Command Processor
Preliminary Document

I) Preliminary Info

A) The subprocess structure at the command level of a process contains 4 subprocesses. 1) The Bead ghost, which is a restage of the old bead, intercepts all of the old bead calls, user errors and interrupts. 2) BEAD Services, which does the simulation of the old bead calls, as well as other services. 3) The line collector, which talks to the teletypes. 4) The command processor, which handles primary control of the process, in conversation with the users teletype, and which this document describes.

B) logical program structure
The command processor actually contains 3 separate programs. 1) The command processor proper, which is used for calling subprocesses; 2) Services, which is used for a number of utility functions; 3) and the Bead ghost, which is called by the Bead ghost subprocess to handle errors and interrupts.

C) With one exception, what each of these programs uses the same form of command line; a verb followed by 0 or more parameters. The verb and parameters are separated by 1 or more blanks, and the command line is terminated by 0 or more blanks followed by a carriage return. The line may have initial blanks which are ignored.

The exceptions in the subsystem calls accepted by the C.

D) With few exceptions, what the parameters have a common structure, described below, which designate either a 1) datum, an 2) object or the location of a 3) datum or 4) object.

II) Command Processor

The command processor accepts 2 types of command lines. The first type is the standard command line of a verb with 0 or more parameters. These cause special actions. First I list those which will appear in the final version.

Note: In describing standard commands, I state the verb as typed, followed by what is expected of its parameters, if any.

i) ~~Services~~ SERVICES

Causes control to go to Services

Next I list those used in the test version only.

i) USERBUG

Calls Bruce's debugger

ii) ~~IPROC~~ JPROC

Causes simulated JPROC ~~Forum~~ town, should be done exactly once per call of XIPROC.

iii) ~~Keith~~ KEITH

Makes debug call on Keith's LOAD/DUMP/RECOVER

Now also attempts to simulate rest of JPROC

iv) ~~Bill~~

BILL Identifier Identifier

Makes debug call on Bill Bridges Bead simulator.

(Actually a Bead B6=0 call on Bead ghost)

vi) Paul calls directory system with a debug call.

viii) Bruce calls disk system with a debug call

v) Crunch

Causes a Bead B6=4 call on Fake Bead ghost. Used for debugging parts of command processor subprocess, dangerous to use.

The second type of command accepted by the command processor is a subprocess call. This command starts with a standard parameter, nameing a file containing a subprocess descriptor for the desired subprocess. This is followed by 0, 1 or 2 bead type parameters, separated by blanks. The line is terminated by 0 or more blanks followed by a carriage return. A bead type parameter is either an identifier or an integer. (See standard parameters.)

III) Services and Bead Ghost

All command lines here are of the same form, a verb followed by 0 or more parameters. What follows is a list of the verbs, and what they expect to be provided by their parameters. Most parameters are standard. Most verbs are common to services and Bead ghost; some are used by one program only and are so indicated. All verbs are to be typed as written.

1.1) FIN (Services only)

Returns control to the command processor

1.2) RETRY (Bead ghost only)

If the Bead ghost was called by an error or interrupt, and the calling subprocess is in the middle of an XJ then that XJ will be repeated; otherwise same as return.

1.3) RETURN (Bead ghost only)

The subprocess calling the Bead ghost continues at its next instruction.

1.4) PURGE

Reduces subprocess call stack to initial value. Deletes all user subprocesses.

2.1) P.ASCII

Changes mode of PDATA to 4 bits, 7 bits, ..., 7 bits

2.2) P.FULL

Changes mode of PDATA to 60 bits

2.3) P.INST

Changes mode of PDATA to 15 bits, 15 bits, 15 bits, 15 bits

3.1) IN.OCT

Causes all integers without trailing 'D' to be read in octal.

3.2) IN.DEC

Causes all integers without trailing 'B' to be read in decimal

4.1) PDATA Datum

Prints the datum in current print mode

4.2) PDATA Datum.Loc Count

Prints several datum words in current print mode. An interrupt will stop the printing with no damage. (Except in current test mode, while printing from a disk file.)

4.3) PCAP Object

Prints in octal the contents of the 2 words of the capability.

4.4) DISPLAY (See next page)

5.1) MDATA Datum Datum.Loc

Moves datum to given datum.loc, 1 word only

5.2) MCAP Object Object.loc

Moves object to given object.loc, 1 object only
If the object is a disk system object, and the object.loc is a directory.loc, it forms a hard link.

6.1) NEWV Identifier

Creates a variable of given name. Maximum of 8 characters in the identifier. (Current maximum number of variables is 10.) A variable can hold either objects or data.

6.2) ~~NILLV~~ KILLV Identifier

Destroys named variable

7.1) VIEW Datum

Prints out the contents of the 3 words of a subprocess call stack entry. The call stack entry named depends upon whether in Bead ghost or services.

A) Services

0 Services itself (~~printer~~ peripheral and XJ address ~~ball~~ bead)

1 Beads, which called services

2 Builder

etc.

B) Bead ghost

-1 Bead ghost itself (~~printer~~ and XJ address ~~ball~~)
(~~Portion~~ ~~lives~~ in command processor)

0 The Bead ghost subprocess

1 Calling subprocess

etc.

8.1) NEWDF Directory.Loc

Creates a disk file, of current shape, in the given name. The access key part of the directory.loc is ignored. Makes a non scratch entry.

8.2) NEWDR Directory.Loc Datum Datum

Creates a new directory, of ~~site~~ given in first datum, with given name. The access key part of the directory.loc is ignored. Makes a non scratch entry. The second datum is the accounting block flag.

I now list actions which are in for test purposes only. In the final version they will either disappear entirely or appear in heavily modified form.

T.1) ~~USER~~ Identifier (services only)

Sets the running user name and creates a temporary directory. Should be called once only per call of XROOT.

T.2) ~~DEATH~~

Destroys this user process

T.3) ~~BUB~~ BUG

Calls Bead with ..STOP . (The real Bead, ancestor of XROOT.)

T.4) CRUNCH

Causes a Bead to STOP call (B6=4) to be made on fake Bead ghost. Used for debugging parts of the command processor. Even more dangerous here than under the command processor.

Additional test commands for SERVICE & Bead ghost

BEADSEMSW

causes 4 words of error information from BEADS
To ~~print~~ print out. makes a call on BEADS to
get the information. See document on errors
in BEADS

Testonly
BEADSDBUG

causes Beads to make a STOP call on
some Bead ghost. (causing 'disk system has crashed'
message.) Dangerous to use.

Testonly
GETBDFILE param1 param2 (services only)

gets a file (or other object) from old Bead. places
in variable BEADF.

NEWBLK filecc

creates a data block in the given file at the given address.
disk or ecs file

KILLBLK filecc

~~creates a~~ deletes a data block from the given file at the
given address. disk or ecs file.

Test #17

COPYBOFILE param1 param2 directory.loc

Obtains an ECS file from the old Bead with name param1, param2
Creates a disk file with same bytesize in given
directory with given name, copies the ECS file to the
new disk file.

IV) Standard Parameters

These are used to define one of the following:

Datum -60 B.F word

Object -Capability

Datum.Loc -Location of a 60 B.F word, examples are a file and file address, or a subprocess memory address.

Object.Loc Location of a capability, examples are a list and index; or a directory name and access key.

Identifier String of characters

A) A standard parameter is written as a sequence of identifiers, integers or punctuation marks. Identifiers are composed of letters, digits, periods and quoted characters (see below.). An identifier starts with either a letter or a quoted character.

Integers are composed of digits, with possibly a trailing 'B' or 'D'. With a trailing 'B' it is read in octal, with a trailing 'D' it is read in decimal. Otherwise it is read in the current input mode. Maximum valve is 60 Bits.

A single quote mark quotes the following single character. It makes that character part of an identifier. Any character except carriage return can be quoted. In particular, a single quote mark itself can be quoted.

All other single characters not part of identifiers or integers are punctuation marks.

B) Standard parameters are either atomic expressions or compound expressions. Every expression has an immediate valve and a valve determined by content. A content specifies a type of valve desired. Examples are Datum, Object, Datum.loc or Identifier. Some contents can be very specific, for example, Directory.loc .

A content valve is computed by first computing the immediate valve, compare this with the content and do any further evaluation necessary. For example, if a datum is desired, and the immediate valve is a datum.loc, obtain the valve from the location. This evaluation procedure is defined more completely in C.

The atomic expressions are identifiers or integers. The immediate valve of an identifier is the string of characters composing the identifier. The immediate valve of an integer is a 60 Bit datum read in octal or decimal depending on the current input mode and whether the integer has a trailing 'B' or 'D' . A trailing 'B' causes the integer to be read in octal. A trailing 'D' causes the integer to be read in decimal. With no trailing 'B' or 'D' it is read in the current input mode.

We now list the compound expressions. For each one we specify the content of each subexpression, and the immediate valve of the whole expression. We also say how this immediate valve is computed. For some compound expressions with exactly one subexpression, the immediate valve is that of the subexpression; we indicate that with the word 'identity'.

Each compound expression is given as a production in a content free grammar. The terminal symbols are:

\, #, $, +, -, *, /, (,),: and, which stand for themselves; and a number of key words which stand for themselves

The nonterminals appear at the left of the productions below, so no more need be said except that Loc.exp is the name of the whole class of expressions.

1.1) LOC.EXP ::=LOC.TERM

(Identity)

1.2) LOC.EXP ::=LOC.EXP # INTEGER.EXP

Contexts: LOC.EXP Object
INTEGER.EXP Datum

Result: Indexed.Obj

Typical examples are an indexed slot within a clist, or an address within a file.

1.3) LOC.EXP ::= #INTEGER.EXP

Contexts: INTEGER.EXP Datum

Result: Subprocess index

Used for reference to core or clist of a subprocess that has called the Bead ghost. Not available except in the Bead ghost.

1.4) LOC.EXP ::= $ REG # INTEGER.EXP

Contexts: INTEGER.EXP Datum

Result: Register index

Available only in Bead ghost. Used for referring to the registers of a subprocess that has called the Bead ghost.

2.1) LOC.TERM ::=LOC.PRIM

(Identity)

2.2) LOC.TERM ::=LOC.PRIM > IDENTIFIER

Contexts: LOC.PRIM Object
IDENTIFIER Identifies

Result: Scan list loc

Used for making a scan list reference.

2.3) LOC.TERM ::= : IDENTIFIER

Contexts: IDENTIFIER Identifies

Result: Scan list loc

The scan list used is the standard scan list currently associated with the process.

2.4) LOC.TERM ::=LOC.PRIM : IDENTIFIER ; LOC.PRIM

Contexts: $LOC.PRIM_1$ Object
IDENTIFIER Identifier
$LOC.PRIM_2$ Object

Result: Directory loc

A directory loc is formed, using 1st object as a directory, the identifier as the name, and 2nd object as an access key.

2.5) LOC.TERM ::= LOC.PRIM : IDENTIFIER ;

Contexts: LOC.PRIM Object
IDENTIFIER Identifier

Result: Directory loc

A directory loc is formed as in 2.4 using the null access key as access key.

3.1) LOC.PRIM ::=WORD.EXP

(Identity)

4.1) WORD.EXP ::=INTEGER.EXP

(Identity)

4.2) WORD.EXP ::=WORD.PART

Contexts: WORD.PART Word.part
Result: Datum

Ignores the bit count field of the word.part, and takes as valve the datum part.

4.3) WORD.EXP ::=WORD.EXP, WORD.PART

Contexts: WORD.EXP Datum
WORD.PART Word.part

Result: Datum

The 1st datum is shifted left (end off) by the amount of the bit count in the word.part. The result is then or'd with the datum part of the word.part.

5.1) WORD.PART ::=INTEGER.EXP \ INTEGER.EXP

Contexts: INTEGER.EXP$_1$ Datum
INTEGER.EXP$_2$ Datum

Result: A word.part

The resulting word.part consists of a bit count and a datum part. The bit count is the first datum. The datum part is the second datum.

(Note: for 6.1 to 8.3 The contexts and result are all datum)

6.1) INTEGER.EXP ::=INTEGER.TERM

(Identity)

6.2) INTEGER.EXP ::=INTEGER.EXP + INTEGER.TERM

60 Bit integer addition

6.3) INTEGER.EXP ::=INTEGER.EXP - INTEGER.TERM

60 Bit integer subtraction

6.4) INTEGER.EXP ::= + INTEGER.TERM

(Identity)

6.5) INTEGER.EXP ::= - INTEGER.TERM

60 Bit 1's complement. Except for 0, same as 0- datum. A -0 will produce all bits on.

7.1) INTEGER.TERM ::=INTEGER.PRIM

(Identity)

7.2) INTEGER.TERM ::=INTEGER.TERM * INTEGER.PRIM

(not yet implemented, probably 48 bit multiplication)

7.3) INTEGER.TERM ::=INTEGER.TERM / INTEGER.PRIM

(not yet implemented, probably 48 bit division)

8.1) INTEGER.PRIM ::=IDENTIFIER

~~The identifier itself is the result~~ (Identity)

~~The value of the identifier is the result.~~

8.2) INTEGER.PRIM ::=INTEGER

The value of the integer is the result

8.3) INTEGER.PRIM ::= (LOC.EXP)

(Identity)

8.4) Integer.prim ::= ↑ identifier

If the identifier has a value as ~~the~~ entered a variable, that value is the result.

C) Evaluation Procedures

If the type of value in hand does not match the desired type and the desired type is not object or datum, then the evaluation fails. If the type of valve in hand does match the desired type, then no computation is necessary. Otherwise the following procedures are used.

If the type of valve in hand is:

1) Identifier

a) And datum is desired
If a variable with the identifier as name exists and contains a datum, that is the valve.

b) And object is desired
~~If a variable with the identifier as name exists, and contains an object, that is the valve.~~
The identifier is looked up in the ~~current~~ ~~standard~~ ~~scan list~~ standard scan list currently associated with the process

2) Scan list loc

And an object is desired
The given name is looked up in the given scan list, the resulting object is returned as valve.

3) Directory loc

And an object is desired
The given name is looked up in the given directory with the given access key, the resulting object is returned as valve.

4) Indexed object

a) And datum is desired
The given object is assumed to be a file, ECS or disk, and is read with the given datum as address

b) And object is desired
The given object is assumed to be a clist, and an object is fetched with the given datum as index.

5) Indexed subprocess (available only during Bead ghost)

a) And datum is desired
A word is read from the full path of the subprocess, word address 0 refers to word 0 of the subprocess calling the Bead ghost. Negative addresses can be used.

(See variable FULLM later)

b) And object is desired

Or capability is obtained from the full path of the subprocess. Index 0 refers to index 0 of the subprocess calling the Bead ghost. Negative addresses can be used. (See variable FULLC later)

6) Register loc (available only during Bead ghost)

The given datum is used to reference the XJ package of the subprocess calling the Bead ghost. $A4_0$ & B_4 are in word index 4; A7 & B7 in word index 7; X4 in word index 14B and X7 in word index 17B.

D) Some Present Variables

1) FULLM

Contains a datum. When added to a number appearing in an indexed subprocess, permits that number to address core relative to the first word above the field length of the Bead ghost, i.e. #FULLM addresses word 0 of the lst subprocess that is a descendant of the Bead ghost, in the current full path.

2) FULLC

Contains a datum. Action is similar to FULLM except that it is used with the clist of the subprocess.

3) BEADF (temporary, for test only)

The command GETBDFILE followed by 2 identifiers separated by blanks obtains an object from old Bead upon which the test system is running. That object is placed in this variable.

4) ROOTD (test only)

Contains the root directory of the disk system. Very dangerous.

E) MISC

1) The standard scan list consists of

0) Your temporary directo (ownership capability)
1) Null access key
2) The S directory (not an ownership capability)
3) Public access key

2) Your temporary directory initially contains:

Order S: The S directory, non ownership capability
Order TEMPDIR: The temporary directory itself

F) Examples (1st set under services or Bead ghost)

1) PLAP :A

The name A is looked up in the standard scan list. In the current version that means A is first looked up in your temporary directory, if that fails then in S. (A will not be found in S.) If there is an A in your temporary directory, the capability for A will be printed.

2) PLAP [illegible] TEMPDIR:A; TEMPDIR:A

TDLIST is looked up in the standard scan list. Initially it would be found in your temporary directory with value of your temporary directory. Then A will be looked up in the result (your temporary directory) using the null access key.

3) PDATA :A #6 5 A#6 5

A is looked up in the standard scan list, the result is assumed to be a file. Words 6, 7, 10, 11, 12 are printed.

4) PDATA #7 5

Words 7, 10, 11, 12, 13 from your subprocess core are printed.

5) PDATA #FULLM+7 5

Words 7, 10, 11, 12, 13 from the core of the subprocess which is the 1st descendant of the Bead ghost in the current full path are printed.

6) MDATA 6 #7

A 6 is placed in your subprocess core address 7

7) MCAP :A #5

A is looked up in the standard scan list. The resulting capability is placed at index 5 in your subprocess C-list.

Examples under command processor:

1) :EDITOR INPUT

EDITOR is looked up in the standard scan list. It will probably be found under S. The result is assumed to be a file containing a subprocess descriptor. The subprocess is constructed. It is called with displ code representation of input as its 1st parameter.

2) (:TEMPDIR):EDITOR INPUT

Assuming TEMPDIR still names your temporary directory, EDITOR is looked up in your temporary directory. The result is treated as above.

More complicated examples of parameters:

12D\6, 18D\10

Produces following datum (inoctal) (assuming octal input)

6000010