


```

:001 1 0001 0 MODULE QUOTAUTIL (
:002 2 0002 0     LANGUAGE (BLISS32),
:003 3 0003 0     IDENT = 'V04-003',
:004 4 0004 0     ) =
:005 5 0005 1 BEGIN
:006 6 0006 1
:007 7 0007 1
:008 8 0008 1 *****
:009 9 0009 1 *
:010 10 0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
:011 11 0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
:012 12 0012 1 *  ALL RIGHTS RESERVED.
:013 13 0013 1 *
:014 14 0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
:015 15 0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
:016 16 0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
:017 17 0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
:018 18 0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
:019 19 0019 1 *  TRANSFERRED.
:020 20 0020 1 *
:021 21 0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
:022 22 0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
:023 23 0023 1 *  CORPORATION.
:024 24 0024 1 *
:025 25 0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
:026 26 0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
:027 27 0027 1 *
:028 28 0028 1 *
:029 29 0029 1 *****
:030 30 0030 1
:031 31 0031 1 ++
:032 32 0032 1
:033 33 0033 1 FACILITY: F11ACP Structure Level 2
:034 34 0034 1
:035 35 0035 1 ABSTRACT:
:036 36 0036 1
:037 37 0037 1     This module contains routines that implement the ACP control
:038 38 0038 1     functions that operate on the quota file.
:039 39 0039 1
:040 40 0040 1 ENVIRONMENT:
:041 41 0041 1
:042 42 0042 1     STARLET operating system, including privileged system services
:043 43 0043 1     and internal exec routines.
:044 44 0044 1
:045 45 0045 1 --
:046 46 0046 1
:047 47 0047 1
:048 48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 31-May-1979 15:18
:049 49 0049 1
:050 50 0050 1 MODIFIED BY:
:051 51 0051 1
:001 001 CDS0009 0052 1     V04-003 CDS0009     Christian D. Saether     15-Nov-1984
:002 002 CDS0009 0053 1     Expand test for clusterness to look at clu$gl_club.
:003 003 CDS0009 0054 1
:004 004 ACG0468 0055 1     V04-002 ACG0468     Andrew C. Goldstein,     18-Sep-1984 18:35
:005 005 ACG0468 0056 1     Count a writer to the quota file when enabling quotas
:006 006 ACG0468 0057 1
    
```

```

52 0058 1 V04-001 ACG0466 Andrew C. Goldstein, 12-Sep-1984 14:38
53 0059 1 Flush quota file blocks from cache when disabling quotas
54 0060 1
55 0061 1 V03-012 CDS0008 Christian D. Saether 29-Aug-1984
56 0062 1 Deal with potential multi-header quota file caused
57 0063 1 by ACL's.
58 0064 1
59 0065 1 V03-011 CDS0007 Christian D. Saether 23-Aug-1984
60 0066 1 Mark quota fcb stale clusterwide when it is extended.
61 0067 1
62 0068 1 V03-010 ACG0438 Andrew C. Goldstein, 18-Jul-1984 20:32
63 0069 1 Implement quota cache lock; dequeue when cache is released.
64 0070 1 Use central dequeue routine.
65 0071 1
66 0072 1 V03-009 CDS0006 Christian D. Saether 9-May-1984
67 0073 1 Add serialization call to flush_quo_cache routine.
68 0074 1
69 0075 1 V03-008 CDS0005 Christian D. Saether 19-Apr-1984
70 0076 1 Bump REFCNT in quota file fcb also.
71 0077 1
72 0078 1 V03-007 ACG0412 Andrew C. Goldstein, 22-Mar-1984 18:35
73 0079 1 Implement agent access mode support; add access mode to
74 0080 1 protection check call
75 0081 1
76 0082 1 V03-006 ACG0400 Andrew C. Goldstein, 7-Mar-1984 17:07
77 0083 1 Implement cluster-wide quota cache, remove marking
78 0084 1 of SCB for quotas.
79 0085 1
80 0086 1 V03-005 CDS0004 Christian D. Saether 1-Mar-1984
81 0087 1 Remove reference to FLUSH_FID.
82 0088 1
83 0089 1 V03-004 CDS0003 Christian D. Saether 30-Dec-1983
84 0090 1 Use L_NORM linkage and BIND_COMMON macro.
85 0091 1
86 0092 1 V03-003 CDS0002 Christian D. Saether 6-Dec-1983
87 0093 1 Volume lock check on quota file modification request
88 0094 1 has changed. NOALLOC is no longer set.
89 0095 1
90 0096 1 V03-002 CDS0001 Christian D. Saether 17-Oct-1983
91 0097 1 Add minimal quota checking support for xqp.
92 0098 1
93 0099 1 V03-001 ACG0308 Andrew C. Goldstein, 14-Jan-1983 14:26
94 0100 1 Fix consistency problems in linking FCB's
95 0101 1
96 0102 1 V02-005 ACG0213 Andrew C. Goldstein, 13-Aug-1981 13:42
97 0103 1 Remove write lock from quota file
98 0104 1
99 0105 1 V02-004 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:27
100 0106 1 Previous revision history moved to F11B.REV
101 0107 1
102 0108 1
103 0109 1
104 0110 1 LIBRARY 'SYSSLIBRARY:LIB.L32';
105 0111 1 REQUIRE 'SRCS:FCPDEF.B32';
106 1102 1
107 1103 1
108 1104 1 FORWARD ROUTINE

```


:	109	1105	1	QUOTA_FILE_OP	:	L_NORM NOVALUE,	!	general quota file operations
:	110	1106	1	FLUSH_QUO_CACHE	:	L_NORM NOVALUE,	!	flush dirty entries from quota cache
:	111	1107	1	DEACC_QFILE	:	L_NORM,	!	deaccess the quota file
:	112	1108	1	RET_QENTRY	:	L_NORM,	!	return quota file entry to user
:	113	1109	1	CONN_QFILE	:	L_NORM NOVALUE,	!	connect the quota file
:	114	1110	1	MAKE_QFCB	:	L_NORM;	!	complete quota file access

```

116 1111 1 GLOBAL ROUTINE QUOTA_FILE_OP (ABD, FIB) : L_NORM NOVALUE =
117 1112 1
118 1113 1 |++
119 1114 1
120 1115 1 FUNCTIONAL DESCRIPTION:
121 1116 1
122 1117 1     This routine implements most of the quota file ACP control functions
123 1118 1     (i.e., the ones that are performed on the open quota file).
124 1119 1
125 1120 1 CALLING SEQUENCE:
126 1121 1     QUOTA_FILE_OP (ARG1, ARG2)
127 1122 1
128 1123 1 INPUT PARAMETERS:
129 1124 1     ARG1: address of buffer descriptor packet
130 1125 1     ARG2: address of user FIB
131 1126 1
132 1127 1 IMPLICIT INPUTS:
133 1128 1     CLEANUP_FLAGS: cleanup action and status flags
134 1129 1     CURRENT_VCB: VCB of current volume
135 1130 1     IO_PACKET: I/O packet being processed
136 1131 1     QUOTA_RECORD: record number of found quota file record
137 1132 1     FREE_QUOTA: record number of first free quota file record
138 1133 1
139 1134 1 OUTPUT PARAMETERS:
140 1135 1     NONE
141 1136 1
142 1137 1 IMPLICIT OUTPUTS:
143 1138 1     PRIMARY_FCB: FCB of quota file
144 1139 1
145 1140 1 ROUTINE VALUE:
146 1141 1     NONE
147 1142 1
148 1143 1 SIDE EFFECTS:
149 1144 1     quota file searched, modified, etc.
150 1145 1
151 1146 1 --
152 1147 1
153 1148 2 BEGIN
154 1149 2
155 1150 2 MAP
156 1151 2     ABD           : REF BBLOCKVECTOR [ ,ABD$C_LENGTH],
157 1152 2                   :           ! buffer descriptor vector
158 1153 2     FIB           : REF BBLOCK;       ! user FIB
159 1154 2
160 1155 2 LITERAL
161 1156 2     RECS_PER_BLOCK = 512 / DQF$C_LENGTH,
162 1157 2
163 1158 2     MAX_QFUNC      = MAXU (FIB$C_DSA_QUOTA,
164 1159 2                          FIB$C_EXA_QUOTA,
165 1160 2                          FIB$C_REM_QUOTA,
166 1161 2                          FIB$C_MOD_QUOTA,
167 1162 2                          FIB$C_ADD_QUOTA
168 1163 2                          ),
169 1164 2
170 1165 2     MIN_QFUNC      = MINU (FIB$C_DSA_QUOTA,
171 1166 2                          FIB$C_EXA_QUOTA,
172 1167 2                          FIB$C_REM_QUOTA,

```



```

173      1168      2      FIBSC_MOD_QUOTA,
174      1169      2      FIBSC_ADD_QUOTA
175      1170      2      );
176      1171      2
177      1172      2      LOCAL
178      1173      2      TEMP1,           ! random temp storage
179      1174      2      TEMP2,           ! more of the same
180      1175      2      FCB              : REF BBLOCK,      ! address of quota file FCB
181      1176      2      BUFFER           : REF BBLOCK,      ! disk block buffer
182      1177      2      Q_RECORD         : REF BBLOCK,      ! record found in quota file
183      1178      2      Q_BLOCK          : REF BBLOCK;       ! quota arg block from user
184      1179      2
185      1180      2      BIND_COMMON;
186      1181      2
187      1182      2      EXTERNAL ROUTINE
188      1183      2      MAKE_FCB_STALE  : L_NORM NOVALUE, ! mark fcb stale clusterwide
189      1184      2      SERIAL_FILE    : L_NORM,          ! serialize on given file
190      1185      2      ALLOCATION_LOCK  : L_NORM,          ! serialize on volume allocation
191      1186      2      SWITCH_VOLUME   : L_NORM,          ! switch volume context
192      1187      2      SEARCH_QUOTA    : L_NORM,          ! find entry in quota file
193      1188      2      CHECK_PROTECT   : L_NORM,          ! check file protection
194      1189      2      GET_QUOTA_LOCK  : L_NORM,          ! take lock on quota cache entry
195      1190      2      REL_QUOTA_LOCK  : L_NORM,          ! release lock on quota cache entry
196      1191      2      WRITE_DIRTY    : L_NORM NOVALUE, ! write dirty buffers
197      1192      2      READ_BLOCK      : L_NORM,          ! read a disk block
198      1193      2      EXTEND_CONTIG   : L_NORM,          ! extend a contiguous file
199      1194      2      WRITE_QUOTA     : L_NORM;          ! write quota file record
200      1195      2
201      1196      2
202      1197      2      ! Do the preliminary setup and validation. All operations handled by this
203      1198      2      ! routine operate on RVN 1 of a volume set and require the quota file to
204      1199      2      ! be connected.
205      1200      2
206      1201      2
207      1202      2      SWITCH_VOLUME (1);
208      1203      2      PRIMARY_FCB = FCB = .CURRENT_VCB[VCS$L_QUOTAFCB];
209      1204      2      IF .FCB=EQL 0
210      1205      2      THEN ERR_EXIT (SS$_QFNOTACT);
211      1206      2
212      1207      2      SERIAL_FILE (FCB [FCBSW_FID]);
213      1208      2
214      1209      2      ALLOCATION_LOCK ();
215      1210      2
216      1211      2      ! Do additional validation which is common for several functions. All but
217      1212      2      ! the disable function require a quota file search and require the quota
218      1213      2      ! argument block (P2) to be present.
219      1214      2
220      1215      2
221      1216      2      IF .FIB[FIBSW_CNTRLFUNC] NEQ FIBSC_DSA_QUOTA
222      1217      2      THEN
223      1218      2      BEGIN
224      1219      2      IF .ABD[ABD$C_NAME, ABD$W_COUNT] LSSU DQF$C_LENGTH
225      1220      2      THEN ERR_EXIT (SS$_INSFARG);
226      1221      2      Q_BLOCK = ABD[ABD$C_NAME, ABD$W_TEXT] + .ABD[ABD$C_NAME, ABD$W_TEXT] + 1;
227      1222      2
228      1223      2      Q_RECORD = SEARCH_QUOTA (.Q_BLOCK[DQF$L_UIC], .FIB[FIB$L_CNTRLVAL], .FIB[FIB$L_WCC], 0);
229      1224      2      IF .FIB[FIB$V_ALL_MEM]

```

```

230 1225 OR .FIB[FIBSV_ALL_GRP]
231 1226 THEN FIB[FIBSC_WCC] = .QUOTA_RECORD;
232 1227
233 1228 ! All functions except disable and examine require write access to the
234 1229 quota file; examine requires read access except when examining one's
235 1230 own quota.
236 1231
237 1232
238 1233 IF .FIB[FIBSW_CNTRLFUNC] NEQ FIBSC_EXA_QUOTA
239 1234 THEN
240 1235 CHECK_PROTECT (WRITE_ACCESS, 0, .FCB, 0)
241 1236 ELSE
242 1237 BEGIN
243 1238 IF .FIB[FIBSV_ALL_MEM]
244 1239 OR .FIB[FIBSV_ALL_GRP]
245 1240 OR .Q_BLOCK[DQFSL_UIC] NEQ
246 1241 .BBLOCK [.IO_PACKET[IRPSL_ARB], ARBSL_UIC]
247 1242 THEN CHECK_PROTECT (READ_ACCESS, 0, .FCB, 0);
248 1243 END;
249 1244
250 1245 ! All functions except disable and add require the quota file search to be
251 1246 successful.
252 1247
253 1248
254 1249 IF .FIB[FIBSW_CNTRLFUNC] NEQ FIBSC_ADD_QUOTA
255 1250 THEN
256 1251 IF .Q_RECORD EQ 0
257 1252 THEN ERR_EXIT (SS$_NODISKQUOTA);
258 1253 END;
259 1254
260 1255 ! Dispatch on the function and do it.
261 1256
262 1257
263 1258 CASE .FIB[FIBSW_CNTRLFUNC] FROM MIN_QFUNC TO MAX_QFUNC OF
264 1259 SET
265 1260
266 1261 [FIBSC_DSA_QUOTA]: ! disable disk quotas
267 1262 BEGIN
268 1263 IF NOT .CLEANUP_FLAGS[CLF_SYSPRV]
269 1264 THEN ERR_EXIT (SS$_NOPRIV);
270 1265 FLUSH_QUO_CACHE ();
271 1266 WRITE_DIRTY (-1);
272 1267 KERNEL_CALL (DEACC_QFILE);
273 1268 END;
274 1269
275 1270 [FIBSC_EXA_QUOTA]: ! examine quota file entry
276 1271 BEGIN
277 1272 KERNEL_CALL (RET_QENTRY, .Q_RECORD, .ABD);
278 1273 END;
279 1274
280 1275 [FIBSC_REM_QUOTA]: ! remove quota file entry
281 1276 BEGIN
282 1277 IF .Q_RECORD[DQFSL_USAGE] NEQ 0
283 1278 THEN ERR_STATUS (SS$_OVRDSKQUOTA);
284 1279 KERNEL_CALL (RET_QENTRY, .Q_RECORD, .ABD);
285 1280 GET_QUOTA_LOCK (.QUOTA_INDEX, LCK$K_EXMODE);
286 1281 CH$FILL (0, DQFSC_LENGTH, .Q_RECORD);

```



```

: 344      1339  2
: 345      1340  2
: 346      1341  2
: 347      1342  2
: 348      1343  2
: 349      1344  1

```

```

[INRANGE, OUNRANGE]: 0; ! should not be called with other functions
TES:
! end of routine QUOTA_FILE_OP

```

```

.TITLE QUOTAUTIL
.IDENT \V04-003\

.EXTRN MAKE_FCB_STALE, SERIAL_FILE
.EXTRN ALLOCATION_LOCK
.EXTRN SWITCH_VOLUME, SEARCH_QUOTA
.EXTRN CHECK_PROTECT, GET_QUOTA_LOCK
.EXTRN REL_QUOTA_LOCK, WRITE_DIRTY
.EXTRN READ_BLOCK, EXTEND_CONFIG
.EXTRN WRITE_QUOTA

.PSECT $CODE$,NOWRT,2

.ENTRY QUOTA_FILE_OP, Save R2,R3,R4,R5,R6,R7,R8,R9 : 1111
0000G 59 0000G CF 03FC 00000 MOVAB WRITE_QUOTA, R9 : 1202
          01 DD 00007 PUSHL #1 : 1203
0000G 50 CF 01 FB 00009 CALLS #1, SWITCH_VOLUME : 1204
          98 AA DO 0000E MOVL -104(BASE), R0 : 1205
          58 54 A0 DO 00012 MOVL 84(R0), FCB : 1207
          08 AA 58 DO 00016 MOVL FCB, 8(BASE) : 1209
          03D4 8F BF 0001C BNEQ 1$ : 1216
          04 00020 CHMU #980 : 1219
          24 A8 9F 00021 1$: RET : 1220
0000G CF 01 FB 00024 PUSHAB 36(FCB) : 1221
0000G CF 00 FB 00029 CALLS #1, SERIAL_FILE : 1223
          50 08 AC DO 0002E MOVL FIB, R0 : 1224
          0A 16 A0 B1 00032 CMPW 22(R0), #10 : 1225
          03 12 00036 BNEQ 2$ : 1226
          008F 31 00038 BRW 10$ : 1227
          50 04 AC DO 0003B 2$: MOVL ABD, R0 : 1228
          20 12 A0 B1 0003F CMPW 18(R0), #32 : 1229
          0114 8F BF 00045 BGEQU 3$ : 1230
          04 00049 CHMU #276 : 1231
          51 04 AC DO 0004A 3$: MOVL ABD, R1 : 1232
          50 10 A1 3C 0004E MOVZWL 16(R1), R0 : 1233
          57 11 A140 9E 00052 MOVAB 17(R1)(R0), Q_BLOCK : 1234
          7E D4 00057 CLRL -(SP) : 1235
          50 08 AC DO 00059 MOVL FIB, R0 : 1236
          10 A0 DD 0005D PUSHL 16(R0) : 1237
          18 A0 DD 00060 PUSHL 24(R0) : 1238
          04 A7 DD 00063 PUSHL 4(Q_BLOCK) : 1239
0000G CF 04 FB 00066 CALLS #4, SEARCH_QUOTA : 1240
          56 50 DO 0006B MOVL R0, Q_RECORD : 1241
          50 08 AC DO 0006E MOVL FIB, R0 : 1242
          05 18 A0 EB 00072 BLBS 24(R0), 4$ : 1243
          06 18 A0 01 E1 00076 BBC #1, 24(R0), 5$ : 1244
          10 A0 02B4 CA DO 0007B 4$: MOVL 692(BASE), 16(R0) : 1245

```


		52	08	AC	D0	00081	5\$:	MOVL	FIB, R2		1233
		0C	16	A2	B1	00085		CMPW	22(R2), #12		
				09	13	00089		BEQL	6\$		
				7E	D4	0008B		CLRL	-(SP)		1235
		7E		58	DD	0008D		PUSHL	FCB		
				01	7D	0008F		MOVQ	#1, -(SP)		
				1E	11	00092		BRB	8\$		
		14	18	A2	E8	00094	6\$:	BLBS	24(R2), 7\$		1238
OF		A2		01	E0	00098		BBS	#1, 24(R2), 7\$		1239
		50		90	AA	0009D		MOVL	-112(BASE), R0		1241
		50		58	A0	000A1		MOVL	88(R0), R0		
		38		04	A7	000A5		CMP	4(Q_BLOCK), 56(R0)		
					0B	13		BEQL	9\$		
					7E	D4	7\$:	CLRL	-(SP)		1242
					58	DD		PUSHL	FCB		
					7E	7C		CLRQ	-(SP)		
	0000G	CF			04	FB	8\$:	CALLS	#4, CHECK_PROTECT		
		50		08	AC	D0	9\$:	MOVL	FIB, R0		1249
		0B		16	A0	B1		CMPW	22(R0), #11		
					09	13		BEQL	10\$		
					56	D5		TSTL	Q_RECORD		1251
					05	12		BNEQ	10\$		
			03E4		8F	BF		CHMU	#996		1252
					04	000C9		RET			
		50		08	AC	D0	10\$:	MOVL	FIB, R0		1258
		0A		16	A0	AF		CASEW	22(R0), #10, #4		
005E	04				000B	000D3	11\$:	.WORD	12\$-11\$,-		
	00A5				0025	000DB			23\$-11\$,-		
									22\$-11\$,-		
									16\$-11\$,-		
									14\$-11\$		
					04	000DD		RET			
		03		01	AA	E8	12\$:	BLBS	1(BASE), 13\$		1263
					24	BF		CHMU	#36		1264
					04	000E4		RET			
	0000V	CF			00	FB	13\$:	CALLS	#0, FLUSH_QUO_CACHE		1265
		7E			01	CE		MNEGL	#1, -(SP)		1266
	0000G	CF			01	FB		CALLS	#1, WRITE_DIRTY		
	0000V	CF			00	FB		CALLS	#0, DEACC_QFILE		1267
					04	000F7		RET			1258
				08	A6	D5	14\$:	TSTL	8(Q_RECORD)		1277
					0A	13		BEQL	15\$		
		06		80	AA	E9		BLBC	-128(BASE), 15\$		1278
	80	AA		0669	8F	B0		MOVW	#1641, -128(BASE)		
				04	AC	DD	15\$:	PUSHL	ABD		1270
					56	DD		PUSHL	Q_RECORD		
	0000V	CF			02	FB		CALLS	#2, RET_QENTRY		
					05	DD		PUSHL	#5		1280
			02C0		CA	DD		PUSHL	704(BASE)		
	0000G	CF			02	FB		CALLS	#2, GET_QUOTA_LOCK		
		6E			00	2C		MOVCS	#0, (SP), #0, #32, (Q_RECORD)		1281
					66	00121					
					56	DD		PUSHL	Q_RECORD		1282
		69			01	FB		CALLS	#1, WRITE_QUOTA		
			02C0		CA	DD		PUSHL	704(BASE)		1283
	0000G	CF			01	FB		CALLS	#1, REL_QUOTA_LOCK		
					04	00130		RET			1258

10	18	A0	FF7C	02	E1	00131	16\$:	BBC	#2, 24(R0), 18\$	1288
				CA	D5	00136		TSTL	-132(BASE)	1291
			0800	05	12	0013A		BNEQ	17\$	1292
				8F	BF	0013C		CHMU	#2048	1293
					04	00140		RET		1295
	08	A6	08	A7	D0	00141	17\$:	MOVL	8(Q_BLOCK), 8(Q_RECORD)	1297
		50	08	AC	D0	00146	18\$:	MOVL	FIB, R0	1298
05	18	A0	03	E1	0014A			BBC	#3, 24(R0), 19\$	1299
	0C	A6	0C	A7	D0	0014F		MOVL	12(Q_BLOCK), 12(Q_RECORD)	1300
		50	08	AC	D0	00154	19\$:	MOVL	FIB, R0	1301
05	18	A0	04	E1	00158			BBC	#4, 24(R0), 20\$	1302
	10	A6	10	A7	D0	0015D		MOVL	16(Q_BLOCK), 16(Q_RECORD)	1303
	0C	A6	08	A6	D1	00162	20\$:	CPL	8(Q_RECORD), 12(Q_RECORD)	1304
				0A	1B	00167		BLEQU	21\$	1305
		06	80	AA	E9	00169		BLBC	-128(BASE), 21\$	1306
	80	AA	0669	8F	B0	0016D		MOVW	#1641, -128(BASE)	1307
				56	DD	00173	21\$:	PUSHL	Q_RECORD	1308
		69		01	FB	00175		CALLS	#T, WRITE_QUOTA	1309
			04	AC	DD	00178	22\$:	PUSHL	ABD	1310
				56	DD	0017B		PUSHL	Q_RECORD	1311
	0000V	CF		02	FB	0017D		CALLS	#2, RET_QENTRY	1312
					04	00182		RET		1313
				56	D5	00183	23\$:	TSTL	Q_RECORD	1314
				05	13	00185		BEQL	24\$	1315
			03DC	8F	BF	00187		CHMU	#988	1316
					04	0018B		RET		1317
		50	02B8	CA	D0	0018C	24\$:	MOVL	696(BASE), R0	1318
				49	12	00191		BNEQ	26\$	1319
	000FFFFF	8F	38	A8	D1	00193		CPL	56(FCB), #1048575	1320
				05	1F	0019B		BLSSU	25\$	1321
			0850	8F	BF	0019D		CHMU	#2128	1322
					04	001A1		RET		1323
		50		08	AC	D0	25\$:	MOVL	FIB, R0	1324
		53		16	A0	3C		MOVZWL	22(R0), TEMP1	1325
		52		18	A0	D0		MOVL	24(R0), TEMP2	1326
				01	DD	001AE		PUSHL	#1	1327
			0101	8F	BB	001B0		PUSHR	#*M<R0,R8>	1328
	0000G	CF		03	FB	001B4		CALLS	#3, EXTEND CONTIG	1329
		56		50	D0	001B9		MOVL	R0, Q_RECORD	1330
				58	DD	001BC		PUSHL	FCB	1331
	0000G	CF		01	FB	001BE		CALLS	#1, MAKE_FCB_STALE	1332
		50	08	AC	D0	001C3		MOVL	FIB, R0	1333
	16	A0		53	B0	001C7		MOVW	TEMP1, 22(R0)	1334
		50	08	AC	D0	001CB		MOVL	FIB, R0	1335
	18	A0		52	D0	001CF		MOVL	TEMP2, 24(R0)	1336
		50	08	AC	D0	001D3		MOVL	FIB, R0	1337
			1C	A0	D4	001D7		CLRL	28(R0)	1338
				2B	11	001DA		BRB	27\$	1339
				05	DD	001DC	26\$:	PUSHL	#5	1340
				01	DD	001DE		PUSHL	#1	1341
				50	D7	001E0		DECL	R0	1342
		50		10	C6	001E2		DIVL2	#16, R0	1343
			30	B840	9F	001E5		PUSHAB	248(FCB)[R0]	1344
	0000G	CF		03	FB	001E9		CALLS	#3, READ_BLOCK	1345
		56		50	D0	001EE		MOVL	R0, Q_RECORD	1346
7E	FFFFFFFF	8F	02B8	01	7A	001F1		EMUL	#1, 696(BASE), #-1, -(SP)	1347
50		50		10	7B	001FC		EDIV	#16, (SP)+, R0, R0	1348

QUOTAUTIL
V04-003

C 13
8-Jan-1985 18:18:45
2-Oct-1984 12:43:36

VAX-11 Bliss-32 V4.0-742
[F11X.BUGSRC]QUOTAUTIL.B32;1

Page 11
(2)

20	00	50	20	C4	00201	MULL2	#32, R0	:
		56	50	C0	00204	ADDL2	R0, Q_RECORD	:
		6E	00	2C	00207	MOVCS	#0, (SP), #0, #32, (Q_RECORD)	: 1331
			66		0020C			:
		66	01	88	0020D	BISB2	#1, (Q_RECORD)	: 1332
	04	A6	04	A7	7D 00210	MOVQ	4(Q_BLOCK), 4(Q_RECORD)	: 1333
	0C	A6	0C	A7	7D 00215	MOVQ	12(Q_BLOCK), 12(Q_RECORD)	: 1335
		69	56	DD	0021A	PUSHL	Q_RECORD	: 1337
			01	FB	0021C	CALLS	#T, WRITE_QUOTA	:
				04	0021F	RET		: 1344

; Routine Size: 544 bytes, Routine Base: \$CODE\$ + 0000

```

351 1345 1 GLOBAL ROUTINE FLUSH_QUO_CACHE : L_NORM NOVALUE =
352 1346 1
353 1347 1 !++
354 1348 1
355 1349 1 FUNCTIONAL DESCRIPTION:
356 1350 1
357 1351 1     This routine flushes dirty entries in the quota cache back to the
358 1352 1     quota file.
359 1353 1
360 1354 1
361 1355 1 CALLING SEQUENCE:
362 1356 1     FLUSH_QUO_CACHE ()
363 1357 1
364 1358 1 INPUT PARAMETERS:
365 1359 1     NONE
366 1360 1
367 1361 1 IMPLICIT INPUTS:
368 1362 1     CURRENT_VCB: VCB of volume
369 1363 1     context set to RVN 1
370 1364 1
371 1365 1 OUTPUT PARAMETERS:
372 1366 1     NONE
373 1367 1
374 1368 1 IMPLICIT OUTPUTS:
375 1369 1     NONE
376 1370 1
377 1371 1 ROUTINE VALUE:
378 1372 1     NONE
379 1373 1
380 1374 1 SIDE EFFECTS:
381 1375 1     quota cache flushed, quota file modified
382 1376 1
383 1377 1 !--
384 1378 1
385 1379 2 BEGIN
386 1380 2
387 1381 2 BUILTIN
388 1382 2     FP;
389 1383 2
390 1384 2 LITERAL
391 1385 2     RECS_PER_BLOCK = 512 / DQF$C_LENGTH;
392 1386 2
393 1387 2 LOCAL
394 1388 2     QUOTA_CACHE      : REF BBLOCK,      ! address of quota cache
395 1389 2     QUOTA_LIST       : REF BBLOCKVECTOR [VCAS$C_QUOLENGTH],
396 1390 2                                     ! address of cache entries
397 1391 2     FCB              : REF BBLOCK,      ! address of quota file FCB
398 1392 2     REC_NUM,         :                   ! record number to read
399 1393 2     STATUS,          :                   ! system service status
400 1394 2     Q_RECORD         : REF BBLOCK,      ! address of record read
401 1395 2     LOCK_STATUS      : VECTOR [2];     ! LKSB for lock conversion
402 1396 2
403 1397 2
404 1398 2 BIND_COMMON;
405 1399 2
406 1400 2 EXTERNAL ROUTINE
407 1401 2     ZERO_ON_ERROR,      ! return zero on error signal (handler)

```


QUOTAUTIL
V04-003

G 13
8-Jan-1985 18:18:45
2-Oct-1984 12:43:36

VAX-11 Bliss-32 V4.0-742
[F11X.BUGSRC]QUOTAUTIL.B32;1

Page 15
(3)

FEFF 000B1	BUGW	
0000* 000B3	.WORD	<BUG\$_XQPERR!4>
04 000B5 5\$:	RET	

: 1457
:
: 1460

: Routine Size: 182 bytes, Routine Base: \$CODE\$ + 0220

```

: 468 1461 1 GLOBAL ROUTINE DEACC_QFILE : L_NORM =
: 469 1462 1
: 470 1463 1 !++
: 471 1464 1
: 472 1465 1 FUNCTIONAL DESCRIPTION:
: 473 1466 1
: 474 1467 1 This routine deaccesses the quota file and releases the FCB if it
: 475 1468 1 is idle. This routine must be aclded in kernel mode.
: 476 1469 1
: 477 1470 1 CALLING SEQUENCE:
: 478 1471 1 DEACC_QFILE ()
: 479 1472 1
: 480 1473 1 INPUT PARAMETERS:
: 481 1474 1 NONE
: 482 1475 1
: 483 1476 1 IMPLICIT INPUTS:
: 484 1477 1 CURRENT_VCB: VCB of volume
: 485 1478 1 context set to RVN 1
: 486 1479 1
: 487 1480 1 OUTPUT PARAMETERS:
: 488 1481 1 NONE
: 489 1482 1
: 490 1483 1 IMPLICIT OUTPUTS:
: 491 1484 1 NONE
: 492 1485 1
: 493 1486 1 ROUTINE VALUE:
: 494 1487 1 1
: 495 1488 1
: 496 1489 1 SIDE EFFECTS:
: 497 1490 1 quota file disconnected from VCB, FCB deallocated
: 498 1491 1
: 499 1492 1 !--
: 500 1493 1
: 501 1494 2 BEGIN
: 502 1495 2
: 503 1496 2 LOCAL
: 504 1497 2 ACCTL, ! calculate remaining access control
: 505 1498 2 LCKMODE, ! lock mode to convert access lock to.
: 506 1499 2 FCB : REF BBLOCK, ! FCB of quota file
: 507 1500 2 STATUS, ! system service status
: 508 1501 2 QUOTA_CACHE : REF BBLOCK; ! address of quota cache block
: 509 1502 2
: 510 1503 2 BIND_COMMON;
: 511 1504 2
: 512 1505 2 EXTERNAL ROUTINE
: 513 1506 2 KILL_BUFFERS : L_NORM, ! flush specified buffers from cache
: 514 1507 2 CONV_ACCLOCK : L_NORM, ! convert access lock
: 515 1508 2 LOCK_MODE : L_JSB fARG, ! calculate lock mode from access ctl
: 516 1509 2 DEQ_LOCK : L_NORM, ! dequeue a lock
: 517 1510 2 DEALLOCATE : L_NORM ADDRESSING_MODE (GENERAL); ! deallocate system dynamic memory
: 518 1511 2
: 519 1512 2
: 520 1513 2 ! Flush the quota file data blocks from the block buffer cache.
: 521 1514 2
: 522 1515 2
: 523 1516 2 KILL_BUFFERS (1, -1);
: 524 1517 2

```



```

: 525      1518 2 ! Decrement access and lock counts on the FCB.
: 526      1519 2 !
: 527      1520 2 !
: 528      1521 2 PRIMARY_FCB = FCB = .CURRENT_VCB[VCBSL_QUOTAFCB];
: 529      1522 2 CURRENT_VCB[VCBSL_QUOTAFCB] = 0;
: 530      1523 2 !
: 531      1524 2 ACCTL = 0;
: 532      1525 2 !
: 001 ACG0468 1526 2 FCB[FCBSW_WCNT] = .FCB[FCBSW_WCNT] - 1;
: 533      1527 2 IF .FCB[FCBSW_WCNT] NEQ 0
: 534      1528 2 THEN ACCTL = FIBSM_WRITE;
: 535      1529 2 !
: 536      1530 2 FCB[FCBSW_TCNT] = .FCB[FCBSW_TCNT] - 1;
: 537      1531 2 !
: 538      1532 2 LCKMODE = 0;
: 539      1533 2 !
: 540      1534 2 IF (FCB[FCBSW_ACNT] = .FCB[FCBSW_ACNT] - 1) NEQ 0
: 541      1535 2 THEN
: 542      1536 2     LCKMODE = LOCK_MODE (.ACCTL);
: 543      1537 2 !
: 544      1538 2 FCB[FCBSW_REFCNT] = .FCB[FCBSW_REFCNT] - 1;
: 545      1539 2 !
: 546      1540 2 ! Convert the access lock to reflect the remaining accessors.
: 547      1541 2 !
: 548      1542 2 !
: 549      1543 2 CONV_ACCLOCK (.LCKMODE, .FCB);
: 550      1544 2 !
: 551      1545 2 ! Release the quota cache lock, if there was one. Unlink and deallocate
: 552      1546 2 ! the quota cache block.
: 553      1547 2 !
: 554      1548 2 !
: 555      1549 2 QUOTA_CACHE = .CURRENT_VCB[VCBSL_QUOCACHE];
: 556      1550 2 IF .QUOTA_CACHE[VCASL_QUOCLKID] NEQ 0
: 557      1551 2 THEN
: 558      1552 2     BEGIN
: 559      1553 2     DEQ_LOCK (.QUOTA_CACHE[VCASL_QUOCLKID]);
: 560      1554 2     END;
: 561      1555 2 !
: 562      1556 2 DEALLOCATE (.QUOTA_CACHE);
: 563      1557 2 CURRENT_VCB[VCBSL_QUOCACHE] = 0;
: 564      1558 2 !
: 565      1559 2 RETURN 1;
: 566      1560 2 !
: 567      1561 1 END;

```

! end of routine DEACC_QFILE

```

                                .EXTRN KILL_BUFFERS, CONV ACCLOCK
                                .EXTRN LOCK_MODE, DEQ_LOCK
                                .EXTRN DEALLOCATE

                                000C 00000
                                7E      01 CE 00002
                                0000G CF      01 DD 00005
                                50      98 02 FB 00007
                                52      54 AA DO 0000C
                                54      A0 DO 00010

                                .ENTRY DEACC_QFILE, Save R2,R3
                                MNEGL #1, -(SP)
                                PUSHL #1
                                CALLS #2, KILL_BUFFERS
                                MOVL -104(BASE), R0
                                MOVL 84(R0), FCB

```

```

: 1461
: 1516
:
: 1521
:

```

08	AA		52	D0	00014		MOVL	FCB, 8(BASE)	:	
	50	98	AA	D0	00018		MOVL	-104(BASE), R0	:	1522
		54	A0	D4	0001C		CLRL	84(R0)	:	
			50	D4	0001F		CLRL	ACCTL	:	1524
		1C	A2	B7	00021		DECW	28(FCB)	:	1526
			05	13	00024		BEQL	1\$:	1527
	50	0100	8F	3C	00026		MOVZWL	#256, ACCTL	:	1528
		20	A2	B7	0002B	1\$:	DECW	32(FCB)	:	1530
			53	D4	0002E		CLRL	LCKMODE	:	1532
	51	1A	A2	3C	00030		MOVZWL	26(FCB), R1	:	1534
			51	D7	00034		DECL	R1	:	
1A	A2		51	B0	00036		MOVW	R1, 26(FCB)	:	
			51	D5	0003A		TSTL	R1	:	
			06	13	0003C		BEQL	2\$:	
			0000G	30	0003E		BSBW	LOCK MODE	:	1536
	53		50	D0	00041		MOVL	R0, LCKMODE	:	
		18	A2	B7	00044	2\$:	DECW	24(FCB)	:	1538
			52	DD	00047		PUSHL	FCB	:	1543
			53	DD	00049		PUSHL	LCKMODE	:	
0000G	CF		02	FB	0004B		CALLS	#2, CONV_ACCLOCK	:	
	50	98	AA	D0	00050		MOVL	-104(BASE), R0	:	1549
	52	5C	A0	D0	00054		MOVL	92(R0), QUOTA_CACHE	:	
		04	A2	D5	00058		TSTL	4(QUOTA_CACHE)	:	1550
			08	13	0005B		BEQL	3\$:	
		04	A2	DD	0005D		PUSHL	4(QUOTA_CACHE)	:	1553
0000G	CF		01	FB	00050		CALLS	#1, DEQ_LOCK	:	
			52	DD	00055	3\$:	PUSHL	QUOTA_CACHE	:	1556
00000000G	00		01	FB	00067		CALLS	#1, DEALLOCATE	:	
	50	98	AA	D0	0006E		MOVL	-104(BASE), R0	:	1557
		5C	A0	D4	00072		CLRL	92(R0)	:	
	50		01	D0	00075		MOVL	#1, R0	:	1559
			04	00078			RET		:	1561

; Routine Size: 121 bytes, Routine Base: \$CODE\$ + 02D6


```

: 569 1562 1 GLOBAL ROUTINE RET_QENTRY (Q_RECORD, ABD) : L_NORM =
: 570 1563 1
: 571 1564 1 !++
: 572 1565 1
: 573 1566 1 FUNCTIONAL DESCRIPTION:
: 574 1567 1
: 575 1568 1 This routine copies the specified quota file record into the
: 576 1569 1 result string area of the buffer descriptor packet. This routine
: 577 1570 1 must be called in kernel mode.
: 578 1571 1
: 579 1572 1 CALLING SEQUENCE:
: 580 1573 1 RET_QENTRY (ARG1, ARG2)
: 581 1574 1
: 582 1575 1 INPUT PARAMETERS:
: 583 1576 1 ARG1: address of quota file record
: 584 1577 1
: 585 1578 1 IMPLICIT INPUTS:
: 586 1579 1 NONE
: 587 1580 1
: 588 1581 1 OUTPUT PARAMETERS:
: 589 1582 1 ARG2: address of buffer descriptor packet
: 590 1583 1
: 591 1584 1 IMPLICIT OUTPUTS:
: 592 1585 1 NONE
: 593 1586 1
: 594 1587 1 ROUTINE VALUE:
: 595 1588 1 1
: 596 1589 1
: 597 1590 1 SIDE EFFECTS:
: 598 1591 1 NONE
: 599 1592 1 !--
: 600 1593 1
: 601 1594 1 BEGIN
: 602 1595 2
: 603 1596 2 MAP
: 604 1597 2
: 605 1598 2 Q_RECORD : REF BBLOCK, ! quota file record
: 606 1599 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH];
: 607 1600 2 ! descriptor arg
: 608 1601 2
: 609 1602 2 ! If the user provided a result length buffer, give him the length
: 610 1603 2 ! of the record.
: 611 1604 2
: 612 1605 2
: 613 1606 2 IF .ABD[ABD$C_RES], ABD$W_COUNT] GEQ 2
: 614 1607 2 THEN
: 615 1608 2 BEGIN
: 616 1609 2 (.ABD[ABD$C_RES], ABD$W_TEXT] + ABD[ABD$C_RES, ABD$W_TEXT] + 1) < 0, 16 > = DQF$C_LENGTH;
: 617 1610 2 END;
: 618 1611 2
: 619 1612 2 ! If the user provided a result string buffer, return as much of the
: 620 1613 2 ! quota record as will fit (zero filling the buffer).
: 621 1614 2
: 622 1615 2
: 623 1616 2 CH$COPY (DQF$C_LENGTH, .Q_RECORD, 0,
: 624 1617 2 .ABD[ABD$C_RES, ABD$W_COUNT],
: 625 1618 2 .ABD[ABD$C_RES, ABD$W_TEXT] + ABD[ABD$C_RES, ABD$W_TEXT] + 1);

```



```

: 631 1623 1 GLOBAL ROUTINE CONN_QFILE (ABD, FIB) : L_NORM NOVALUE =
: 632 1624 1
: 633 1625 1 !++
: 634 1626 1
: 635 1627 1 FUNCTIONAL DESCRIPTION:
: 636 1628 1
: 637 1629 1 This routine causes the quota file for the volume set to be
: 638 1630 1 connected and made active.
: 639 1631 1
: 640 1632 1 CALLING SEQUENCE:
: 641 1633 1 CONN_QFILE (ARG1, ARG2)
: 642 1634 1
: 643 1635 1 INPUT PARAMETERS:
: 644 1636 1 ARG1: address of buffer descriptor vector
: 645 1637 1 ARG2: address of user FIB
: 646 1638 1
: 647 1639 1 IMPLICIT INPUTS:
: 648 1640 1 CLEANUP_FLAGS: cleanup action and status flags
: 649 1641 1 CURRENT_RVN: RVN of currently selected volume
: 650 1642 1 CURRENT_VCB: VCB of currently selected volume
: 651 1643 1
: 652 1644 1 OUTPUT PARAMETERS:
: 653 1645 1 NONE
: 654 1646 1
: 655 1647 1 IMPLICIT OUTPUTS:
: 656 1648 1 PRIMARY_FCB: FCB created for quota file
: 657 1649 1
: 658 1650 1 ROUTINE VALUE:
: 659 1651 1 NONE
: 660 1652 1
: 661 1653 1 SIDE EFFECTS:
: 662 1654 1 directory searched, quota file accessed (FCB created, etc.)
: 663 1655 1
: 664 1656 1 --
: 665 1657 1
: 666 1658 2 BEGIN
: 667 1659 2
: 668 1660 2 MAP
: 669 1661 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH],
: 670 1662 2 : : buffer descriptor arg
: 671 1663 2 FIB : REF BBLOCK; : user FIB
: 672 1664 2
: 673 1665 2 LOCAL
: 674 1666 2 FCB : REF BBLOCK, : FCB of quota file
: 675 1667 2 HEADER : REF BBLOCK, : file header of quota file
: 676 1668 2 BUFFER : REF BBLOCK; : disk block buffer
: 677 1669 2
: 678 1670 2 BIND_COMMON;
: 679 1671 2
: 680 1672 2 EXTERNAL ROUTINE
: 681 1673 2 REBLD_PRIM_FCB : L_NORM NOVALUE, ! rebuild fcb from header
: 682 1674 2 BUILD_EXT_FCBS : L_NORM NOVALUE, ! build extension fcbs
: 683 1675 2 ARBITRATE_ACCESS : [ JSB_2ARGS, ! arbitrate file access
: 684 1676 2 SERIAL_FICE : L_NORM, ! serialize on given file
: 685 1677 2 FIND : L_NORM, ! find file in directory
: 686 1678 2 SWITCH_VOLUME : L_NORM, ! switch volume context
: 687 1679 2 SEARCH_FCB : L_NORM ADDRESSING_MODE (GENERAL), ! search FCB list

```

```

: 688      1680      2      READ HEADER      : L_NORM,      ! read file header
: 689      1681      2      CREATE_FCB      : L_NORM;      ! create an FCB
: 690      1682
: 691      1683
: 692      1684      2      ! Check caller privilege - must be "system".
: 693      1685
: 694      1686
: 695      1687      2      IF NOT .CLEANUP_FLAGS[CLF_SYSPRV]
: 696      1688      2      THEN ERR_EXIT (SS$_NOPRIV);
: 697      1689
: 698      1690      2      ! Find the quota file in the directory. The quota file must be located
: 699      1691      2      ! RVN 1 if this is a volume set.
: 700      1692
: 701      1693
: 702      1694      2      IF .CLEANUP_FLAGS[CLF_DIRECTORY]
: 703      1695      2      THEN FIND (.ABD, .FIB, 0);
: 704      1696      2      SWITCH VOLUME (.FIB[FIB$_FID_RVN]);
: 705      1697      2      IF .CURRENT_RVN GTRU 1
: 706      1698      2      THEN ERR_EXIT (SS$_BADQFILE);
: 707      1699
: 708      1700      2      ! Make sure the quota file is not already active.
: 709      1701
: 710      1702
: 711      1703      2      IF .CURRENT_VCB[VCBS$_QUOTAFCB] NEQ 0
: 712      1704      2      THEN ERR_EXIT (SS$_QFACTIVE);
: 713      1705
: 714      1706      2      ! Find the FCB, if any, and read the header.
: 715      1707
: 716      1708
: 717      1709      2      SERIAL_FILE (FIB [FIB$_FID]);
: 718      1710
: 719      1711      2      FCB = PRIMARY_FCB = SEARCH_FCB (FIB[FIB$_FID]);
: 720      1712
: 721      1713      2      HEADER = READ_HEADER (FIB[FIB$_FID]);
: 722      1714
: 723      1715      2      ! Create an FCB if none exists.
: 724      1716
: 725      1717
: 726      1718      2      IF .FCB EQL 0
: 727      1719      2      THEN
: 728      1720      2          PRIMARY_FCB = FCB = CREATE_FCB (.HEADER)
: 729      1721      2      ELSE
: 730      1722      2          IF .FCB [FCBS$_STALE]
: 731      1723      2          THEN
: 732      1724      2              REBLD_PRIM_FCB (.FCB, .HEADER);
: 733      1725
: 734      1726      2      BUILD_EXT_FCBS (.HEADER);
: 735      1727
: 736      1728      2      ! Check the quota file for suitability (contiguous, file format, etc.)
: 737      1729
: 738      1730
: 739      1731      2      IF NOT .HEADER[FH2$_CONTIG]
: 740      1732      2      OR .BBLOCK [HEADER[FH2$_RECATTR], FATS$_RTYPE] NEQ FATS$_FIXED
: 741      1733      2      OR .BBLOCK [HEADER[FH2$_RECATTR], FATS$_RATTRIB] NEQ 0
: 742      1734      2      OR .BBLOCK [HEADER[FH2$_RECATTR], FATS$_RSIZE] NEQ DQFSC_LENGTH
: 743      1735      2      THEN ERR_EXIT (SS$_BADQFILE);
: 744      1736

```



```

: 745      1737 2 ! Check access interlocks.
: 746      1738 2 !
: 747      1739 2 !
:G01 !ACG0468 1740 2 IF NOT ARBITRATE_ACCESS (FIBSM_WRITE, .FCB)
: 749-1    1741 2 THEN ERR_EXIT (SS$_ACCONFLICT);
: 750      1742 2 !
: 751      1743 2 ! Now hook up the quota file FCB.
: 752      1744 2 !
: 753      1745 2 !
: 754      1746 2 IF NOT KERNEL_CALL (MAKE_QFCB, .FCB)
: 755      1747 2 THEN ERR_EXIT (SS$_INSFMEM);
: 756      1748 2 !
: 757      1749 1 END;

```

```

! allocation failure on quota cache
! end of routine CONN_QFILE

```

				000C 00000	.EXTRN REBLD PRIM_FCB, BUILD_EXT_FCBS	
				AA E8 00002	.EXTRN ARBITRATE_ACCESS	
				24 BF 00006	.EXTRN FIND, SEARCH_FCB	
				04 00008	.EXTRN READ_HEADER, CREATE_FCB	
				06 E1 00009 1\$:	.ENTRY CONN_QFILE, Save R2,R3	: 1623
				7E D4 0000D	BLBS 1(BASE), 1\$: 1687
0B				AC 7D 0000F	CHMU #36	: 1688
				03 FB 00013	RET	
				AC D0 00018 2\$:	BBC #6, (BASE), 2\$: 1694
0000G				A0 3C 0001C	CLRL -(SP)	: 1695
				01 FB 00020	MOVQ ABD, -(SP)	
0000G				AA D1 00025	CALLS #3, FIND	
				76 1A 00029	MOVL FIB, R0	: 1696
				AA D0 0002B	MOVZWL 8(R0), -(SP)	
				A0 D5 0002F	CALLS #1, SWITCH_VOLUME	: 1697
				05 13 00032	CMPL -96(BASE), #1	: 1699
				8F BF 00034	BGTRU 6\$: 1703
			03CC	04 00038	MOVL -104(BASE), R0	: 1704
				04 C1 00039 3\$:	TSTL 84(R0)	: 1709
7E	08	AC		01 FB 0003E	BEQL 3\$: 1711
	0000G	CF		04 C1 00043	CHMU #972	: 1713
7E	08	AC		01 FB 00048	RET	: 1718
	00000000G	00		50 D0 0004F	ADDL3 #4, FIB, -(SP)	: 1720
	08	AA		50 D0 00053	CALLS #1, SERIAL_FILE	
7E	08	AC		50 D0 00056	ADDL3 #4, FIB, -(SP)	: 1713
	0000G	CF		01 FB 0005B	CALLS #1, READ_HEADER	: 1718
		52		50 D0 00060	MOVL R0, FCB	: 1713
				53 D5 00063	MOVL R0, FCB	: 1713
				10 12 00065	ADDL3 #4, FIB, -(SP)	: 1713
				52 DD 00067	CALLS #1, READ_HEADER	: 1718
	0000G	CF		01 FB 00069	MOVL R0, HEADER	: 1718
		53		50 D0 0006E	MOVL R0, HEADER	: 1718
	08	AA		53 D0 00071	TSTL FCB	: 1718
				0D 11 00075	BNEQ 4\$: 1718
		09	23	A3 E9 00077 4\$:	PUSHL HEADER	: 1720
				52 DD 0007B	CALLS #1, CREATE_FCB	: 1720
					MOVL R0, FCB	
					MOVL FCB, 8(BASE)	
					BRB 5\$	
					BLBC 35(FCB), 5\$: 1722
					PUSHL HEADER	: 1724

0000G	CF		53	DD	0007D		PUSHL	FCB		
			02	FB	0007F		CALLS	#2, REBLD_PRIM_FCB		
			52	DD	00084	5\$:	PUSHL	HEADER		1726
0000G	CF		01	FB	00086		CALLS	#1, BUILD_EXT_FCBS		
		34	A2	95	0008B		TSTB	52(HEADER)		1731
			11	18	0008E		BGEQ	6\$		
	01	14	A2	91	00090		CMPB	20(HEADER), #1		1732
			0B	12	00094		BNEQ	6\$		
		15	A2	95	00096		TSTB	21(HEADER)		1733
			06	12	00099		BNEQ	6\$		
	20	16	A2	B1	0009B		CMPW	22(HEADER), #32		1734
			05	13	0009F		BEQL	7\$		
		03BC	8F	BF	000A1	6\$:	CHMU	#956		1735
			04	04	000A5		RET			
	51		53	DD	000A6	7\$:	MOVL	FCB, R1		1740
	50	0100	8F	3C	000A9		MOVZWL	#256, R0		
			0000G	30	000AE		BSBW	ARBITRATE_ACCESS		
	05		50	E8	000B1		BLBS	R0, 8\$		
		0800	8F	BF	000B4		CHMU	#2048		1741
			04	04	000B8		RET			
			53	DD	000B9	8\$:	PUSHL	FCB		1746
0000V	CF		01	FB	000BB		CALLS	#1, MAKE_QFCB		
	04		50	E8	000C0		BLBS	R0, 9\$		
		0124	8F	BF	000C3		CHMU	#292		1747
			04	04	000C7	9\$:	RET			1749

; Routine Size: 200 bytes, Routine Base: \$CODE\$ + 0382


```

: 759      1750 1 GLOBAL ROUTINE MAKE_QFCB (FCB) : L_NORM =
: 760      1751 1
: 761      1752 1 ++
: 762      1753 1
: 763      1754 1 FUNCTIONAL DESCRIPTION:
: 764      1755 1
: 765      1756 1     This routine hooks up the specified FCB to be the FCB for the
: 766      1757 1     volume (set) quota file. This routine must be called in kernel mode.
: 767      1758 1
: 768      1759 1 CALLING SEQUENCE:
: 769      1760 1     MAKE_QFCB (ARG1)
: 770      1761 1
: 771      1762 1 INPUT PARAMETERS:
: 772      1763 1     ARG1: address of FCB to hook up
: 773      1764 1
: 774      1765 1 IMPLICIT INPUTS:
: 775      1766 1     CURRENT_VCB: VCB of volume
: 776      1767 1
: 777      1768 1 OUTPUT PARAMETERS:
: 778      1769 1     NONE
: 779      1770 1
: 780      1771 1 IMPLICIT OUTPUTS:
: 781      1772 1     NONE
: 782      1773 1
: 783      1774 1 ROUTINE VALUE:
: 784      1775 1     1 if successful
: 785      1776 1     0 if allocation failure on cache block
: 786      1777 1
: 787      1778 1 SIDE EFFECTS:
: 788      1779 1     quota file FCB hooked into FCB list and quota pointer
: 789      1780 1
: 790      1781 1 --
: 791      1782 1
: 792      1783 2 BEGIN
: 793      1784 2
: 794      1785 2 MAP
: 795      1786 2     FCB          : REF BBLOCK;    ! FCB to hook up
: 796      1787 2
: 797      1788 2 LOCAL
: 798      1789 2     QUOTA_CACHE : REF BBLOCK,    ! quota cache block allocated
: 799      1790 2     ACB         : REF BBLOCK;    ! AST control block within quota block
: 800      1791 2
: 801      1792 2 BIND_COMMON;
: 802      1793 2
: 803      1794 2 EXTERNAL
: 804      1795 2     CLUSGL_CLUB : ADDRESSING_MODE (GENERAL),
: 805      1796 2     SCH$GL_SWPPID : ADDRESSING_MODE (GENERAL);
: 806      1797 2     ! PID of swapper process
: 807      1798 2
: 808      1799 2 EXTERNAL ROUTINE
: 809      1800 2     ALLOCATE      : L_NORM ADDRESSING_MODE (GENERAL), ! allocate system dynamic memory
: 810      1801 2     CACHE_LOCK    : L_NORM, ! get special cache lock
: 811      1802 2     XQPSURLOCK_QUOTA : ADDRESSING_MODE (GENERAL);
: 812      1803 2     ! release lock with value block
: 813      1804 2
: 814      1805 2
: 814      1806 2 ! Allocate the cache block and link it to the VCB.

```

```

: 815 1807 2 !
: 816 1808 2 2
: 817 1809 2 2 QUOTA_CACHE = ALLOCATE (MAXU (.CURRENT_VCB[VCBSW_QUOSIZE], 1) * VCASC_QUOLENGTH
: 818 1810 2 2 + $BYTEOFFSET (VCASL_QUOCLIST), CACHE_TYPE);
: 819 1811 2 2 IF .QUOTA_CACHE EQL 0
: 820 1812 2 2 THEN RETURN 0;
: 821 1813 2 2 QUOTA_CACHE[VCASW_QUOSIZE] = MAXU (.CURRENT_VCB[VCBSW_QUOSIZE], 1);
: 822 1814 2 2 CURRENT_VCB[VCBSL_QUOCACHE] = .QUOTA_CACHE;
: 823 1815 2 2
: 824 1816 2 2 ! Initialize the AST control blocks in the quota cache header. One is
: 825 1817 2 2 ! used to post blocking AST's to the swapper to release cache entries.
: 826 1818 2 2 ! The other is used to trip the cache flush process to flush the entire
: 827 1819 2 2 ! cache.
: 828 1820 2 2
: 829 1821 2 2
: 830 1822 2 2 ACB = QUOTA_CACHE[VCASB_QUOACB];
: 831 1823 2 2 ACB[ACBSB_RMOD] = PSL$C_KERNEL + ACBSM_NODELETE;
: 832 1824 2 2 ACB[ACBSL_PID] = .SCH$G_SWPPID;
: 833 1825 2 2 ACB[ACBSL_AST] = XQPSUNLOCK_QUOTA;
: 834 1826 2 2 ACB = QUOTA_CACHE[VCASB_QUOFLUSHACB];
: 835 1827 2 2 ACB[ACBSB_RMOD] = PSL$C_KERNEL + ACBSM_NODELETE;
: 836 1828 2 2
: 837 1829 2 2 ! Bump up the access counts in the FCB to show an accessed file.
: 838 1830 2 2 ! Lock it against truncates.
: 839 1831 2 2
: 840 1832 2 2
: 841 1833 2 2 FCB[FCBSW_REFCNT] = .FCB[FCBSW_REFCNT] + 1;
: 842 1834 2 2 FCB[FCBSW_ACNT] = .FCB[FCBSW_ACNT] + 1;
: 843 1835 2 2 FCB[FCBSW_TCNT] = .FCB[FCBSW_TCNT] + 1;
: 001 :ACG0468 1836 2 2 FCB[FCBSW_WCNT] = .FCB[FCBSW_WCNT] + 1;
: 002 :ACG0468 1837 2 2
: 003 :ACG0468 1838 2 2 ! If the quota file is already write accessed, take out the cache lock
: 004 :ACG0468 1839 2 2 ! on the write access to prevent use of the cache.
: 005 :ACG0468 1840 2 2
: 006 :ACG0468 1841 2 2
: 007 :ACG0468 1842 2 2 IF .FCB[FCBSW_WCNT] GTRU 1
: 008 :CDS0009 1843 2 2 AND .BBLOCK [CURRENT_UCB[UCBSL_DEVCHAR2], DEV$V_CLU]
: 009 :CDS0009 1844 2 2 AND .CLUSGL CLUB NEQ 0
: 851-7 1845 2 2 AND .FCB[FCBSL_CACHELKID] EQL 0
: 852 1846 2 2 THEN CACHE_LOCK (.FCB[FCBSL_LOCKBASIS], FCB[FCBSL_CACHELKID], 2);
: 853 1847 2 2
: 854 1848 2 2 ! Finally enter the quota file pointer in the VCB.
: 855 1849 2 2
: 856 1850 2 2
: 857 1851 2 2 CURRENT_VCB[VCBSL_QUOTAFCB] = .FCB;
: 858 1852 2 2
: 859 1853 2 2 CLEANUP_FLAGS[CLF_DEACQFILE] = 1;
: 860 1854 2 2
: 861 1855 2 2 RETURN 1;
: 862 1856 2 2
: 863 1857 1 END;

```

! end of routine MAKE_QFCB

```

.EXTRN CLUSGL CLUB, SCH$GL SWPPID
.EXTRN ALLOCATE, CACHE_LOCK
.EXTRN XQPSUNLOCK_QUOTA

```



```
: 864      1858  1  
: 865      1859  1 END  
: 866      1860  0 ELUDOM
```

PSECT SUMMARY

```
: Name                Bytes                Attributes  
: $CODE$              1284 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
```

Library Statistics

```
: File                ----- Symbols ----- Pages Processing  
:                   Total   Loaded   Percent   Mapped   Time  
: _$255$DUA18:[SYSLIB]LIB.L32;1  18619    103      0       1000    00:02.0
```

COMMAND QUALIFIERS

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

```
: Size:                1284 code + 0 data bytes  
: Run Time:            01:05.5  
: Elapsed Time:       01:42.9  
: Lines/CPU Min:      1703  
: Lexemes/CPU-Min:    55798  
: Memory Used:        321 pages  
: Compilation Complete
```


