bcc	title	prefix/class-number.revision
	MCALLS FOR MANIPULATING PMT AND SPT	PMTSPT/W-2

checked	authors	approval date	revision date
RAINER SCHULZ	Jack Freeman	5/9/69	
checked		**classification** Working Document	
approved Mel		**distribution** Company Private	**pages** 15

ABSTRACT and CONTENTS

The names, argument lists, values, and error returns of the system calls which manipulate the Process Memory Table are described. This is a working description intended for the use of people currently coding sub-processes. A few of the functions which diddle the Sub-process Table are also described.

MCALLs for Manipulating PMT

The following pages are intended to provide working descriptions of the Monitor Calls which Utility and User Sub-processes will use to manipulate the Process Memory Table. The names, argument lists, and values of both successful and unsuccessful returns are detailed. An earlier document described the actions performed by these functions.

One operation, "Put page p of file f in PMT entry n," is not mentioned here. It is described in the documents on "System Calls to the Basic File System."

In all the descriptions, S stands for the calling sub-process.

(1) ACQPMT - Acquire PMT Entry

Declaration: SYSTEM FUNCTION ACQPMT(N);

Success Return: RETURN N;

Failure Returns:

(1) FRETURN Ø if S does not have APMT status

(2) FRETURN 1 if PMT(N) is not available. This can be true for a number of reasons:

(a) N $>$ NPMTE

(b) N $\neq$ -1 and N $<$ M, where M = 1 if S has Master Sub-process (MSP) status and M = NMPMTE + 1 if not

(c) N = -1 and there are no free PMT slots in the interval [M, NPMTE], where M is as defined in (b).

(d) PMT[N] is not free.

(2) NPPMT - Create a New Private Memory Page and Put Its Real Name in PMT

Declaration: SYSTEM FUNCTION NPPMT(N);

Success Return: RETURN;

Error Returns:

(1) FRETURN Ø if S doesn't have AM status

(2) FRETURN 1 unless N is in the interval [1, NPMTE] and S controls PMT[N].

(3) FRETURN 2 if PMT[N] isn't empty.

(3) RNPMT - Put a Given Real Name in PMT

Declaration: SYSTEM FUNCTION RNPMT(N, COMPLEX RN);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø if S does not have the privileged SD status.

(2) FRETURN 1 unless N is in the interval [1, NPMTE] and PMT[N] is controlled by S

(3) FRETURN 2 unless PMT[N] is empty.

The real name is taken out of words Ø, 1 and 2 of the argument RN.

(4) CLRPMT - Clear PMT Entry

Declaration: SYSTEM FUNCTION CLRPMT(N);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø unless N is in the interval [1, NPMTE] and S controls PMT[N].

(5) DELPMT - Delete PMT Entry

Declaration: SYSTEM FUNCTION DELPMT(N);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø unless N is in the interval [1, NPMTE] and S controls PMT[N].

(6) SPMTAL - Set Access Lock for PMT Entry

Declaration: SYSTEM FUNCTION SPMTAL(N,L);

Success Return: RETURN;

Failure Returns:

(1) FRETURN $\emptyset$ unless N is in the interval [1, NPMTE] and S controls PMT[N] or $AL(N) \wedge KEY(S) \neq \emptyset$. In the latter case AL(N) gets set to $(L \ KEY(S) \ AL(N) \ \overline{KEY(S)})$

The eleven bits of the lock field are taken out of bits 13-23 of L.

(7) SPMTCL - Set Control Lock for PMT Entry

Declaration: SYSTEM FUNCTION SPMTCL(N,L);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø unless N is in the interval [1, NPMTE] and S controls PMT[N].

(2) FRETURN 1 if (L AND NOT (CL(PMT[N]) OR KEY(S))) ≠ Ø.

The eleven bits of the lock field are taken out of bits 13-23 L.

(8) SPMTRO - Set Read Only Bit for PMT Entry

Declaration: SYSTEM FUNCTION SPMTRO(N,A);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø unless N is in the interval [1, NPMTE] and S controls PMT[N].

(2) FRETURN 1 if A = Ø and PMT[N] is a file page with RO currently set.

The RO bit is taken out of bit 23 of A.

(9) READPMT - Read PMT Entry

Declaration: SYSTEM COMPLEX FUNCTION READPMT(N);

Success Return: RETURN PMT[N];

Failure Returns:

(1) FRETURN Ø if N isn't in the interval [1, NPMTE].

(1) ACQSPT - Acquire an SPT Entry

Declaration: SYSTEM FUNCTION ACQSPT(T);

Success Return: RETURN T;

Failure Returns:

(1) FRETURN Ø unless S has Acquire SPT (ASPT) status

(2) FRETURN 1 if SPT entry T is unavailable. This will be the case if any of the following holds:

(a) T > NSPTE

(b) T ≠ -1 and T < 1

(c) T = -1 and no SPT entries are free

(d) SPT(T) is not free

Action: If T is -1, SPT is searched for a free entry and the index of the first such entry is used for T.

All fields of SPT[T] are cleared and then the following things are done.

(1) NAME(T) ◄- 2↑(T-1)

(2) USP(T) ◄- (NAME(S) if TF(S) else USP(S))

(3) KEY(T) ◄- NAME(T)

(4) FATHER(T) ◄- S

(5) UNO(T) -- PRUNO, the Process User Number T.

(6) T is incorporated into the SPT control structure by putting NAME(T) into KEY(S) and the KEYs of all S's ancestors.

The SPT index, T, is the value of the call.

(2) DELSPT - Delete an SPT Entry

Declaration: SYSTEM FUNCTION DELSPT(T);

Success Return: RETURN;

Failure Returns:

(1) FRETURN Ø unless T is in the interval [1, NSPTE] and S controls SPT[T]

(2) FRETURN 1 if T appears on the Sub-process Call Stack (SPCS) at any level except the very top

Action: NAME(T) is removed from all KEYs, USPs and CALL MASKs in SPT and from all ACCESS LOCKs and CONTROL LOCKs in OFT and PMT. Any elements of OFT and PMT whose CONTROL LOCKs become Ø as a result of this are DELETEd.

If any SPT entries have T as their FATHERs, they are given FATHER(T) instead.

NAME(T) is cleared, making SPT(T) free.

If T = S, so that T is on the top of SPCS, SPCS is popped up and the RETURN from the MCALL goes to the caller of T.

(3) SETSPT - Set the Contents of an SPT Entry

Declaration: SYSTEM FUNCTION SETSPT (T, ARRAY V);

Success Return: RETURN;

Failure Returns: There are a large number of these because many of the fields of an SPT entry can be written only in prescribed ways.

(1) FRETURN Ø unless T is in the interval [1, NSPTE] and S controls SPT[T].

(2) FRETURN 1 if S is trying to changing TF of T from User to Utility and doesn't have Create Utility Sub-process (CUSP) status.

(3) FRETURN 2 unless NAME(V) = NAME(T)

(4) FRETURN 3 unless USP(V) = USP(T) or USP(V) = USP(S) or USP(V) is a Utility Sub-process controlled by S

(5) FRETURN 4 if (KEY(V) AND NOT (KEY(T) OR KEY(S))) ≠ Ø

(6) FRETURN 5 unless all sub-processes whose names appear in CM(V) currently exist.

(7) FRETURN 6 unless FATHER(V) is FATHER(T) or is controlled by S.

(8) FRETURN 6 if FATHER(V) is T or has T as an ancestor

(9) FRETURN 6 if the number of ancestors of FATHER(V) is greater than the number of SPCS entries which would be available if the topmost occurrence of T on SPCS were the top of SPCS, or something like that.

(10) FRETURN 7 if EP(V) is not in the ring selected by TV(V)

(11) FRETURN 8 if any of KF(V), KF1(V), KF2(V) are different from the corresponding fields of SPT[T]

(12) FRETURN 9 if EG(V) is not in the ring selected by TF(V)

(13) FRETURN 1Ø if UNO(V) is different from UNO(T) and S does not have Master Sub-proces (MSP) status

(14) FRETURN 11 if any of TAK(V), PAK1(V), and PAK2(V) are different from the corresponding fields of SPT[T]

(15) FRETURN 12 if (TEM(V) AND NOT (TEM(T) OR TEM(S))) ≠ Ø

(16) FRETURN 13 if (TM(V) AND NOT (TM(T) OR TEM(S))) ≠ Ø

(17) FRETURN 14 if (CAPB(V) AND NOT (CAPB(T) OR CAPB(S))) ≠ Ø

(18) FRETURN 15 if (SB(V) AND NOT (SB(T) OR CAPB(S))) ≠ Ø

(19) FRETURN 16 unless the PMT index in every byte of MAP(V) either is identical with the one in the corresponding byte of MAP(T), or points to a PMT entry to which S has access, or is Ø.

Action: If any caller ever manages to satisfy all the above conditions, SPT[T] will be written from the array, V.

4/30/69

Format of Sub-Process Table (SPT)