Identification

Interactive Debugging Aids for Initial Multics
M. Wantman

Discussion

MSPM section BX.10.00 describes an elaborate system of
debugging aids for Multics. While that section describes
the system envisioned for the ultimate versions of Multics,
it was thought desirable to have a simplified debugging
system which could be used with Initial Multics. This
interim version is expandable and eventually should include
all the features described in BX.10.00.

Facilities

The interim debugging system provides ways for a user
to obtain the following types of information:

a)   machine conditions and register contents

b)   segment names and numbers and access information

c)   selective dumps in octal of a segment between
     user-specified limits.

d)   dump contents of an entire process directory

e)   forward or backward stack traces

f)   argument lists.

g)   list of available requests

In addition, the user may make changes which affect the
subsequent execution of the process. These are

h)   make a segment known or unknown to the process

i)   make an octal patch to a segment

No provision has been made for symbolic addressing of
locations within segments, or for changing the contents
of registers or segment locations. These, as well as
other features described in BX.10.00, will be implemented
in stages, and BX.10.00A will be updated periodically.

Usage

Entry to the debugging system is accomplished through
the procedure "probe". Requests for specific information
are then given to "probe", which passes them on to the
appropriate procedures. "probe" is invoked in one of
two ways:

    1)   from command level. This can be done after
a process incurs an unexpected signal and "unclaimed_signal"
has called the listener. It can also be done after the
user quits a process that has been running and the quit
responder is invoked.

    2)   from a running process. This occurs if a call to probe
is encountered in a procedure segment. Once the call to probe
has been executed, the user can "snoop" around at his leisure.
Execution of the original procedure can be continued by simply
leaving probe via the "quit" request.

Once probe has been entered, it handles all user input
lines until the "quit" request is given.

The following requests are recognized:

arglist	print argument list for one stack frame
dump_process	print entire contents of a process directory
info	print a list of requests available
initiate	make a segment known to the process
output	change routing of output
quit	leave probe
segdump	dump segment in octal
seginfo	list segment status information
set	place information in write-permit segment
stack	print stack trace
state	obtain machine conditions
terminate	make a segment unknown

arglist

The format of this request is

        arglist s1 s2

It prints the values of arguments which were passed to
the stack frame designated by s2 in the segment s1.   s1
is the name or number of the stack segment, and s2 is
either the location in octal of the beginning of the stack
frame, or the name of a segment. If it is a segment name,
the stack is searched backwards for the latest occurrence
of a frame used by the segment. If no appropriate frame
is found, a diagnostic message is printed.

If only one argument is given to arglist, the argument is
interpreted as a location within the current stack segment.
That is,

        arglist seg_name

is interpreted as

        arglist stack_xx seg_name

dump_process

Request format is

        dump_process   -id-

where id is the unique id of the process expressed in
octal or as a character string. If it is not present,
the user's process is assumed. The process directory
is scanned and each segment is dumped in octal. Normally
output will be directed to a segment for later printing.

info

Execution of this request will cause printing of a short
description of the requests accepted by probe. The message
will refer the user to this document for more complete
information.

initiate

Makes a segment known to the process and assigns a segment
number. The format of the request is

        initiate path -callname-

The segment found at path in the file system is initiated
with call name callname. If callname is not specified,
the entry name of the segment will be used. If the segment
is successfully made known or is already known, the comment

segment (path) initiated with call name (callname). Number (n)

is printed. If it could not be initiated, the comment

            segment (path) not initiated

is printed.

output

All output from probe is directed initially to the user's
console. It may be desired to have output go to a segment
and have the segment printed off-line. The request

            output segment s

will direct output to the segment s. If no segment by
that name exists, one will be created with access RWA.
The comment

            output directed to segment s. Number (n)

will be printed on the console. If the segment is filled to the
64K limit, its bit count is set and another uniquely-named
segment is started.

Output will be redirected back to the console by the request

            output console

The segment that was being written will have its bit count
set. If output is directed to the segment again, the
later information will be appended to the end of the segment.

quit

When this request is given probe returns to command level.
If output had been directed to a segment, the bit count
on the branch is set.

segdump

This request is used to dump all or part of a segment
in octal. The segment must be known to the process, and
may be specified by name or number. The format of the
request is

            segdump s -lbound- -hbound-

s is the name or number of the segment to be dumped, and
the lower and upper bounds are given by lbound and hbound
respectively. If the lower and upper bounds are not specified,
the entire segment is dumped. If the upper bound is not
specified the segment is dumped from n1 to the current
length as defined by the file system.

If the segment is not known to the process, the comment

                segment s not yet initiated

is printed. The segment may be made known by the initiate
request. If the user does not have read access, the information
is copied via ring-0-peek into the users stack and printed.

seginfo

This request prints a list of names and numbers of segments
known to the process. The format is

                seginfo s1 s2 -all- -long-

s1 and s2 are segment names or numbers in octal, and indicate
the range of segment numbers for which the information
should be printed. For example, the request

                seginfo 200 test_proc

will print the names and numbers of all segments whose
numbers lie between 200 and the segment number of test_proc.
If neither s1 nor s2 is specified, the information is
printed for all segments. If only s1 is specified, the
information is printed for s1 only. If the lower bound
is higher than the upper bound, a comment is printed.

The presence of the optional parameter "all" in the request
will cause all the call names by which a segment is known
to be printed. The presence of the parameter "long" will
cause printing of the current length of each segment,
its access attributes, and its date of creation.

set

This request allows the user to make changes to any segment
in his process for which he has write permission. The format
of the request is

        set s|n v1 -v2- -v3- ...

S is the name or number of the segment to be altered, and n
is the location where the first new value is to go. Blanks
may or may not be present on either side of the vertical bar.
The status of segment s is examined to determine if the user
has write access. If he does not, a message is printed and
no patching is attempted.

Otherwise the values represented by v1(v2 v3 ...) are placed
in locations n(n+1,n+2,...) of segment s. A message is printed
containing both the old and new values to minimize the probability
of error and to facilitate restoration of the original
values.

stack

Multics keeps a partial history of the course taken by
a process. The "stack" request makes some of that
information available. It prints the name and number
of the procedure using the frame, the starting location
of the frame, and its size. The format of the request
is

        stack -s1- -s2- -f- -args-

where all parameters are optional.

s1 is the stack segment to be examined, s2 is the location
in the stack where tracing is to begin, "f" indicates
that the trace is to proceed forward in the stack, and
"args" asks for an argument list (as in the "arglist"
request) to be printed for each stack frame.

s1 may be specified by name or number. If it is not given,
the current stack segment is traced from the end to the
beginning. s2 may be specified only if s1 is given.
If s2 is given as an octal number, it is interpreted as
the starting location of a stack frame. If s2 is the
name of a segment, the stack is examined from the end
for a frame belonging to s2. If one is found, tracing
begins at the frame. If none is found, a message is
printed for the user.

The forward option "f" can be specified only if s1 and
s2 are given. If it is present, the trace proceeds from
s2 to the current end of the stack. The option "args"
can occur anywhere in the request. If it is present,
"stack" will print a list of all arguments passed to each
stack frame.

state

The "state" request prints machine conditions as requested by
the user. It searches backward through the current stack
trying to find an occurrence of a frame belonging to "signal".
If one is found, and it is preceded immediately by an
occurrence of the FIM (fault interrupt module), then the
FIM frame contains the machine condtions at the instant
the fault occurred.

If state receives a number as an argument, it prints the
machine conditions existing at that location in the current
stack segment. If no signal-FIM combination can be found,
register contents are extracted from the stack frame of
the procedure that called probe.

The following parameters are recognized by state:

arith   print A, Q, and exponent registers

timer   print timer register

location   print location of fault and the
                         effective address

index   print 8 index registers

bases   print 8 base registers

cunit   print control unit and ring number

If no parameters are given, all the above information is printed.

terminate

This request makes a segment unknown by removing it from
the KST (known segment table) . The format is

              terminate s1

S1 is the pathname of the segment to be terminated. If
no directory is specified the current working directory
is assumed.

(Note: When a segment is made unknown, its linkage section
is not removed from the combined linkage section. If
a different version of the segment is later made known,
you will continue to work with the older linkage section.
This can lead to unpredictable results).

Summary of requests

arglist	-s1- s2
dump_process	id
info	
initiate	path -callname-
output	medium -name-
quit	
segdump	s1 -lbound- -hbound-
seginfo	-s1- -s2- -all- -long-
set	s1|n v1 -v2- -v3-
stack	-stackseg- -s1- -f- -args-
state	-loc-
terminate	callname

Implementation

Entry to the set of debugging commands is achieved by
calling the procedure probe with no arguments. Probe
then calls dispatch_request with no arguments for each
new request line. After completion of each request probe
prints the line.

-

to indicate that the user should enter his next request.

The procedure dispatch_request calls read_in to get the
request line and hands it to the debugging parser via
the parse_scan$prime entry. It then calls parse_scan$atom
to get the first group of characters. This should be
the name of one of the basic requests. A table of request
names and corresponding procedures to call is kept in
the eplbsa segment debug_data. Dispatch_request searches
this table and calls debug_data with the index to the
request table, which makes the call to the appropriate
procedure.

Each request makes calls to parse_scan to get its arguments
and interprets them appropriately. If a request finds
that it cannot do what is asked of it, it prints an error
comment and returns to dispatch_request which returns
to probe which waits for the next request.

If the search made by dispatch_request fails, the request
is passed directly to the shell, which treats it as a
normal command line. Thus any Multics command can be
given without leaving probe.

For details of the implementation of the individual requests,
see MSPM sections BX.10.05-BX.10.20. For abstracts, see
MSPM section BS.