	00006	MOVAB	332(BASE), R4				
	6D	00C3	CF	DE	0000B	MOVAL	7\$, (FP)				
	50		65	D0	00010	MOVL	(R5), R0	1892			
24	A0		64	06	29	CMPC3	#6, (R4), 36(R0)				
				04	12	BNEQ	1\$				
	50			01	D0	MOVL	#1, R0	1893			
				04	0001D	RET					
				64	B5	0001E	1\$:				
				08	12	00020	TSTW	(R4)	1895		
				05	A4	95	00022	BNEQ	2\$		
				03	12	00025	TSTB	5(R4)	1896		
				0091	31	00027	BNEQ	2\$			
				80	AA	D0	0002A	2\$:			
	CO	AA		65	D0	0002F	BRW	5\$			
		56		AA	D0	00032	MOVL	-128(BASE), -64(BASE)	1900		
		53		14	AA	D0	00036	MOVL	(R5), FILE_FCB	1901	
	0000G	CF		00	FB	00036	MOVL	20(BASE), SAVCURRINDX	1902		
				02	DD	C0038	CALLS	#0, SAVE_CONTEXT	1903		
				54	DD	0003D	PUSHL	#2	1904		
	0000G	CF		02	FB	0003F	PUSHL	R4			
		52		50	D0	00044	CALLS	#2, OPEN_FILE			
				5C	13	00047	MOVL	R0, WINDOW			
				54	18	A2	D0	00049	BEQL	4\$	1905
				51	90	AA	D0	0004D	MOVL	24(WINDOW), FCB	1908
				50	04	AC	D0	00051	MOVL	-112(BASE), R1	1910
				02	00	EF	00055	MOVL	FIB, R0		
7E	OB	A1		6E	2E	A0	91	0005B	EXTZV	#0, #2, 11(R1), -(SP)	
				6E	2E	A0	9A	00061	MPB	46(R0), (SP)	
				65	DD	00065	3\$:	BLEQU	3\$		
				7E	04	7D	00067	MOVZBL	46(R0), (SP)		
	0000G	CF		04	FB	0006A	PUSHL	(R5)		1909	
		33		50	E9	0006F	MOVQ	#4, -(SP)			
		14	AA	53	D0	00072	CALLS	#4, CHECK_PROTECT			
		03	AA	08	88	0C076	BLBC	R0, 4\$			
				7E	D4	0007A	MOVL	SAVCURRINDX, 20(BASE)			
				04	AC	DD	0007C	BISB2	#8, 3(BASE)	1920	
				8F	BB	0007F	CLRL	-(SP)	1921		
							PUSHL	FIB	1922		
							PUSHR	#*M<R4,R6>			

	0000V	CF		04	FB	00083		CALLS	#4, COPY_INFO		
		53		50	DO	00088		MOVL	R0, STATUS		
				52	DD	0008B		PUSHL	WINDOW		1923
	0000G	CF		01	FB	0008D		CALLS	#1, CLOSE_FILE		
	0000G	CF		00	FB	00092		CALLS	#0, RESTORE_CONTEXT		1924
				65	DD	00097		PUSHL	(R5)		1925
7E	10	AA		04	C1	00099		ADDL3	#4, 16(BASE), -(SP)		
	0000G	CF		02	FB	0009E		CALLS	#2, READ_HEADER		
				29	11	000A3		BRB	68		1926
	0000G	CF		00	FB	000A5	4\$:	CALLS	#0, RESTORE_CONTEXT		1929
	80	AA	C0	AA	DO	000AA		MOVL	-64(BASE), -128(BASE)		1930
				65	DD	000AF		PUSHL	(R5)		1931
7E	10	AA		04	C1	000B1		ADDL3	#4, 16(BASE), -(SP)		
	0000G	CF		02	FB	000B6		CALLS	#2, READ_HEADER		
				01	DD	000BB	5\$:	PUSHL	#1		1939
				04	AC	DD	000BD	PUSHL	FIB		
				65	DD	000C0		PUSHL	(R5)		
			00D0	CA	DD	000C2		PUSHL	208(BASE)		
	0000V	CF		04	FB	000C6		CALLS	#4, COPY_INFO		
		53		50	DO	000CB		MOVL	R0, STATUS		
		50		53	DO	000CE	6\$:	MOVL	STATUS, R0		1941
					04	000D1		RET			1943
					0000	000D2	7\$:	.WORD	Save nothing		1869
				7E	D4	000D4		CLRL	-(SP)		
				5E	DD	000D6		PUSHL	SP		
		7E	04	AC	7D	000D8		MOVQ	4(AP), -(SP)		
	0000V	CF		03	FB	000DC		CALLS	#3, PROPAGATE_HANDLER		
				04	000E1			RET			

: Routine Size: 226 bytes, Routine Base: \$CODE\$ + 058F

```

: 950      1944 1 ROUTINE PROPAGATE_HANDLER (SIGNAL, MECHANISM) =
: 951      1945 1
: 952      1946 1 :++
: 953      1947 1
: 954      1948 1 : FUNCTIONAL DESCRIPTION:
: 955      1949 1
: 956      1950 1 :     This routine is the condition handler for the file attribute
: 957      1951 1 :     propagation.  It unwinds and returns a value of zero to
: 958      1952 1 :     indicate a failure.
: 959      1953 1
: 960      1954 1 : CALLING SEQUENCE:
: 961      1955 1 :     PROPAGATE_HANDLER (ARG1, ARG2)
: 962      1956 1
: 963      1957 1 : INPUT PARAMETERS:
: 964      1958 1 :     ARG1: address of the signal array
: 965      1959 1 :     ARG2: address of the mechanism array
: 966      1960 1
: 967      1961 1 : IMPLICIT INPUTS:
: 968      1962 1 :     none
: 969      1963 1
: 970      1964 1 : OUTPUT PARAMETERS:
: 971      1965 1 :     none
: 972      1966 1
: 973      1967 1 : IMPLICIT OUTPUTS:
: 974      1968 1 :     Value of the routine that caused the exception is returned as zero.
: 975      1969 1
: 976      1970 1 : ROUTINE VALUE:
: 977      1971 1 :     $$$_RESIGNAL or none
: 978      1972 1
: 979      1973 1 : SIDE EFFECTS:
: 980      1974 1 :     none
: 981      1975 1
: 982      1976 1 : --
: 983      1977 1
: 984      1978 2 BEGIN
: 985      1979 2
: 986      1980 2 MAP
: 987      1981 2     SIGNAL          : REF BBLOCK,          : Signal argument array
: 988      1982 2     MECHANISM       : REF BBLOCK;          : Mechanism argument array
: 989      1983 2
: 990      1984 2 : If the condition is change mode to user (ERR_EXIT) set the saved value
: 991      1985 2 : of R0 to zero (indicating a failure) and unwind to the PROPAGATE_ATTR
: 992      1986 2 : routine.
: 993      1987 2
: 994      1988 2 IF .SIGNAL[CHFSL_SIG_NAME] EQL $$$_CMODUSER
: 995      1989 2 THEN
: 996      1990 3     BEGIN
: 997      1991 3     MECHANISM[CHFSL_MCH_SAVRO] = 0;          : Note failure
: 998      1992 3     SUNWIND (DEPADR = MECHANISM[CHFSL_MCH_DEPTH],
: 999      1993 3     NEWPC = 0);
: 1000     1994 2     END;
: 1001     1995 2
: 1002     1996 2 RETURN $$$_RESIGNAL;          : Ignored when unwinding
: 1003     1997 2
: 1004     1998 1 END;          : End of routine PROPAGATE_HANDLER

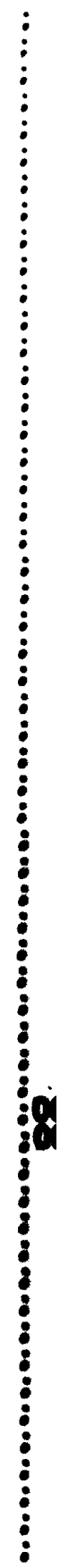
```

.EXTRN SYSSUNWIND

0000 0000 PROPAGATE HANDLER:

	50	04	AC	D0	00002	.WORD	Save nothing	:	1944
00000424	8F	04	A0	D1	00006	MOVL	SIGNAL, R0	:	1988
	50	08	AC	D0	00010	BNEQ	1\$:	1991
		CC	A0	D4	00014	MOVL	MECHANISM, R0	:	1993
			7E	D4	00017	CLRL	12(R0)	:	1996
7E	08	AC	08	C1	00019	CLRL	-(SP)	:	1998
00000000G	00		02	FB	0001E	ADDL3	#8, MECHANISM, -(SP)	:	
	50	0918	8F	3C	00025	CALLS	#2, SYSSUNWIND	:	
			04	0002A	1\$:	MOVZWL	#2328, R0	:	
						RET		:	

; Routine Size: 43 bytes, Routine Base: \$CODE\$ + 0671



```

: 1006 1999 1 ROUTINE COPY_INFO (OLD_FILE_FCB, NEW_FILE_FCB, FIB, NEW_FILE) : L_NORM =
: 1007 2000 1
: 1008 2001 1 :++
: 1009 2002 1
: 1010 2003 1 FUNCTIONAL DESCRIPTION:
: 1011 2004 1
: 1012 2005 1 This routine actually copies the propagated information. This
: 1013 2006 1 routine must be called in kernel mode. The propagation takes
: 1014 2007 1 place according to the following rules:
: 1015 2008 1
: 1016 2009 1 UIC - For a newly created file, the file takes the UIC of the
: 1017 2010 1 creator unless the creator has resource rights to the
: 1018 2011 1 owner of the directory. In which case, the UIC of the
: 1019 2012 1 directory owner is used. For a new version of an
: 1020 2013 1 existing file, the UIC of the creator is used if the
: 1021 2014 1 creator does not have resource rights to either the
: 1022 2015 1 old version owner or the directory owner. If the
: 1023 2016 1 creator has resource rights to the old version owner,
: 1024 2017 1 that UIC is used. If not, and the creator has resource
: 1025 2018 1 rights to the directory owner, the directory owner
: 1026 2019 1 UIC is used.
: 1027 2020 1
: 1028 2021 1 Protection - For a newly created file, the protection is taken from
: 1029 2022 1 the directory default protection ACE, if it exists. If
: 1030 2023 1 it does not exist, the process default protection is used.
: 1031 2024 1 For a new version of an existing file, the protection is
: 1032 2025 1 taken from the old version of the file.
: 1033 2026 1
: 1034 2027 1 ACL - For a newly created file, the ACL is taken from the
: 1035 2028 1 directory default ACL. If no directory default ACL
: 1036 2029 1 exists, no ACL is propagated. For a new version of
: 1037 2030 1 an existing file, the ACL is taken from the old
: 1038 2031 1 version of the file.
: 1039 2032 1
: 1040 2033 1 CALLING SEQUENCE:
: 1041 2034 1 COPY_INFO (ARG1, ARG2, ARG3, ARG4)
: 1042 2035 1
: 1043 2036 1 INPUT PARAMETERS:
: 1044 2037 1 ARG1: address of the old file's FCB (if one)
: 1045 2038 1 ARG2: address of the new file's FCB
: 1046 2039 1 ARG3: address of the FIB
: 1047 2040 1 ARG4: 1 if defaults for a new file
: 1048 2041 1 0 if defaults for a new version of an existing file
: 1049 2042 1
: 1050 2043 1 IMPLICIT INPUTS:
: 1051 2044 1 DIR_FCB: address of parent directory FCB
: 1052 2045 1
: 1053 2046 1 OUTPUT PARAMETERS:
: 1054 2047 1 none
: 1055 2048 1
: 1056 2049 1 IMPLICIT OUTPUTS:
: 1057 2050 1 none
: 1058 2051 1
: 1059 2052 1 ROUTINE VALUE.
: 1060 2053 1 1
: 1061 2054 1
: 1062 2055 1 SIDE EFFECTS:

```



```

1063 2056 1  | The ACL building routine is called to update the new file's file
1064 2057 1  | headers with the copied ACL.
1065 2058 1  |
1066 2059 1  | --
1067 2060 1  |
1068 2061 2  BEGIN
1069 2062 2  |
1070 2063 2  MAP
1071 2064 2  | OLD_FILE_FCB : REF BBLOCK,      ! Address of old file's FCB
1072 2065 2  | NEW_FILE_FCB : REF BBLOCK,      ! Address of new file's FCB
1073 2066 2  | FIB          : REF BBLOCK;      ! Address of the FIB
1074 2067 2  |
1075 2068 2  LINKAGE
1076 2069 2  | L_SEARCH_RIGHT = JSB (REGISTER = 2, REGISTER = 4;
1077 2070 2  |                  REGISTER = 1, REGISTER = 5);
1078 2071 2  |
1079 2072 2  | L_FINDACL      = JSB (REGISTER = 3, REGISTER = 5;
1080 2073 2  |                  REGISTER = 6, REGISTER = 1;
1081 2074 2  |                  REGISTER = 1);
1082 2075 2  |
1083 2076 2  LOCAL
1084 2077 2  | PCB          : REF BBLOCK,      ! PCB address of I/O packet owner
1085 2078 2  | ARB          : REF BBLOCK,      ! Access rights block address
1086 2079 2  | IDENTIFIER,  : REF BBLOCK,      ! Identifier being sought
1087 2080 2  | RIGHTS_DESC, : REF BBLOCK,      ! Rights list descr addr
1088 2081 2  | ID_FOUND     : REF BBLOCK,      ! Addr of ID found
1089 2082 2  | RIGHTS_SEG   : REF BBLOCK,      ! Addr of rights segment
1090 2083 2  | ACE_ADDRESS  : REF BBLOCK,      ! Pointer to default protection ACE
1091 2084 2  | OLD_ACL_SEGMENT : REF BBLOCK,   ! Address of old ACL segment
1092 2085 2  | NEW_ACL_SEGMENT : REF BBLOCK;   ! Address of new ACL segment
1093 2086 2  |
1094 2087 2  EXTERNAL
1095 2088 2  | SCH$GL_PCBVEC : REF VECTOR ADDRESSING_MODE (ABSOLUTE); ! PCB vector
1096 2089 2  |
1097 2090 2  BIND_COMMON;
1098 2091 2  |
1099 2092 2  EXTERNAL ROUTINE
1100 2093 2  | EX$SEARCH_RIGHT : L_SEARCH_RIGHT ADDRESSING_MODE (GENERAL),
1101 2094 2  |                  ! Search for specified ID
1102 2095 2  | EX$FINDACL      : L_FINDACL ADDRESSING_MODE (GENERAL), ! Locate an ACE
1103 2096 2  | ACL_INIT_QUEUE  : ADDRESSING_MODE (GENERAL), ! Initialize ACL queue
1104 2097 2  | ACL_COPYACL     : L_NORM, ! Routine to propagate desired ACEs
1105 2098 2  | CHANGE_OWNER    : L_NORM; ! Change file owner UIC
1106 2099 2  |
1107 2100 2  ENABLE PROPAGATE_HANDLER;
1108 2101 2  |
1109 2102 2  ! Initialize some necessary pointers.
1110 2103 2  |
1111 2104 2  PCB = .SCH$GL_PCBVEC[(IO_PACKET[IRPSL_PID])<0,16>];
1112 2105 2  ARB = .IO_PACKET[IRPSL_ARB];
1113 2106 2  RIGHTS_DESC = ARB[ARB$C_RIGHTSLIST];
1114 2107 2  |
1115 2108 2  ! If is a new file, propagate the information from the parent directory
1116 2109 2  ! or the creator of the file as necessary.
1117 2110 2  |
1118 2111 2  IF .NEW_FILE
1119 2112 2  THEN

```

```

: 1120      2113      3      BEGIN
: 1121      2114      3      IF .DIR_FCB NEQ 0
: 1122      2115      3      THEN
: 1123      2116      4      BEGIN
: 1124      2117      4      CHANGE_OWNER (.DIR_FCB[FCBSL_FILEOWNER], .NEW_FILE_FCB, 0);
: 1125      2118      4      NEW_FILE_FCB[FCBSW_FILEPROT] = .PCB[PCBSL_DEFPROT];
: 1126      2119      4      IF .BBLOCK[DIR_FCB[FCBSR_ORB], ORBSV_ACL_QUEUE]
: 1127      2120      4      THEN
: 1128      2121      5      BEGIN
: 1129      2122      5      OLD_ACL_SEGMENT = .DIR_FCB[FCBSL_ACLFL];
: 1130      2123      5      UNTIL .OLD_ACL_SEGMENT EQLA DIR_FCB[FCBSL_ACLFL]
: 1131      2124      5      DO
: 1132      2125      6      BEGIN
: 1133      2126      6      ACE_ADDRESS = 0;
: 1134      2127      6      IF EXES$FINDACL (ACESC DIRDEF,
: 1135      2128      6      .OLD_ACL_SEGMENT[ACL$W_SIZE] - ACL$C_LENGTH,
: 1136      2129      6      OLD_ACL_SEGMENT[ACL$S_LIST], .ACE_ADDRESS;
: 1137      2130      6      ACE_ADDRESS)
: 1138      2131      6      THEN
: 1139      2132      7      BEGIN
: 1140      2133      7      (NEW_FILE_FCB[FCBSW_FILEPROT])<0,4> = .ACE_ADDRESS[ACESL_SYS_PROT];
: 1141      2134      7      (NEW_FILE_FCB[FCBSW_FILEPROT])<4,4> = .ACE_ADDRESS[ACESL_OWN_PROT];
: 1142      2135      7      (NEW_FILE_FCB[FCBSW_FILEPROT])<8,4> = .ACE_ADDRESS[ACESL_GRP_PROT];
: 1143      2136      7      (NEW_FILE_FCB[FCBSW_FILEPROT])<12,4> = .ACE_ADDRESS[ACESL_WOR_PROT];
: 1144      2137      7      EXIT[COOP];
: 1145      2138      6      END;
: 1146      2139      6      OLD_ACL_SEGMENT = .OLD_ACL_SEGMENT[ACL$S_FLINK];
: 1147      2140      5      END;
: 1148      2141      5      ACL_INIT_QUEUE (NEW_FILE_FCB[FCBSR_ORB]);
: 1149      2142      6      RETURN ACL_COPYACL T.DIR_FCB, .NEW_FILE_FCB, (IF .FIB[FIB$V_DIRACL]
: 1150      2143      5      THEN 2 ELSE 1);
: 1151      2144      4      END;
: 1152      2145      4      RETURN 1;
: 1153      2146      3      END;
: 1154      2147      2      END;
: 1155      2148      2
: 1156      2149      2      ! If it is a new version of an existing file, propagate the information
: 1157      2150      2      ! from the old version of the file, the parent directory, or the creator
: 1158      2151      2      ! of the file.
: 1159      2152      2
: 1160      2153      2      ! First, set the owner of the new file.
: 1161      2154      2
: 1162      2155      2      IF NOT CHANGE_OWNER (.OLD_FILE_FCB[FCBSL_FILEOWNER], .NEW_FILE_FCB, 0)
: 1163      2156      2      AND .DIR_FCB NEQ 0
: 1164      2157      2      THEN CHANGE_OWNER (.DIR_FCB[FCBSL_FILEOWNER], .NEW_FILE_FCB, 0);
: 1165      2158      2
: 1166      2159      2      ! Next, propagate the protection from the old file.
: 1167      2160      2
: 1168      2161      2      NEW_FILE_FCB[FCBSW_FILEPROT] = .OLD_FILE_FCB[FCBSW_FILEPROT];
: 1169      2162      2
: 1170      2163      2      ! Last, but not least, copy the ACL (excluding ACEs marked as NOPROPAGATE).
: 1171      2164      2
: 1172      2165      2      IF .BBLOCK[OLD_FILE_FCB[FCBSR_ORB], ORBSV_ACL_QUEUE]
: 1173      2166      2      THEN
: 1174      2167      3      BEGIN
: 1175      2168      3      ACL_INIT_QUEUE (NEW_FILE_FCB[FCBSR_ORB]);
: 1176      2169      3      RETURN ACL_COPYACL T.OLD_FILE_FCB, .NEW_FILE_FCB, 2)

```

```

: 1177      2170 3   END
: 1178      2171 2   ELSE RETURN 1;
: 1179      2172 2
: 1180      2173 1   END;

```

! End of routine COPY_INFO

```

.EXTRN EXESSEARCH RIGHT
.EXTRN EXESFINDACL, ACL_COPYACL
.EXTRN CHANGE_OWNER

```

OBFC 00000 COPY_INFO:

						.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R11	1999
						MOVAB	CHANGE_OWNER, R8	
						MOVAB	ACL_INIT_QUEUE, R7	
						MOVAB	2087(BASET), R4	2088
						MOVAL	13\$, (FP)	
						MOVL	#NSCH\$GL PCBVEC, R1	2104
						MOVL	-112(BASE), RO	
						ADDL2	#12, RO	
						MOVZWL	(RO), RO	
						MOVL	(R1)(RO), PCB	
						MOVL	-112(BASE), RO	2105
						MOVL	88(RO), ARB	
						ADDL2	#32, RIGHTS_DESC	2106
						BLBS	NEW_FILE, 2\$	2111
						BRW	9\$	
						MOVL	(R4), RO	2114
						BEQL	1\$	
						CLRL	-(SP)	2117
						PUSHL	NEW_FILE_FCB	
						PUSHL	88(RO)	
						CALLS	#3, CHANGE_OWNER	
						MOVL	NEW_FILE_FCB, RO	2118
						MOVW	276(PCB), 112(RO)	
						MOVL	(R4), RO	2119
						BBS	#1, 99(RO), 3\$	
						BRW	12\$	
						MOVL	128(RO), OLD_ACL_SEGMENT	2122
						ADDL3	#128, (R4), RO	2123
						CMPL	OLD_ACL_SEGMENT, RO	
						BEQL	6\$	
						CLRL	ACE_ADDRESS	2126
						MOVAB	12(OLD_ACL_SEGMENT), R6	2129
						MOVZWL	8(OLD_ACL_SEGMENT), R5	2129
						SUBL2	#12, R5	
						MOVL	#9, R3	2129
						JSB	EXESFINDACL	
						BLBC	RO, 5\$	
						MOVL	NEW_FILE_FCB, RO	2133
						INSV	8(ACE_ADDRESS), #0, #4, 112(RO)	
						MOVL	NEW_FILE_FCB, RO	2134
						INSV	12(ACE_ADDRESS), #4, #4, 112(RO)	
						MOVL	NEW_FILE_FCB, RO	2135
						INSV	16(ACE_ADDRESS), #0, #4, 113(RO)	
						MOVL	NEW_FILE_FCB, RO	2136
						INSV	20(ACE_ADDRESS), #4, #4, 113(RO)	

				05	11	000BB		BRB	6\$		2132
				62	DO	000BD	5\$:	MOVL	(CLD_ACL_SEGMENT), OLD_ACL_SEGMENT		2139
				A7	11	000C0		BRB	4\$		2123
7E	08	AC	00000058	8F	C1	000C2	6\$:	ADDL3	#88, NEW_FILE_FCB, -(SP)		2141
		67		01	FB	000CB		CALLS	#1, ACL_INIT_QUEUE		
		50		AC	DO	000CE		MOVL	FIB, R0		2142
04	38	A0	0C	02	E1	000D2		BBC	#2, 56(R0), 7\$		
				02	DD	000D7		PUSHL	#2		
				02	11	000D9		BRB	8\$		
				01	DD	000DB	7\$:	PUSHL	#1		
			08	AC	DD	000DD	8\$:	PUSHL	NEW_FILE_FCB		
				64	DD	000E0		PUSHL	(R4)		
				48	11	000E2		BRB	11\$		
				7E	D4	000E4	9\$:	CLRL	-(SP)		2155
			08	AC	DD	000E6		PUSHL	NEW_FILE_FCB		
		50		04	AC	DO	000E9	MOVL	OLD_FILE_FCB, R0		
			58	A0	DD	000ED		PUSHL	88(R0)		
		68		03	FB	000F0		CALLS	#3, CHANGE_OWNER		
		12		50	E8	000F3		BLBS	R0, 10\$		
				64	D5	000F6		TSTL	(R4)		2156
				0E	13	000F8		BEQL	i0\$		
				7E	D4	000FA		CLRL	-(SP)		2157
			08	AC	DD	000FC		PUSHL	NEW_FILE_FCB		
		50		64	DO	000FF		MOVL	(R4), R0		
			58	A0	DD	00102		PUSHL	88(R0)		
		68		03	FB	00105		CALLS	#3, CHANGE_OWNER		
		50		04	AC	7D	00108	10\$:	MOVQ	OLD_FILE_FCB, R0	2161
	70	A1		70	A0	B0	0010C		MOVW	112(R0), 112(R1)	
		50		04	AC	DO	00111		MOVL	OLD_FILE_FCB, R0	2165
18	63	A0		01	E1	00115		BBC	#1, 99(R0), 12\$		
7E	08	AC	00000058	8F	C1	0011A		ADDL3	#88, NEW_FILE_FCB, -(SP)		2168
		67		01	FB	00123		CALLS	#1, ACL_INIT_QUEUE		
				02	DD	00126		PUSHL	#2		2169
		7E		04	AC	7D	00128		MOVQ	OLD_FILE_FCB, -(SP)	
		0000G		03	FB	0012C	11\$:	CALLS	#3, ACL_COPYACL		
					04	00131		RET			2171
		50		01	DO	00132	12\$:	MOVL	#1, R0		
					04	00135		RET			2173
					0000	00136	13\$:	.WORD	Save nothing		2088
				7E	D4	00138		CLRL	-(SP)		
				5E	DD	0013A		PUSHL	SP		
		7E		04	AC	7D	0013C		MOVQ	4(AP), -(SP)	
	FE90	CF		03	FB	00140		CALLS	#3, PROPAGATE_HANDLER		
				04	00145			RET			

: Routine Size: 326 bytes, Routine Base: \$CODE\$ + 069C

: 1181 2174 1
: 1182 2175 1 END
: 1183 2176 0 ELUDOM

PSECT SUMMARY

```
:
:      Name                Bytes                Attributes
: $CODE$                2018 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
```

Library Statistics

```
:
:      File                ----- Symbols ----- Pages Processing
:      _$255$DUA18:[SYSLIB]LIB.L32;1  Total   Loaded   Percent   Mapped   Time
:                                     18619    140      0        1000    00:01.9
```

COMMAND QUALIFIERS

```
:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:CREATE/OBJ=OBJ$:CREATE MSRC$:CREATE/UPDATE=(BUG$:CREATE)
```

```
: Size:                2018 code + 0 data bytes
: Run Time:            01:11.2
: Elapsed Time:       01:44.6
: Lines/CPU Min:      1832
: Lexemes/CPU-Min:    36427
: Memory Used:        549 pages
: Compilation Complete
```

