NAME

dbx - source-level debugger

SYNOPSIS

dbx [-c <u>cmd</u>] [-C] [-e] [-I<u>dir</u>] [-kbd]
[-q] [-r] [-s <u>startup</u>]
[-sr <u>startup</u>] [<u>execfile</u> [<u>corefile</u> | <u>process</u>-<u>id</u>]]

DESCRIPTION

dbx is a utility for source-level debugging and execution of programs written in C++, ANSI C, FORTRAN, and Pascal. dbx accepts the same commands as debugger(1), but uses a standard terminal (tty) interface. For a full overview of dbx and Debugger features see the manual, <u>Debugging</u> <u>a</u> <u>Program</u>.

<u>execfile</u> is an executable file, produced by a compiler, with the -g option to include a symbol table. This symbol table contains the names of all the source files used to create the <u>execfile</u> (all of which may be browsed), as well as a wide range of debugging information. Debugging support is limited for any parts of a program not compiled with the -g option.

(Solaris 2.x)
Initially, symbol table information is stored piecemeal in each of the object files the compiler creates. (If compilation does not create an object file, all debugging information is stored in the <u>execfile</u>.) Distributing the debugging information in the object files allows dbx to read and load debugging information as it is needed. This feature is called Auto-Read. If you need to move object files from their original location, make sure that dbx knows where to find them. (See the use and pathmap commands.) If it is not feasible to keep program .o files around, you can disable Auto-Read by compiling using the - xs option, which instructs the compiler to have the linker place all debugging information in the program executable.

If, when starting dbx, no <u>execfile</u> is specified, use the debug command to specify a program to be debugged.

(Solaris 2.x)
If you know a <u>process</u>-<u>id</u> but not the <u>execfile</u>, you can use the - (dash) as the <u>execfile</u> and give the <u>process</u>-<u>id</u> option to attach the process to dbx.

If a <u>corefile</u> argument is specified in the command line, you can use dbx to examine the state of the program when the core file was produced.

During startup, dbx searches for .dbxrc first (ksh mode). If .dbxrc is not found, dbx prints a warning message and

searches for .dbxinit (dbx mode). The search order is: ./.dbxrc, then ~/.dbxrc.

(SPARC)
Runtime Checking (RTC) is a fully integrated feature of dbx; it uses the full capabilites of dbx including setting breakpoints and examining variables. With RTC, you can automatically detect runtime errors in an application during the development phase. Compiling with the - g flag provides source line number correlation in the error messages. RTC can check programs compiled with the optimization - O flag. You do not have to recompile, relink, or modify the makefile to use RTC.

For proper operation, RTC requires dynamic linking with libc, use of the standard libc functions malloc/free/realloc, and does not support attachment to a running process.

To use RTC, start dbx with the -C option or set the environment variable SW_RTC or specify the -C option each time you start dbx.

OPTIONS

-c <u>cmd</u>
Execute <u>cmd</u> after initialization. For more than one <u>cmd</u>, use double quotes around the string of commands, separating each with a semi-colon.

-C
Enables Runtime Checking to be activated, but does not turn on checking.

-e
Echo all input commands to the standard output.

-I <u>dir</u>
Add <u>dir</u> to the list of directories in which to search for a source file. dbx normally searches the current directory and the directory where <u>execfile</u> is located. The directory search path can be reset with the use command.

-kbd
Debug a program that sets the keyboard into up-down translation mode. This flag is necessary if a program uses up-down decoding.

-q
Silence echoing of two loading messages: "Reading symbol table for..." and "Attached to ...". (This option is intended mainly for use by the auto-traceback facility.)

-r
Execute <u>execfile</u> immediately. Parameters follow the <u>execfile</u> name (redirection is handled properly). If the program terminates successfully, dbx exits. Otherwise, dbx reports the

reason for termination and waits for a response. dbx reads from the terminal (/dev/tty) when -r is specified and standard input is a file or pipe.

-s startup Read initialization commands from the file named startup instead of from .dbxrc.

-sr startup Read initialization commands from the temporary file named startup instead of from .dbxrc and then remove startup.

The basic commands to know are: run, to run the program being debugged; where, to obtain a stack trace with line numbers; print, to display variables; and stop, to set breakpoints.

Expressions

dbx expressions are combinations of variables, constants, procedure calls, and operators. Variables are either variables in the program being debugged or special dbx variables whose names begin with $. Hexadecimal constants must be preceded by a ‘0x’ and octal constants by a ‘0’. Character constants must be enclosed in single quotes. In general, the expression rules followed are those of the current language.

Operators

+ - * / %	Add, subtract, multiply, divide, and remainder.
<< >> & | ~	Left-shift, right-shift, bitwise AND, bitwise OR, and bitwise complement.
& *	Address of operator and contents of operator.
< > <= >= == != !	Less than, greater than, less than or equal to, greater than or equal to, equal, not equal, and negation.
&& ||	Logical AND and logical OR
sizeof (cast)	Size of variable or type cast.
. ->	Field reference and pointer field reference.
::	C++ scope resolution operator.

Scope Rules

dbx resolves scope conflicts based on the values of the current file and function. These values are updated as files and functions are entered and exited during execution. You can also change them explicitly by using the file and func commands. When the current function is changed, the current file is updated along with it, and vice versa.

C++ Symbolic Names

CC mangles the original source names of locals, member functions, functions, and so on. dbx will demangle all those mangled names into C++ symbolic format in dbx output.

(Solaris 2.x)

Thread Identification

In some commands the use of id refers to the thread id (tid) or light weight process id (lid). These take the form of t@N or l@N.

Handler Identification

Event handlers are identified with an integer number hid (see status, delete, and handler commands).

COMMANDS

!!	Execute previous command.
! num	Execute previous command number num. (see history command)
! - num	Execute a previous command from the current command-number minus num.
! str	Execute most recent previous command that started with str.
:s/l/r/	Substitute r for l in the previous command and execute it.
:p	Print (but do not execute) the previous command.
^l^r^	Substitute r for l in the previous command and execute it.
. file	This command causes dbx to read commands from file. Any variable assignments and function definitions affect the current environment. $PATH is searched if the path option is set. See also source.

```
:          Does nothing and returns 0 (true).

[expr]     This command is an alternative  syntax  for  "test
           expr". See test for more information.

alias      alias is normally an alias for  either  dalias  or
           kalias. Type "alias alias" to find out which, then
           see either help kalias or help dalias as appropri-
           ate.  The  default  is  kalias  unless dbx found a
           .dbxinit file and no .dbxrc file,  in  which  case
           the default is dalias.

assign  var = exp
           Assign the value of  the  expression  exp  to  the
           variable var

bg         Put the current job into the background.

bg   %+    Put the current job into the background.

bg  %-     Put the previous job into the background.

bg  %n     Put job number n into the background.

bg  %str   Put the job which begins with str into  the  back-
           ground.

bg  %?str  Put the job which  contains  str  into  the  back-
           ground. The % is optional in all cases.

bind       Displays the current emacs editing key bindings.

bind  key=def
           Bind key to definition def.

bind  key  Display the current definition for key.

bind key=  Remove binding of key.

bind  -m [key]=def
           Define key to be a macro with definition def.

bind -m    Same as bind.

break      Break out of enclosing loop ( while, until, or for
           ).

break   n  Break out of n enclosing loops.

bsearch  string
           Search backward  for  the  string  string  in  the
           current file.
```



```
bsearch    Repeat the previous search.

builtin    cmd [args]
           Execute the cmd , suppressing  the  normal  search
           for a function or alias named cmd. If cmd is not a
           builtin, it displays an error message.

button  seltype cmd
           Define a software button (debugger)  where  selec-
           tion type, seltype , is:

           ignore - Ignore the selection

           literal - Take the selection literally

           expand - Expand  selection  to  longest  syntactic
           identifier

           command -Expand to  a  command  from  the  command
           subwindow

           lineno - Line number from the source subwindow for
           the current selection

call   proc ([params])
           Call the procedure

cancel     Cancel the current signal. (Useful within a WHEN.)

catch      Print a list of the caught signals.

catch  num num...
           Catch signal(s) numbered num.

catch  sig sig...
           Catch signal(s) named by sig.

_cb_aoutchange  old a.out name new a.out name
           This user-defined ksh  function  (if  defined)  is
           invoked by dbx whenever the program being debugged
           changes. The definition of change in this  context
           is that the full path name is different. This is a
           good place to set a.out specific breakpoints.

           For example:
             _cb_aoutchange()
             {
             case "$2" in
             *producer)      stop in make_some_more ;;
             *consumer)      stop in get_another_one ;;
             esac
             }
```

_cb_prompt
This user-defined ksh function (if defined) is invoked by dbx immediately before it prints the top level prompt (PS1). This is a good place to set PS1 so that it reflects the current state of the process. _cb_prompt is called with no arguments.

cd Change to the user's $HOME directory.

cd <u>dir</u> Change the current directory to <u>dir</u>.

cd - Change to the previous directory ($OLDPWD).

cd <u>old</u> <u>new</u>
Starting with the current directory, replace the string <u>old</u> with <u>new</u> and change to the result.

(begin SPARC-specific commands)

check
Print current status of Runtime Checking (RTC).

check -access
Turn on runtime access checking.

check -all
Equivalent to check -access and check -leaks.

check -leaks
Turn on leak checking. See uncheck for information on how to turn off checking. See suppress for information on suppressing errors.

(end SPARC-specific commands)

clear
Clear all breakpoints at the current stopping point.

clear <u>lineno</u>
Clear all breakpoints at <u>lineno</u>.

(begin Solaris 2.x only commands)

collector <u>command</u> <u>list</u>
Specify one or more of the collector commands.

collector close
Close current experiment.

collector open
Open a new experiment.

collector profile <u>options</u>
Specify one or more of the profile options.

mode stack|pc_only|off

Specify profile data collection mode.

timer <u>milliseconds</u>

Specify profile timer period.

collector quit
Stop collecting performance data.

collector sample <u>options</u>
Specify one or more of the sample options.

mode continuous|manual|off

Specify data collection mode.

period <u>seconds</u>

Specify data collection period.

collector show <u>options</u>
Show settings of one or more categories.

<u>no</u> <u>options</u> - Show all settings.

profile - Show profile settings.

sample - Show sample settings.

store - Show store settings.

working_set - Show working_set settings.

collector status
Inquire status about current experiment.

collector store <u>options</u>
Specify one or more of the store options.

directory <u>string</u>

Specify directory where experiment is stored.

filename <u>string</u>

Specify experiment file name.

collector working_set <u>options</u>
Specify one or more of the working set options.

mode on|off

Specify data collection mode.

(end Solaris 2.x only commands)

commands
List all builtin commands. See also import.

cont [sig sig] [id]
Continue execution with signal signo. The id, if present, specifies which tid or lid to continue. Default is all.

cont at line [sig sig id]
Continue execution at line line with signal sig. See continue for dbx command language loop control. The id, if present, specifies which tid or lid to continue. Default is all.

continue
Continue executing commands with the next iteration of the nearest enclosing loop (while, until, or for).

Continue execution with the next iteration of the nth enclosing loop. See cont for process control.

dalias
(Dbx alias) List all currently defined aliases

dalias name
List the definition, if any, of alias name

dalias name def
Define name to be an alias for def. def may contain white space. A semicolon or newline terminates the definition. See also kalias.

dbxenv
Display the dbx changeable variables and their current values.

The following dbxenv commands have the given default values:

case - sensitive

shortfnames - off

makeargs - "CC=cc -g"

speed - 0.50

stringlen - 512

autoflush - on

logfile - /tmp/dbx.log.PID

stepevents - off

stackmaxsize - 100

stackverbose - on

lookaside - off

warnings - -1

follow_fork_mode - parent

follow_fork_inherit - off

compress_stabs - on

aout_cache_size - 1

lo_cache - on

mt_watchpoints - off

func_overload - on

oper_overload - on

pop_autodestruct - on

setpgrp - off

dynamic_type - off

errlogfile - /tmp/dbx.errlog.PID

autocontinue - off

error_limit - 1000

autosuppress - off

thread_stats - off

dbxenv case sensitive
Treat upper and lower case distinctly

dbxenv case insensitive

```
          Fold upper and lower case to lower case

     dbxenv shortfname on|off
          Display short pathnames for files.

     dbxenv makeargs args
          Set the arguments passed to make

     dbxenv speed num
          Set the speed of tracing execution. This is a  floating
          point value representing seconds.

     dbxenv stringlen num
          Set maximum number of chars printed for char *.

     dbxenv autoflush on|off
          Automatically call fflush  after  every  function  call
          made in the debugged process by dbx.

     dbxenv logfile filename
          Rename command logfile to filename

     dbxenv stepevents on|off
          Allow breakpoints while step/next'ing

     dbxenv stackmaxsize num
          Set the default number of frames for the where command.

     dbxenv stackverbose on|off
          Governs the printing of arguments and line  information
          in where.

     dbxenv lookaside on|off
          Find file static symbols, even when not in scope.

     dbxenv warnings number
          Limit number of incomplete class warnings; -1 means  no
          limit

(begin Solaris 2.x only commands)
     dbxenv follow_fork_mode ...
          When process executes a fork/vfork/fork1 ...

     dbxenv follow_fork_mode parent
          Stay with parent.

     dbxenv follow_fork_mode child
          Follow child.

     dbxenv follow_fork_mode both
          Follow both parent and child. Only valid  when  running
          under the Debugger.
```



```
     dbxenv follow_fork_mode ask
          Ask which of the above the user wants.

     dbxenv follow_fork_inherit ...
          When following child ...

     dbxenv follow_fork_inherit on
          Inherit events.

     dbxenv follow_fork_inherit off
          Do not inherit events.

(end Solaris 2.x only commands)
     dbxenv compress_stabs on|off
          Read debugging information for each include  file  only
          once.

     dbxenv aout_cache_size num
          Size of a.out loadobject cache;  set  this  to  n  when
          debugging  n  programs  (serially) from a single dbx. A
          num of zero still allows  caching  of  shared  objects.
          See next entry.

     dbxenv lo_cache on|off
          Enable/disable loadobject cache entirely.

(Solarix 2.x)
     dbxenv mt_watchpoints on|off
          Allow/disallow watchpoint facility for MT programs.

     dbxenv dynamic_type on|off
          When on, -d is the default  for  printing,  displaying,
          and inspecting.

     dbxenv func_overload on|off
          If on, do automatic overload resolution.

     dbxenv oper_overload on|off
          If on, do automatic operator overload  resolution  (C++
          only).

     dbxenv pop_autodestruct on|off
          If on, automatically call appropriate  destructors  for
          locals when 'pop'ing a frame.

     dbxenv setpgrp on|off
          If on, when a program  is  run,  setgprp(2)  is  called
          right after the fork.

     dbxenv errlogfile filename
          Name of file where RTC errors will be logged  if  auto-
          continue is set.
```

dbxenv autocontinue on|off
: Log RTC errors to errlogfile and continue.

dbxenv error_limit num
: Number of RTC errors that will be reported.

dbxenv autosuppress on|off
: If on, RTC error at a given location is reported only once.

Note that each of the above has a corresponding ksh environment variable: DBX_case, DBX_shortfname, etc. The variable may be assigned directly, or the dbxenv command may be used; they are equivalent.

debug
: Print the name and arguments of the program being debugged.

debug -c prog [core]
: Begin debugging execfile with corefile corefile.

debug -p prog pid
: Begin debugging execfile with process process-id.

debug -r | -clone
: The -r option, if present, causes dbx to retain all display's, trace's, when's, and stop's. If no -r option is given, an implicit delete all and undisplay 0 is performed. The clone option, if present, causes another dbx to begin execution, and thus permits debugging more than one process at a time. -clone is valid only if running under Debugger.

debug -clone
: Starts another dbx debugging nothing. Valid only if running under Debugger.

delete hid ...
: Remove trace's, when's, or stop's of given hid, see also status and handler.

delete 0 | all | -all
: Remove all trace's, when's, and stop's.

delete -temp
: Remove all temporary handlers.

demangle mangled name
: Demangle mangled name. If the mangled name begins with a $, it must be backslashed or otherwise quoted.

detach
: Detach the debugger from the debugee. Cancel any pending signals.

detach -sig sig
: Detach while forwarding the given signal.

dis
: Disassemble 10 instructions starting at the value of +. (See examine).

dis addr [count]
: Disassemble count instructions (default is 10), starting at address addr.

dis addr1 addr2
: Disassemble instructions from addr2 through addr2.

dis count
: Disassemble count instructions starting at +.

display
: Print the list of expressions being displayed.

display exp , ...
: Display the value of expressions exp , ... at every stopping point.

document cmd
: Allows the user to provide the documentation for a command cmd. The document command reads from standard input. To add documentation for a new command named foo use:

(dbx) document foo

foo arg #Echoes the arg with each letter converted to upper case.

^D

(Type Control-D to end the input)

To remove the documentation for command foo,use

(dbx) document foo </dev/null

down

Move down the call stack one level.

down number
: Move down the call stack number levels.

down -h number
    Move down the call stack, but don’t skip hidden frames.

dump Print all variables local to the current procedure.

dump proc
    Print all variables local to proc.

edit Edit the current file.

edit filename
    Edit the specified file filename.

edit proc
    Edit the file containing function or procedure proc.

eval cmd [args]
    Evaluate the command cmd as if it had been entered from the keyboard. This is useful when a command must be composed programmatically, then executed. Example:

    (dbx) x=’echo $$’

    (dbx) $x

    $$

    (dbx) eval $x

    1234

examine
    The examine command should be used to examine the contents of memory.

examine +
    Display the contents of the next address in the default format.

examine addr1 addr2
    Display the data at the given range of addresses in the default format.

examine addr1 addr2 / fmt
    Display the data at the given range of addresses in the given format and set the default format (fmt).

examine addr [count] fmt
    Display count items, beginning at the given address addr, in the given format fmt (and set the default format).

examine addr=fmt
    Display the value of the given address addr in the given format fmt.

examine addr
    Display the value of the given address in the default format. The addr may be + which indicates the next address to be displayed.

    Legal values for fmt are:

    i - instruction (disassembly)

    d,D - decimal (2 or 4 bytes)

    o,O - octal (2 or 4 bytes)

    x,X - hexadecimal (2 or 4 bytes)

    b - octal (1 byte)

    c - character

    w - wide character

    s - string

    W - wide character string

    f - hex and float (4 bytes, 6 digit prec.)

    F - hex and float (8 bytes, 14 digit prec.)

    g - same as F

    E - hex and float (16 bytes, 14 digit prec.)

exec cmd [args]
    Causes dbx to exec() the given command. This results in dbx being terminated and the new command running with the same process ID.

exists name
    Returns 0 if name is found in the current program, 1 if name is not found

exit Causes dbx to exit with return code 0. Same as quit.

exit n
    Exit with return code n. Same as quit n.

export

Lists all exported ksh variables.

export name
Marks the variable name as exported.

export name=value
Marks the variable name as exported and assigns it a new value.

export name1 name2 ...
Marks the given variables as exported.

false
Does nothing and returns non-zero (255).

fc (Fix Command) Invokes the editor ($FCEDIT) on the most recent command. The result is executed when you leave the editor.

fc -e editor
Use editor instead of $FCEDIT to edit the previous command.

fc -l
Lists the last 16 commands in the history.

fc -ln
Lists the last 16 commands without history numbers.

fc -lr
Lists the last 16 commands in reverse order.

fc -e cmd
Reexecute previous command which matches cmd. cmd may be a positive number (the history number), a negative number (subtracted from the current history number), a string (matches the most recent command starting with the given string), or nothing (meaning the previous command). The builtin alias r is set to fc -e.

fc -e - old = new cmd
Like the above, but substitutes new for old before executing the command.

fg Put the current job into the foreground.

fg %+
Put the current job into the foreground.

fg %-
Put the previous job into the foreground.

fg %n
Put job number n into the foreground.

fg %str

Put the job which begins with str into the foreground.

fg %?str
Put the job which contains str into the foreground. The % is optional in all cases.

file Print the name of the current file.

file filename
Change the current file.

files
List the names of all files which contribute debugging information to the current program (those which were compiled -g).

files regexp
List the names of all -g files that match the given regular expression.

fix Fix the current file. The current source file is recompiled and dynamically linked into the running process.

fix filename filename ...
Fix files in list.

fix -f
Force fixing the file, even if the source has not been modified.

fix -a
Fix all files modified.

fix -g
Strip -O flags and add -g flag.

fix -c
Print compilation line (may include some options added internally by dbx).

fix -n
No execution mode.

fixed
List names of all fixed files.

frame

Display the frame number of the current frame.

frame number
Set the current frame to frame number.

frame -h number
Set the current frame, even if frame number is hidden.

func Print the name of the current function.

func proc
Change the current function to function or procedure proc.

funcs
List all functions in the current program.

funcs -f filename
List all functions in file.

funcs regexp
List all functions which match the regular expression.

getopts opts name
Checks the positional parameters ($1, $2, etc.) for legal options, based on the value of opts, and assigns the result to name. opts is a string of option letters. If an option letter is followed by a colon (:), it requires an argument.

getopts opts name args ...
Use the given args rather than the positional parameters.

handler -enable hid , ...
Enable given handlers, all for all.

handler -disable hid , ...
Disable given handlers, all for all.

handler -reset hid
Reset trip counter for given handler.

handler -count hid newlimit
Set new count limit for given event.

handler -count hid
Print value of trip counter for given handler.

hash List the tracked aliases.

hash cmd

Make cmd a tracked alias.

hash -r
Unset all tracked aliases. When a UNIX command is executed, if the hashall option is set and the command was given as a simple name (no slashes), then the name is entered as a tracked alias for the pathname corresponding to the executed program.

help Print a summary of the builtin dbx commands.

help changes
Print changes to dbx commands.

help cmd
Print help about cmd.

help dbxrc
Print information about the dbx initialization file.

help topic subtopic
Print help for topic and subtopic.

help -k keyword
Print name of each topic that contains keyword.

hide List the stack frame filters currently in effect.

hide regexp
Hide stack frames matching regexp. The regular expression matches either the function name, or the name of the loadobject, and is a sh or ksh file matching style regular expression.

history
Print a list of the most recent commands.

history +
Resume recording command.

history -
Stop recording commands in the history.

history num
Resize the history list to hold num commands.

ignore
Print a list of the ignored signals.

ignore num ...
Ignore signal(s) numbered num.

```
     ignore  sig ...
          Ignore signal(s) named by sig.

     intercept  typename
          C++ only: Intercept throws of type  typename.

     intercept -a
          C++ only: Intercept all throws.

     intercept
          C++ only: List intercepted types.

(begin Solaris 2.x only commands)
     inspect  exp
          Display the expression in the Data  inspector,  if  the
          Data  Inspector  is  running; otherwise, just print the
          expression.

     inspect   -r
          C++ only, print the value of the expression.

     inspect  -d
          Show derived type of expression instead of static type,
          C++ only.

     inspect  +d
          Do not use derived type of expression when  the  dbxenv
          dynamic type is on, C++ only.

(end Solaris 2.x only commands)
     jobs List all active jobs. The list includes the job number,
          an  indicator  of the current (+) and previous (-) job,
          the process ID, a status indicator  (Stopped,  Running,
          Done,  Signal  n),  and the first several characters of
          the command. See fg and bg for more information.

     kalias
          (ksh alias) List all currently defined aliases.

     kalias  name
          List the definition, if any, of alias name.

     kalias  name = def ...
          Define name to be an alias for def. def must be  quoted
          if it contains white space. See also dalias.

     kill Kill the program being debugger.

     kill -l
          List all known signal numbers, names, and descriptions.

     kill  job ...
```



```
          Send the SIGTERM signal to the listed jobs.

     kill -sig job ...
          Send the given signal to the listed jobs.  job  may  be
          a  process  ID  or  may be specified in any of the ways
          listed under fg, but the %  is  not  optional  in  this
          case.

     kprint   arg ...
          (ksh print) Echo the arguments.

     kprint -n arg ...
          Echo the arguments, without a trailing newline.

     kprint -e arg ...
          Echo  the  arguments,   expanding   backslash   escapes
          (default setting).

     kprint  -r arg...
          Echo  the  arguments,   without   expanding   backslash
          escapes.

     kprint -u n   arg ...
          Echo the arguments to file descriptor n.

     kprint - arg ...
          Echo the arguments. This  ensures  that   arg   is  not
          treated  as an option, even if it starts with a dash (-
          ).  The builtin alias echo is equal to kprint.

     language
          Print the name of the current language.

     language  lang
          Set current language to lang, where lang is  either  c,
          ansic, c++, pascal, or fortran.

     let  arg...
          Evaluate one or more arithmetic expressions. The return
          code  is  0  (true) if the last expression evaluated is
          non-zero; 1 (false) otherwise.

     line Display the current line number.

     line num
          Set the current line number.

     list List 10 lines.

     list  num
          List line number num.
```

list *first* [,] *last*
List lines from *first* to *last*.

list *proc*
List the source to *proc*. In the second and third forms above, *num* or *last* may be $, which denotes the last line of the file.

loadobject
Print the name of the current loadobject.

loadobjects
List names of current loadobjects.

loadobjects -a
List names of all loaded loadobjects, even those not in use by the current program.

loadobjects -v
List text address ranges and names of current loadobjects.

loadobjects -a -v
List text address ranges and names of all loaded loadobjects.

(begin Solaris 2.x only commands)

lwp Display current LWP.

lwp *lwpid*
Switch to given LWP [*lwpid*].

lwps List all LWPs in the current process.

make Invoke make for the program being debugged.

menu *seltype* *cmd*
Define an item in the buttons menu (debugger) where selection type, *seltype*, is:

ignore Ignore the selection.

literal Take the selection literally.

expand Expand selection to longest syntactic identifier.

command Expand to a command from the command subwindow.

lineno Lineno from the source subwindow.

module
Print the name of the current module.

module [-f] [-q] *name*
Read in debugging information for the module *name*.

module [-f] [-q] -a
Read in debugging information for all program modules.

[-f] - Force reading of debugging info, even if the file is newer than the executable (use with caution!).

[-q] - Use Quiet mode.

modules [a]
List names of all program modules (with and without debugging info available).

modules -debug
List names of all modules containing debugging info.

modules -read
List names of modules containing debugging info that have been read in already.

next Step one line (skip OVER calls).

next *n*
Step *n* lines (skip OVER calls)

next -sig *sig*
Deliver the given signal while nexting.

next... *tid*
Step the given thread. When a function call is skipped over, all LWPs are implicitly resumed for the duration of that function call. Non-active threads cannot be stepped.

next... *lid*
Step the given LWP. Will not implicitly resume all LWPs when skipping a function.

nexti
Step one machine instruction (skip OVER calls).

nexti *n*
Step *n* machine instructions (skip OVER calls).

nexti -sig *sig*
Deliver the given signal while nexting.

nexti lid
The given lid.

nexti tid
The LWP on which the given thread is active. Will not implicitly resume all LWP when skipping a function.

pathmap
List all existing path mappings. The pathmapping can be used when an entire directory has been moved or is known by a different name on the machine being used to do the debugging.

pathmap -c from to
Establish a new mapping from from to to where from and to are filepath prefixes. If -c is used, the mapping is applied to the current working directory as well.

pathmap -d from
Delete the given mapping.

pop Pop current topframe from stack.

pop num
Pop num frames from stack.

pop -f num
Pop frames from stack until specified frame number.

print exp
Print the value of the expression(s) exp.

print -r exp
C++ only: Print the value of the expression exp including its inherited members.

pwd Print the current Working Directory.

quit Quit, exit dbx with return code 0. Same as exit.

quit n
Exit with return code n. Same as exit n. If a process was attached to, it is automatically detached from before exiting.

read name
Accept a line from stdin, assign it to the ksh variable name.

read n1 n2 ...
Accept a line from stdin, assign the first word to n1, the second word to n2 , etc. The last name receives

all of the remainder of the line if there are more words in the input than names.

read name?prompt
Print prompt on stderr, then accept a line from stdin and assign it to name.

read -e ...
Treat backslash at the end of a line as a line continuation character (default).

read -r ...
Do not treat backslash specially.

read -u n ...
Read from file descriptor n rather than stdin.

readonly
List all ksh variables with the -r attribute.

readonly name...
Set the -r attribute for the given variables.

readonly name=val ...
Assign a new value and then set the -r attribute for the given variables. See typeset for more information.

replay [-num]
Replay all (or all minus num) commands since last run/rerun debug command.

rerun
Begin executing the program with no arguments.

rerun args
Begin executing the program with new arguments.

restore [filename]
Restore dbx to the state it was in when it was saved by the save command.

return
Causes a ksh function to return the exit code of the last command executed.

return n
Causes a ksh function to return with exit code n.

rprint exp
Print the value of the expression exp. No special quoting rules apply, so that print a>b will put the value of a into file b.

rprint -r
    C++ only. Include any inherited members.

run Begin executing the program with the current arguments.

run args
    Begin executing the program with new arguments.

save [-num] filename
    Save all (or all minus num) commands since last run/rerun debug command to default file filename.

search string
    Search forward for string in the current file.

search
    Repeat search, using last search string

set Lists all ksh variables and their values

set arg
    Sets the positional parameters to arg...

set -- arg
    Sets the positional parameters, even if arg starts with a dash.

set -o opt
    Sets option opt.

set +o opt
    Clears option opt.

set -o
    Lists all currently set options.

set +o
    Lists all currently set options.

set -opt
    Sets the option with abbreviation opt.

set +opt
    Clears the option with abbreviation opt.

setenv name string
    Set environment variable name to string.

sh command
    Execute a shell command, even if command is a dbx builtin. For example: sh file foo

shift
    Shift the positional parameters one position to the left, dropping $1.

shift n
    Shift the positional parameters n positions to the left, dropping the first n arguments.

(SPARC)
showleaks
    Report new memory leaks since the last showleaks command.

source filename
    Execute commands from file filename. $PATH is not searched.

status
    Print trace's, when's, and stop's in effect.

status -s > filename
    The same, but redirect output into file filename For example:

    (dbx) status -s>bpts

    (dbx) source bpts

    See also the -r flag of the debug command.

status hid
    Print the status of handler hid.

step Single step one line (step INTO calls).

step n
    Single step n lines (step INTO calls).

step up
    And out of the current function.

step -sig sig
    And deliver the given signal.

step... tid
    Step the given thread. When a function call is skipped over, all LWPs are implicitly resumed for the duration of that function call. Non-active threads cannot be stepped.

step... lid
    Step the given LWP. Will not implicitly resume all LWPs

when skipping a function.

stepi
Single step one machine instruction (step INTO calls).

stepi n
Single step [n] machine instructions (step INTO calls).

stepi -sig sig
Single step one machine instruction (step INTO calls) and deliver the given signal.

stepi... id
Single step one machine instruction for only the given id.

stop [-update]
Stop execution now. This form is only valid within the body of a when.

stop -noupdate
Like above, but do not update various GUI displays.

stop at line
Stop execution at the line.

stop in func
Stop execution when func is called. stop has a general syntax as follows:

stop event-specification modifier. When the specified event is fired, the process is stopped.

stopi at addr
Stop execution at location addr.

stopi in func
Stop execution when func is called. stopi is similar to stop except that when the program stops due to a stopi, dbx will show the location in disassembly. stopi has a general syntax as follows:

stopi event-specification [modifier]. When the specified event is fired, the process is stopped.

(begin SPARC-specific commands)
suppress
History of suppress/unsuppress commands (not including "-d" and "-reset" commands).

suppress -d
List of errors being suppressed in functions that are

not compiled for debugging. This list is per loadobject. These errors can be unsuppressed only by using the unsuppress -d command.

suppress -d errors
Modify the default suppressions for all loadobjects by further suppressing error.

suppress -d errors in loadobjects
Modify the default suppressions in the loadobjects by further suppressing errors.

suppress -last
Suppress the error at the current error location.

suppress -reset
Set the default suppression mask to the original value (startup time).

suppress errors
Suppress errors everywhere.

suppress errors in [funcs] [files] [loadobjects]
Supress error in list of funcs and list of loadobjects.

suppress error at line
Suppress error at line.

suppress errors at file:line
Suppress errors at line in file.

suppress errors addr addr
Suppress errors at location addr.

errors are blank separated and can be any combination of:

all - All errors

aib - Possible memory leak - address in block

air - Possible memory leak - address in register

baf - Bad free

duf - Duplicate free

mel - Memory leak

maf - Misaligned free

mar - Misaligned read

```
          maw - Misaligned write

          oom - Out of memory

          rua - Read from unallocated memory

          rui - Read from uninitialized memory

          wro - Write to read-only memory

          wua - Write to unallocated memory

          See also the dbxenv autosuppress on|off command.

(end SPARC-specific commands)
     test  expr
          Returns true (0) if expr evaluates to true,  false  (1)
          otherwise.  The following operators are recognized:

          -r file        file is readable

          -w file        file is writable

          -x file        file is executable (searchable, if  file
          is a  directory)

          -f file        file is a regular file

          -d file        file is a directory

          -c file        file is a character special file

          -b file        file is a block special file

          -p file        file is a named pipe (FIFO)

          -u file        file has its set-uid-bit set

          -g file        file has its set-gid-bit set

          -k file        file has its sticky bit set

          -s file        file is greater than 0 bytes

          -O file       file is owned by this user

          -G file       file has same group ID as this user

          -L file       file is a symbolic link

          -S file       file is a socket special file
```



```
          -t fd          fd is an open file  descriptor  which  is
          associated with a tty

          -z string      string is zero length

          -n string      string is non-zero length

          -o opt        option opt is set

          !              negates following expression

          -o             binary 'or'

          -a             binary 'and'

          (expr)        grouping (must be quoted)

          Note: The -o operator is taken to mean option  only  if
          the word following it is one of the options accepted by
          the set command.

(begin Solaris 2.x only commands)
     thread
          Display current thread.

     thread  tid
          Switch to thread tid. In all the following  variations,
          an optional tid implies the current thread.

     thread -info [tid]
          Print everything known about the given thread.

     thread -locks [tid]
          Print all locks held by the given thread.

     thread -suspend [tid]
          Put the given thread into suspended state.

     thread -continue [tid]
          Un-suspend the given thread

     thread -hide [tid]
          "Hide" the given (or current) thread. It will not  show
          up in the generic 'threads' listing.

     thread -unhide [tid]
          "Unhide" the given (or current) thread.

     thread -unhide all
          "Unhide" all threads.

     threads
```

Print the list of all known threads.

threads -all
Print threads normally not printed (zombies).

threads -mode all|filter
Controls whether 'threads' prints all threads or filters them by default.

threads -mode auto|manual
Under the GUI this enables automatic updating of the thread listing in the Thread Inspector.

threads -mode
Echo the current modes.

(end Solaris 2.x only commands)

times
Prints the accumulated user and system times for dbx and for all of its child processes.

toolenv font _file_
Set debugger's font.

toolenv width _num_
Set debugger's width in characters.

toolenv srclines _num_
Set _num_ of lines in source subwindow.

toolenv cmdlines _num_
Set _num_ of lines in command subwindow.

toolenv displines _num_
Set _num_ of lines in display subwindow.

toolenv topmargin _num_
Set top margin in source subwindow.

toolenv botmargin _num_
Set bottom margin in source subwindow. toolenv is silently ignored unless dbx is running under control of the debugger graphical user interface.

trace
Trace each source line.

trace in _proc_
Trace each source line while in proc.

trace at _line_
Trace given source line.

trace _proc_
Trace calls to the procedure.

trace inmember _func_
Trace calls to any member function named _func_.

trace infunction _func_
Trace when any function named _func_ is called.

trace inclass _class_
Trace calls to any member function of _class_.

trace _exp_ at _line_
Print _exp_ when _line_ is reached.

trace _var_ in _proc_
Trace changes to the variable _var_ in the procedure _proc_. Any of the above forms may be followed by the form if _cond_ which acts like a filter. Trace output is emitted only if _cond_ evaluates to true.

tracei
Trace each instruction.

tracei _var_
Trace changes to _var_. Uses automatic stepi.

tracei var at _addr_
Trace value of _var_ at _addr_. Any of the above forms may be followed by the form if _cond_, which acts like a filter. Trace output is emitted only if _cond_ evaluates to true.

trap Displays currently set traps and actions.

trap _action_ _cond_
Causes dbx to execute _action_ whenever any _cond_ occurs. The possible values for action are:

null - ignore the condition

- - reset the handling to the default

omitted - same as '-'

cmd - execute the command when the condition is triggered

Valid values for the _cond_ are:

0 - occurs when dbx exits.

sig - a signal number

name - a signal name; ‘kill -l’ lists the valid names

EXIT - same as 0

true Does nothing and returns zero.

typeset
Displays the names of all ksh variables and their attributes.

typeset -f
Displays the names of all ksh functions.

typeset -x
Same as export.

typeset -r
Same as readonly.

typeset -i
Displays the names and values of all ksh variables with the -i (integer) attribute.

typeset -t
Displays the names and values of all ksh variables with the -t (traced) attribute.

typeset -q
Displays the names of all ksh functions with the -q (quote) attribute.

typeset -[xritq] _name_
Sets the attribute for the named variable or function.

typeset +[xritq] _name_
Clears the attribute.

typeset -i _n_ _var_
Sets the integer attribute and the output base for variable _var_. _n_ must be from 2 to 36, inclusive.

ulimit
Displays the current file size limit (same as -f).

ulimit -cdmstf
Displays the current hard limit for the following:

-c corefile size in blocks

-d size of the data segment (kilobytes)

-m physical memory (kilobytes)

-s size of the stack segment (kilobytes)

-t time limit (seconds)

-f size of file written (blocks)

ulimit -CDMSTF
Displays the current soft limit for each of the above.

ulimit -cdmstf _n_
Sets the hard limit (must be super user to raise a hard limit).

ulimit -CDMSTF _n_
Sets the soft limit (cannot be made greater than the current hard limit).

umask
Displays the current umask as a three-digit octal number.

umask _n_
Sets the umask to the given value (interpreted as an octal umber). The umask specifies the bits which will be set to zero in the file permissions of any file created by this process. Typical values are 022 (not writable by the group or the world), and 027 (not writable by the group, no access for the world).

unalias _name_
Remove the alias definition of the given names, if any.

unbutton _cmd_
Remove all buttons containing _cmd_ (debugger).

(begin SPARC-specific commands)
uncheck
Print current status of runtime checking.

uncheck -access
Turn off access checking.

uncheck -all
Equivalent to uncheck -access and uncheck -leaks.

uncheck -leaks
Turn off leak checking.

(end SPARC-specific commands)
undisplay exp,...

Undo a display exp command.

undisplay n
Undo the display command numbered n.

undisplay 0
Undo all display commands.

unhide 0
Delete all stack frame filters.

unhide regexp
Delete stack frame filter regexp.

unhide num
Delete stack frame filter number num. The hide command lists the filters with numbers.

unintercept typename
C++ only: Un-intercept throws of type typename.

unintercept -a
C++ only: Undo intercept -a.

unintercept
C++ only: List intercepted types.

unmenu cmd
Remove all menu items containing cmd (debugger).

unset var...
Remove the definition of the given ksh variables, if any.

unset -f func...
Remove the definition of the given ksh function, if any.

(begin SPARC-specific commands)

unsuppress
History of suppress/unsuppress commands.

unsuppress -d
List errors being unsuppressed in functions that are not compiled for debugging. This list is per loadobject. Any other errors can be suppressed only by using the suppress -d command.

unsuppress -d errors
Modify the default suppressions for all loadobjects by further unsuppressing errors.

unsuppress -d errors in loadobjects
Modify the default suppressions in the loadobjects by further unsuppressing errors.

unsuppress -last
At error location unsuppress present error.

unsuppress -reset
Set the default suppression mask to the original value (startup time).

unsuppress errors
Unsuppress errors everywhere.

unsuppress errors in [funcs] [files] [loadobjects]
Unsuppress errors in list of funcs and list of loadobjects.

unsuppress errors at line
Unsuppress errors at line.

unsuppress errors at file:line
Unsuppress errors at line in file.

unsuppress errors addr addr
Unsuppress errors at location addr.

(end SPARC-specific commands)

up
Move up the call stack one level.

up number
Move up the call stack number levels.

up -h [number]
Move up the call stack, but don’t skip hidden frames.

use
Print the directory search path.

use dir
Change the directory search path.

wait
Wait for the current background job to terminate.

wait job
Wait for the specified background job to terminate. See fg for the valid syntax for job.

whatis name
Print the declaration of the variable name.

whatis -t type
Print the declaration of the type type.

whatis -e exp
Print the type of the expression exp.

whatis -r typename
Print information about typename and its base classes.

when at line { cmd; }
Execute command(s) when line reached.

when in proc { cmd; }
Execute command(s) when proc called.

when has a general syntax as follows:

when event-specification [modifier] {cmd ... ;}

When the specified event is fired, the cmds are executed.

whence name
Displays the meaning of the given names. If the name is a keyword or ksh function, it displays just the name. If the name is a builtin command, it displays builtin and the command name. If the name is an alias, it displays the definition of the alias. If the name is a tracked alias or UNIX command, it displays the full pathname to the command.

whence -v name
Verbose display. For functions, includes the definition.

wheni at addr { cmd; }
Execute command(s) when address reached. wheni has a general syntax as follows: wheni event-specification [modifier] {cmd ... ;} When the specified event is fired, the cmds are executed.

where
Print a procedure traceback.

where num
Print the num top frames in the traceback.

where -f num
Start traceback from frame num.

where -h
Include hidden frames.

where -q
Quick traceback (only function names).

where -v
Verbose traceback (include function args and line info).

Any of the above forms may be followed by an id to obtain the traceback for the specified entity.

whereami
Display the source line corresponding to the current location (top of the stack), and the source line corresponding to the current frame, if different.

whereami -instr
Same as above, except that the current disassembled instruction is printed instead of the source line.

whereis name
Print all declarations of name

which name
Print full qualification of name.

whocatches type
Tell where (if at all) an exception of type type would be caught if thrown at the current point of execution. Assume the next statement to be executed is a throw x where x is of type type, and display the line number, function name, and frame number of the catch clause which would catch it.

ENVIRONMENT
dbx checks the environment variable EDITOR for the name of the text editor to use with the edit command. The numerous ksh environment variables are used also.

FILES

core	default core file
.dbxrc	local dbx initialization file
~/.dbxrc	user's dbx initialization file
.dbxinit	local dbx alternate (obsolete) initialization file
~/.dbxinit	user's dbx alternate (obsolete) initialization file

SEE ALSO
bcheck(1), csh(1), dbxrc(4), dbxinit(4), debugger(1), kill(1), ksh(1), make(1), rtc_patch_area(1), and Debugging a Program

For detailed information on the Collector, see Performance Tuning an Application.

NOTES

Because the cc command does not generate or support 8-bit symbol names,it is inappropriate to make dbx 8-bit clean. See cc(1V) for an explanation about why cc is not 8-bit clean.