Bead 4.1 stuff
later (4.2?)
errors $\longrightarrow$ replaced with "return with interrupt" !

November 24, 1970

Actions In the Event of A Panic

I Panics cause an interrupt to be directed at the bead ghost in the users process. There are 2 kinds of panic, major and minor. A major panic is generated by holding down the break button. It requires about 2 seconds. This 2 seconds is cumulative, that is, it can be obtained from several periods of at least 1/10 second each. A minor panic is currently generated by control shift P. Eventually, the particular character causing minor panic will be placed under program control.

II After a panic has been detected by the bead ghost, the following algorithm is followed:

A. If the subprocess immediately below the bead ghost in the stack (former running subprocess) is not a system subprocess, then the action depends on whether major or minor panic.

1. Major panic
   A debugger is called. (Currently this is the bead itself, and will type out "panic bead" here.)
2. Minor panic
   An error return to the user subprocess is made with error class = e. panic = 12_{10} error number = 0
   (if this error is detected by the bead ghost, a debugger is called as in 1 - see above.)

B. If the subprocess immediately below the bead ghost in the stack is a system subprocess, then the bead ghost returns to it with error.

   error class = e. panic
   error number = 0 if minor panic, 1 if major panic

The system subprocess then cleans itself. Then a choice is made depending on whether the system subprocess is a controlling subprocess or not, (see below for definition) and whether this was a major or minor panic.

1. If it is not a controlling subprocess, and this was a minor panic, then return with error is made.
   error class = e. panic
   error number = 0
2. If it is not a controlling subprocess, and this was a major panic, then the bead ghost is jump called and the bead ghost proceeds as in II - A.

3. If it is a controlling subprocess, it requests a new command from the user.

III The effect of the above algorithm is to clean up and remove from the call stack all non-controlling system subprocesses until either a controlling system subprocess or a user subprocess is reached. If a controlling system subprocess is reached, it regains control, usually by requesting a command from the user. If a user subprocess is reached then what happens depends on whether this was major or minor panic. If major, a debugger is called immediately; if minor, an error is generated for the user.

IV The controlling subprocesses include:

A. The command processor

B. The debugger

C. (Possibly) disk, S

(Note: Disk, S will initially be a user subprocess. Don't panic out of disk!! But, if time permits, I will fix it to respond properly to interrupts.)

V What is described above will not work completely until bead vers. 4.2 (I hope)

Under bead vers. 4.1 funny things happen if the bead was running at the time of the interrupt. I will not attempt to catalog them here, but will give one example. If the bead was waiting for a command from the user, and you interrupt then it types out "panic bead here." If you subsequently restore then it makes no response, but it in fact goes back to waiting for a command from the user.