1024

Bell Laboratories

subject: snapshot() - A Callable Core snapshot Function (or Subroutine) for Debugging Programs

date: 19 November, 1974

from: S. T. Schuyler

PROGRAMMER'S NOTES

In order to facilitate program debugging and testing, a function named "snapshot" was developed. The snapshot function is designed to output the contents of specified core locations (virtual address space), either by word or by byte, in any number base from 2 to 20, optionally outputting the contents of the general registers and stack, to any output destination, and with any linewidth between 40 and 130 characters per line. This routine may be called from either assembler or C language source programs. The use of snapshot() is described below.

Use of snapshot() from C source programs:

The snapshot function will accept up to eight optional arguments. Each argument position is well defined and must be used in the order shown. The first example illustrates using snapshot with all arguments. Less than all of the arguments may be used, and these situations will also be illustrated.

All Argument Call:

snapshot(from,to,regs,message,base,destination,linewidth,mode):

Illustration 1

Definition of arguments:

Argument Number:	Use:
1. from	is a virtual address where you desire the snapshot function to begin outputting core contents. If the value of "from" (or the first argument) exceeds your allocated virtual address space, no snapshot may occur. If snapshot is called with no arguments, it is defaulted to zero. Note: There is a gap between your program virtual space and your stack virtual space, attempts to access this space are not possible and cause snapshot to terminate.

Argument Number:	Use:
2. to	is a virtual address where you desire the snapshot function to stop outputting the contents of core. If the value of "to" (second argument) exceeds your virtual address space, snapshot will terminate at the last address in your virtual space. In addition if the value of "to" is less than the value of "from", then snapshot will reverse the arguments. If no value is passed, "to" is defaulted to zero.
3. regs	the value of regs is an integer between 0 and 48, and is used to control the outputting of the contents of the general registers and stack. regs < = 0 : no register or stack contents. regs = 1 : register contents only (default). 2 < = regs < = 48 : register and stack contents, stack to a depth of "regs". Note: the bottom of the stack is virtual address 177776. If a stack depth is requested which exceeds this address, then snapshot stops at this address. This is not considered to be an error.
4. message	"message" (argument four) is a virtual address or pointer to a null terminated string of characters. If the value of "message" is -1 or 0, it is ignored. Default is zero, or no message.
5. base	the number base to be used for interpreting the contents of the desired core addresses, registers and stack. 2 < = base < = 20; default is base = 8.
6. destination	argument six is a file descriptor obtained from an open() function call. It is assumed to be a number between 1 and 10. Any number which is outside this range causes snapshot to default to the standard output. Destination = 1 is standard output.
7. linewidth	is an integer which specifies the width of a printline on the destination device or file. Any value between 40 and 130 is legal, default is 80.

Argument Number:	Use:
8. mode	an integer value, either 0 or 1, used to control whether snapshot outputs core contents by word (mode = 0), or by byte (mode = 1, default).

In summary the arguments are:

virtual (address	virtual ,address	,integer	virtual ,address	,integer	,integer	,integer	,integer);
(from	,to	,regs	,message	,base	,destntn	,linewth	,mode);
0 to ?	,0 to ?	,0 1 2 - 48	,"....."	,2 to 20	,1 to 10 fildes- from open(),	,40 to ,130 ,(80 dflt)	,0 or 1

Any or all arguments may be left out. All arguments defaulted yield a snapshot of the register contents in base 8 on the standard output. The only restriction on leaving out arguments are as follows:

If argument number i, 2 < = i < =8, is to be used then a value must be assigned to args j, 1 < = j <=(i-1).

That is, if you are using arguments numbered 2 and 5, and not 1,3,4,6,7,8, then the call in illustration 2 (below) would be proper.

Less than all argument call:

```
snapshot(0,to,0,0,base);
```

or if using only arg. 5

```
snapshot(0,0,0,0,base);
```

Illustration 2

In other words, a minimum of a word of zeros must be passed for non-used arguments preceding any used arguments in a given snapshot call.

All arguments are passed to snapshot on the stack as values.
Those not used, or erroneously used, are defaulted.

Assembler Linkage

```
/ In your assembler routine, move those arguments on to the stack
/ that you will be using, inserting words of zeros for those not
/ used as per the above rules.

        mov arg1,-(sp)
        mov arg2,-(sp)
             .
             .
             .
        mov arg_1,-(sp)

/
/ then call snapshot using the pc
/
        jsr pc,←snapshot
/
/ immediately following the call to snapshot, insert an add
/ instruction which adds the passed number of arguments to
/ the sp, effectively popping the stack.
/
       add $???,sp
/
/ note: "???" is an integer constant.
/
/ Example:
/
        mov $0,-(sp)        /from
        mov $64,-(sp)       /to
        mov $8,-(sp)        /regs: r & s; depth = 8.
        mov $0,-(sp)        /no message
        mov $2,-(sp)        /base
/
/ total of five arguments
/
        jsr pc,←snapshot
        add $5,sp           /pop stack
/
/ the add instruction must be there, a routine in snapshot
/ will be called, nargs(), which looks for the add instruction.
/
```

A copy of snapshot output is attached.

```
#
main()
{ extern snapshot();
  extern char *snaptitle[];

  int *f,*t,r,b;

  f = snaptitle[0];
  t = f + 30;
  r = 8;

      snapshot(t,f,r,"this is by word",8,1,80);

}
```

```
*** BEGIN SNAPSHOT OUTPUT ***
from address: 000134
  to address: 000230
REGISTERS:              STACK CONTENTS
R0: 000010                    177756
R1: 000001                    000124
R2: 000000                    000230
R3: 000000                    000134
R4: 000124                    000010
R5: 177756                    010252
SP: 177716                    000010
PC: 000124                    000001
this is by word

000134  006015  025052  020052  042502  044507  020116  ..*** BEGIN
000150  047123  050101  044123  052117  047440  052125  SNAPSHOT OUT
000164  052520  020124  025052  020052  000000  005015  PUT *** ....
000200  071146  066557  060440  062114  062562  071563  from address
000214  020072  000000  005015  020040  067564  060440  : ....  to a
000230  062144  062562  071563  020072  000000  005015  ddress: ....
*** END SNAPSHOT OUTPUT ***
```

MH-8624-STS-jd

S. T. Schuyler