12/18/69

The BEAD Users Guide

This document describes the interim system BEAD.

1.0 The BEAD

The BEAD is the root (initial) subprocess in every process. It is created by IPROC when (CONTROL) (SHIFT) P is entered on a null teletype. The BEAD is divided into three sections: 1) the Teletype Driver, 2) the Command Processor, and 3) the Request Processor. The Teletype Driver is described elsewhere and will not be mentioned here; the Command Processor executes commands typed by the user; the Request Processor handles requests made by subprocesses running under the BEAD.

The BEAD maintains a list of objects in a file called MASTR,OPERATE and a corresponding list of capabilities for these objects in a C-list called MASTC,OPERATE. Each object is identified by a seven character object name and an eight character user name. Associated with each object is a busy bit, which is set whenever anyone has a hold on the object. (See Appendix A for description of the directory.)

A subsystem resides on a file object. By typing a command, it is possible to direct the BEAD to take a named file object and to create a subprocess from the specifications contained at the beginning of the file object. (See Section 6.0 for the subprocess specification formats.)

2.0 The Command Processor

The syntax for a command is:

<command> ::= {<word>,} ... (CR)

<word> ::= any letters except ','

Note: Trailing commas on a command are ignored.

Commands to the BEAD fall into four categories: object commands, which manipulate objects by name; debugging commands; overall control commands; and response commands, which answer questions posed, or comments made, by the BEAD.

2.1 Object Commands

Object commands should, in general, only be typed in a clean BEAD, i.e., one in which no subsystems are active. The BEAD will allow most object commands to be accepted while a subsystem is active, but the user must be extremely careful and must know how the command will interact with the active subsystem.

2.1.1 DELETE,{*name*},{*uname*}

The named file is deleted from the system. If *uname* is omitted, the current user name is supplied. K is equivalent to DELETE

2.1.2 SNATCH,{*name*},{*uname*}

The busy bit in the named file is cleared. S may be used instead of SNATCH.

2.1.3 CALL,{*name*},{*uname*},{*params*}

The subprocess specifier is obtained from the named file and a subprocess is created. If any additional parameters, *params*, are typed on the CALL command, they are passed to the subsystem. A maximum of two additional parameters may be specified and are passed in X4 and X5. C may replace CALL. This command will not be accepted if a a subsystem is active.

2.1.4 RECALL,{*objname*},{*user name*},{*params*}

This command is used to restart an already active subsystem. It is usually typed after an error is intercepted or after the user has interrupted his process. ~~The class code for the subsystem "*objname*, *user name*" is obtained from the directory and a call operation is created to call the subsystem.~~ The call stack is cleared before the subsystem is called. A maximum of two *params* may be specified as for CALL.

2.2 Debugging Commands

The BEAD includes an octal debugger which is used mainly for debugging the processors. In the following commands,

<number list> ::= {*octal number*,} ... ,

Thus 4,5,6,, is a number list. A number list appearing at the end of a command may omit both trailing commas. The value of a number list is the sum of its elements, e.g., 17_8 is the value of the number list above.

2.2.1 PF,*name*,{*uname*},<number list> <number list>

This command prints on the teletype the contents of the named file starting at the address given by the value of the first number list. The number of words printed is given by the value of the second number list. For example,

PF,SCOPE,S,56,47,,10 (CR)

would print on the teletype 10_8 words from the file SCOPE,S starting at address 125_8.

2.2.2 E,*name*,{*uname*},*half1*,*half2*,<number list>

A 60 bit word is constructed using the 10 octal digits specified by *half1* as the upper 30 bits and the 10 octal digits specified by *half2* as the lower 30 bits. This word is written into the named file at the address specified by the value of number list.

2.2.3 P,<number list> <number list>

The contents of core are printed starting at the value of the first number list and continuing for the number of words specified by the value of the second number list.

2.2.4 EC,*half1*,*half2*,<number list>

This command functions in the same way as the E command except that the word is stored into core at the address specified by the value of <number list> .

2.2.5 VIEW[,<number list>]

The call stack entry for the n-th subprocess (where n is the value of <number list>) in the call stack is displayed. VIEW (CR) is equivalent to VIEW,1 (CR) .

2.3 General Commands

2.3.1 USER,*uname*

This command sets the current user name to *uname*. The default for a newly created BEAD is "~~youdummy~~".

2.3.2 PURGE

This command clears the call stack and destroys the subsystem currently active. Warning: If a subsystem was active and possessed any objects, the information in the directories will not be correct and could lead to errors later. It is best to use the RECALL command and leave the subsystem gracefully.

2.3.3 BLOCK,*blksize*

The block size for all files created by the BEAD is set to *blksize*, which must be a power of two.

2.3.4 LIST

This command is used to produce a list of all files currently in the system.

2.3.5 RECOVER[*fname*[,*uname*]]

This command enters an event on a diddle event channel located in the directory under *fname*,*uname*. It should be used discriminately and actually should never be needed. The default for *fname*,*uname* is the directory lock. The default for *uname* is "operate".

2.3.6 CLEAR[*fname*[,*uname*]]

This command removes all of the diddle events located in the directory under *fname*,*uname*, thereby locking access to the directory. It should never be needed. The defaults are the same as for RECOVER.

3.0 The Request Processor

The request processor directs subsystem teletype I/O operations, retrieves objects from the directory for subsystems, updates directory information for subsystems, and performs control operations.

3.1 Teletype I/O Parameters

B1 - Pointer to string description or input buffer

B6 - function: 5 ⇒ input; 6 ⇒ output

3.2 Directory Operations

B1 - Pointer to 4 words area to hold directory entry

B7 - Capability index to return a capability for named object

X1 - Name of object, in BCD left-justified. (Needed only for LOCATE.)

X2 - User name of object or 0 if default user name is to be used. (Needed only for LOCATE.)

B6 - Function: 0 ⇒ locate; 1 ⇒ update; 2 ⇒ delete; 3 ⇒ call

On a locate (0): if the object is busy, the user is informed and has the option of TRYing again or CONTINUEing anyway; if the object does not exist, a file is created.

3.3 Other Requests

3.3.1 STOP (B6 = 4)

..STOP is typed by the BEAD. A response command, RETURN, is available to return to the subsystem.

3.3.2 CHARACTER OUT (B6 = 7, X1 = character)

The character in X1 is typed out on the teletype.

3.3.3 LOCK (B6 = 8)

The system is locked. Used only by the system dumper to freeze the system.

3.3.4 UNLOCK (B6 = 9)

The system is unlocked.

4.0 Special Objects

The following is a list of system operations that are in the directory. They are primarily used in subprocess descriptors to obtain operations needed in that subprocess. All have OPERATE as their user name.

Object name	Description of object
ALLOC	Allocation Block
READ	Read a file
WRITE	Write on a file
SENDE	Send an event
GETE	Get an event or hang
CCLIST	Create a C-list
CFILE	Create a file (Should not be used; call BEAD to locate file)
CBLK	Create file block
CPROC	Create process
CEVENT	Create an event channel
CSPROC	Create subprocess
CCC	Create class code
SAVE	Save registers
RESTOR	Restore registers
DSCAP	Display capability from full C-list
DSARB	Display capability from arbitrary C-list
MVECAP	Move capability within full C-list
CAPIN	Move capability into full C-list
CAPOUT	Move capability from full C-list
ESMGEN	Set ESM in process
ESMLOC	Set Local ESM
MKOPR	Make operation
RETURN	Subprocess return
FRETRN	Subprocess F-return
FIXC	Change "none" PS to fixed capability in operation
FIXD	Fix datum
UDAT	User datum
UCAP	User-supplied capability
ACAP	Any capability
ADDOPT	Add option bit(s)
PROBE	Check for existence of file block

Object name	Description of object
JUMP	Subprocess jump
COPYOP	Copy operation
DELBLK	Delete block from file
DELFIL	Delete file (Should not be used unless CFILE was used.)
REDSHP	Read shape of a file
MAPZRO	Zero map entry
MPCHRW	Map change (Read/write)
MPCHRD	Map change (Read only)
MOVBLK	Move file block
DISMAP	Display map
DISPST	Display entire call stack
DISSEN	Display stack entry
DISFMAP	Display full map
DELCL	Destroy C-list
PINT	Send process interrupt
ADDORD	Add order to operation
MODPC	Modify P-counter in stack
SELF	IPROC's C-list

4.1 Subsystems Available

4.1.1 SCOPE Simulator

CALL,SCOPE,S

and DONE (CR) when finished

See SCOPE Simulator document for details.

4.1.2 EDITOR

CALL,EDITOR,S,*file*

and F (CR) when done

See EDITOR document for details

4.1.3 GETTPE

C,GETTPE,S

All files written on a tape with an earlier call to DUMPTPE are reloaded. Files on the tape supersede files in the system.

4.1.4 DUMPTPE

C,DUMPTPE,S,*fname,uname*

DUMPTPE dumps files on tape in a format such that GETTPE can restore them, and deletes those files from the system.

DUMPTPE expects to find the list of files (and their user names to be dumped on the file *fname,uname*. This list is in the following format:

$Filename_1$
$Username_1$
$Filename_2$
$Username_2$
$Filename_3$
$Username_3$
Etc.

This file is not saved on the tape and remains in the system. The first file on the tape does, however, contain a modified list of the files and is used by GETTPE.

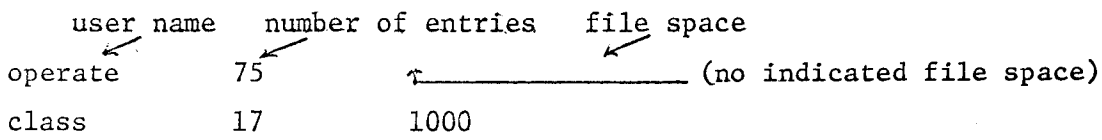
4.1.5 DLIST

DLIST is a routine designed to display relevent portions of the BEAD directory. It can be called as follows:

1. CALL,DLIST,S

   This will print a list of all user names appearing in the directory. With each user name will be the number of entries (in octal) of that user name and the total file space (in octal) indicated by the directory to be used by files of that user name.

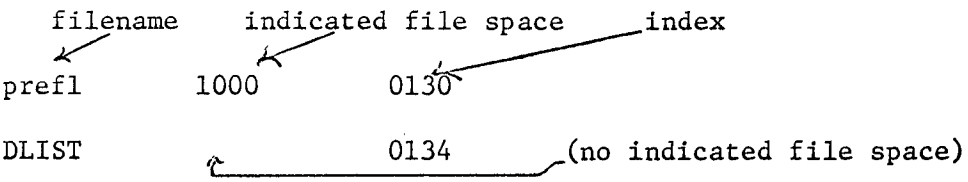
E.g.



2. CALL,DLIST,S,*username*

This will print a list of all file names appearing in the directory with the given user name. With each filename will be the file space (in octal) indicated by the directory to be used by that file and the directory index of the entry (in octal). The actual file address within "MASTR,OPERATE" of the entry is 4*index. (Each object has a 4 word entry. See Appendix A.)

E.g.



4.1.6 DUMP

CALL,DUMP,S

This routine makes a new deadstart tape.

4.1.7 GETFILE

CALL,GETFILE,S,*file*

This routine loads the next tape file onto the specified file in SCOPE simulator format. (See SCOPE document for SCOPE simulator format files.)

4.1.8 PRINTER

fname,username are the file name and user name of the file to be printed. Immediately after being called, the printer driver will ask whether or not the first character of each line should be interpreted as a SCOPE carriage control character. If this action is desired type YES (CR). The printer driver looks for a Y in col. 1. Therefore Y (CR) is equivalent to YES (CR) while b YES (CR) is not. The default option (i.e., no Y in col. 1) will be single spacing with auto page eject, and the first character of each line being printed.

Note that there is a lockout on the printer. If a message "EVCH CLEARED" appears, the user has the printer. If, after this message appears, an interrupt is sent and the printer is not allowed to finish normally (i.e., no restore is done), this channel must be re-set by using RECOVER. The name of the lockout is PRNLOCK,OPERATE.

ASCII-CDC DISPLAY CODE MAPPING

ASCII	Printer Char	ASCII	Printer Char
b	b	/	/
!	v	0	0
"	↓	1	1
#	≡	2	2
$	$	3	3
%	%	4	4
&	^	5	5
'	≠	6	6
(	(	7	7
)	)	8	8
*	*	9	9
+	+	;	;
,	,	:	:
—	—	<	<
.	.	=	=

ASCII	Printer Char
>	>
?	≥
@	≤
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[	[
\	↱
]	]
↑	↑
←	→
`	≠

ASCII	Printer Char
a	A
b	B
c	C
d	D
e	E
f	F
g	G
h	H
i	I
j	J
k	K
l	L
m	M
n	N
o	O
p	P
q	Q
r	R
s	S
t	T
u	U
v	V
w	W
x	X
y	Y
z	Z
{	(
|	b
}	)
~	b
all others	b

5.0 BEAD Type Outs

5.1 INTERRUPTED

This is typed out when IPROC sends an interrupt to your process. IPROC will send an interrupt upon receiving (CTRL) (SHIFT) P. The P-counter can be obtained by a VIEW command. The registers are saved in locations 32_8 - 41_8 of the BEAD. The length of the BEAD is 3450_8. A response command RESTORE is available to restore the registers and continue where the process was interrupted.

5.2 ERROR INTERCEPTED

This is typed out whenever an error reaches the BEAD. Actions to be taken are similar to interrupts.

5.3 ILLEGAL COMMAND

Typed by the command processor to indicate that the command just entered was illegal.

5.4 BAD ACTION DIRECTIVE

Typed by the Request Processor upon receiving a bad B6.

5.5 *fname uname* IS BUSY

Typed when a request is made for a busy object. Two response commands are available:

CONTINUE (CR) is typed to continue anyway; the object will be given to the requestor, so care must be taken

TRY (CR) is typed to try again and check to see if the busy bit has been cleared. If the file is still busy, the busy message will re-appear.

5.6 BEAD HERE

This message is typed when the BEAD is in a clear state.

5.7 OK

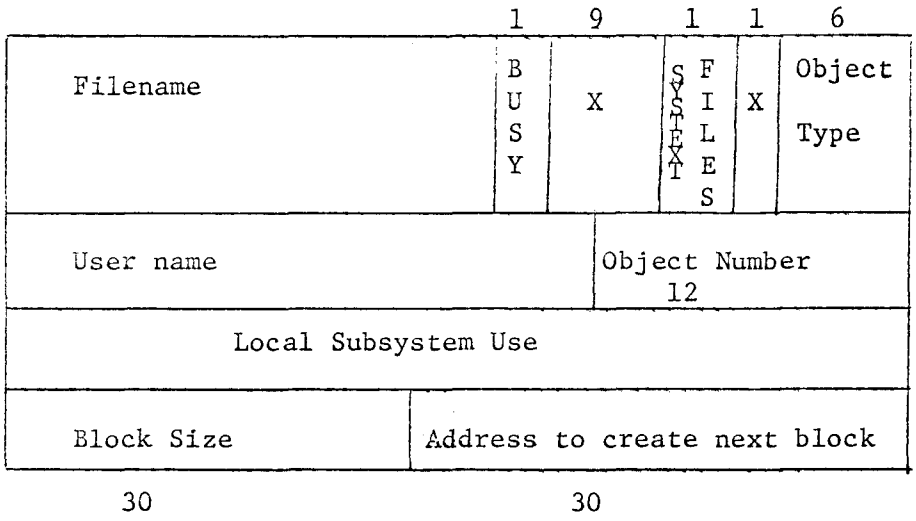
A standard response to a command signifying successful completion.

5.8 ENTER USER NAME

This message is typed when the BEAD is first created by IPROC. A user name should be entered via the USER command.

APPENDIX A

Format of Directory Entries



On a locate, the above entry is read.

On a delete, the above entry in the subsystem address space is used to obtain the object number and to zero the entry in the directory and destroy the object.

On an update, the above entry is rewritten into its place in the directory.

Subprocess Descriptors

		Number of words
	∅	2
	Name of class code LJZf	1
α	# of map entries	2
	Space for compiled map	1
	FL of subprocess	1
	entry point	1
β	length of C-list	1
	length of scratch file	1
	map specifiers	(α)*6 words atmost
	-1 (flag to mark end of map specifiers)	1
	C-list specifiers	(β)*2 words at most
	0 (flag to mark end of map specifiers)	1

Map specifiers

fname
uname
file address
CM address
word count
read only if 1

Use fname of ∅ to specify the scratch file. Generally, code for subprocess is on the same file as the descriptor.

C-list specifiers

fname
uname

Use fname of ALLOC and uname of OPERATE for filler. Trailing entries can be ignored. Note: Index of first specifier is 14.

The first 14 entries of the C-list are filled as follows:

0 - Allocation block
1 - Call on BEAD operation
2 - Call on yourself operation
3 - READ operation
4 - Write operation
5 - Send event operation
6 - Hang operation
7 - BEAD class code
8 - empty
9 - empty
10 - This C-list
11 - empty
12 - Your scratch file
13 - Your class code