NAME

sdb − SNOBOL4 debugger

SYNOPSIS

sdb [*options* . . .] *program.sno*

DESCRIPTION

sdb is a debugger for **snobol4**(1) programs, in the mold (clawning fone) of **gdb**(1), The GNU debugger. **sdb**(1) uses **readline**(3) for command line editing/history when available.

Commands:

break *LABEL_OR_STATEMENT_NUMBER*
Set a breakpoint.

bt
Display call stack backtrace.

commands *BREAKPOINT_NUMBER*
Add sdb commands to execute (ie; print & continue) to a breakpoint.

condition *BREAKPOINT_NUMBER EXPR*
If *EXPR* is supplied, it is used as a predicate to make the breakpoint conditional.

continue *EMPTY_OR_COUNT*
Continue from breakpoint. The optional count specifies how many times to continue past this breakpoint (sets "ignore" count).

delete *BREAKPOINT_NUMBER_OR_EMPTY*
Delete a single breakpoint, or all breakpoints.

disable *BREAKPOINT_NUMBER_OR_EMPTY*
Temporarily disable a breakpoint, or all breakpoints.

enable *BREAKPOINT_NUMBER_OR_EMPTY*
Reenable a breakpoint, or all breakpoints.

finish
Resume debugging after current function returns. Will display function return type and value, if any.

help
Display help.

ignore *BREAKPOINT_NUMBER COUNT*
Set breakpoint ignore count.

info
Display list of breakpoints and their status.

list *EMPTY_OR_STATEMENT_NUMBER*
Display source code.

next *EMPTY_OR_COUNT*
Single step execution, skipping over function calls.

print *EXPRESSION*
Evaluate expression and print result. Can be used to call functions, or set variables.

quit
Exit debugger.

step *EMPTY_OR_COUNT*
Single step.

watch *VARIABLE*
Set watchpoint on a variable (break when value changes).

what
Display the datatype of variable contents (or expression).

where
An alias for bt.

A blank line repeats the previous command.

Non-ambiguous abbreviations of commands can be used (ie; "s", "n").

The GNU Readline library (when available) will be used for sdb input for command editing and history.

The keyboard interrupt character (eg; Control C) will stop a running program and return control to the **sdb**(1) command prompt.

If your program calls the **SDB()** function, it will act as a breakpoint. You can check whether **sdb**(1) is loaded with the **FUNCTION** predicate, ie;

```
FUNCTION('SDB') SDB()
```

SEE ALSO

snobol4(1), **gdb**(1), **readline**(3), **snobol4readline**(3)

AUTHOR

Philip L. Budne

Inspired by Fred Weigel's DDT.SNO and **SITBOL**'s SNODDT.

NOTA BENE

sdb(1) uses a wide variety of system facilities and will interact poorly with any programs that use any number of features, including:

- Altering listing settings with directive/control lines.
- Altering **&STLIMIT**, **&ERRLIMIT**, or **&TRACE**.
- Calling **SETEXIT()**

BUGS

If you try to put a breakpoint on a label or line with no code or goto fields, the breakpoint will never be triggered.

You cannot put a breakpoint on the **END** label (however control always returns to sdb when the **END** label is reached).

There is no "run" command; you cannot restart the program without quitting and losing breakpoint settings.

Interrupt character trapping is in it's infancy, and only occurs at the start of a each statement executed.

The interrupt character is silently ignored when at the **sdb** command prompt.

sdb does not read an init file (ie; .sdbinit).