NAME adb – general-purpose debugger

SYNOPSIS **adb** [**–w**] [**–k**] [**–I** *dir*] [**–P** *prompt*] [**–V** *mode*] [*objectfile* [*corefile* [*swapfile*]]]

AVAILABILITY SUNWcsu

DESCRIPTION

adb is an interactive, general-purpose debugger. It can be used to examine files and provides a controlled environment for the execution of programs.

objectfile is normally an executable program file, preferably containing a symbol table. If the file does not contain a symbol table, it can still be examined, but the symbolic features of **adb** cannot be used. The default for *objectfile* is **a.out**. *corefile* is assumed to be a core image file produced after executing *objectfile*. The default for *corefile* is **core**. *swapfile* is the image of the swap device used. It is valid only when used with the **–k** option.

OPTIONS

–w Create both *objectfile* and *corefile*, if necessary, and open them for reading and writing so that they can be modified using **adb**.

–k Perform kernel memory mapping; use when *corefile* is a system crash dump or **/dev/mem**, or when using a *swapfile*.

–I *dir* Specify a colon-separated list of directories where files to be read with **$<** or **$<<** (see below) will be sought; the default is **/usr/platform/***plat-name***/lib/adb:/usr/lib/adb**, where *plat-name* is the name of the platform implementation. *plat-name* can be found using the **–i** option of **uname**(1).

–P *prompt* Specify the **adb** prompt string.

–V *mode* (SPARC only) Specify the disassembly and register display mode. Options are: 0 (v8), 1 (generic V9), and 2 (v9 plus Sun Ultra-SPARC specific instructions). The default mode is determined by the type of corefile being examined.

USAGE

adb reads commands from the standard input and displays responses on the standard output. It does not supply a prompt by default. It ignores the QUIT signal. INTERRUPT invokes the next **adb** command. **adb** generally recognizes command input of the form:

[*address*] [**,** *count*] [*command*] [**;**]

address and *count* (if supplied) are expressions that result, respectively, in a new current address, and a repetition count. *command* is composed of a verb followed by a modifier or list of modifiers.

The symbol '**.**' represents the current location. It is initially zero. The default *count* is '**1**'.

Expressions

. The value of *dot*.
+ The value of *dot* incremented by the current increment.
^ The value of *dot* decremented by the current increment.
& The last *address* typed. (In older versions of **adb**, '"' was used.)
integer A number. The prefixes **0o** and **0O** indicate octal; **0t** and **0T**, decimal; **0x**

	and **0X**, hexadecimal (the default).
*int***.***frac*	A floating-point number.
'*cccc***'**	ASCII value of up to 4 characters.
<*name*	The value of *name*, which is either a variable name or a register name.
symbol	A symbol in the symbol table.
(*exp***)**	The value of *exp*.

Unary Operators

**exp*	The contents of location *exp* in *corefile*.
%*exp*	The contents of location *exp* in *objectfile* (In older versions of **adb**, '@' was used).
–*exp*	Integer negation.
˜*exp*	Bitwise complement.
#*exp*	Logical negation.

Binary Operators

Binary operators are left associative and have lower precedence than unary operators.

+	Integer addition.
–	Integer subtraction.
*	Integer multiplication.
%	Integer division.
&	Bitwise conjunction ("AND").
|	Bitwise disjunction ("OR").
#	*lhs* rounded up to the next multiple of *rhs*.

Variables

Named variables are set initially by **adb** but are not used subsequently.

0	The last value printed.
1	The last offset part of an instruction source.
2	The previous value of variable 1.
9	The count on the last **$<** or **$<<** command.

On entry the following are set from the system header in the *corefile* or *objectfile* as appropriate.

b	The base address of the data segment.
d	The data segment size.
e	The entry point.
m	The 'magic' number
t	The text segment size.

Commands

Commands to **adb** consist of a *verb* followed by a *modifier* or list of modifiers.

Verbs

?	Print locations starting at *address* in *objectfile*.
/	Print locations starting at *address* in *corefile*.
=	Print the value of *address* itself.
:	Manage a subprocess.
>	Assign a value to a variable or register.
RETURN	Repeat the previous command with a *count* of 1. Increment '**.**'.
!	Shell escape.

?, /, *and* **=** *Modifiers*

The following format modifiers apply to the commands **?**, **/**, and **=**. To specify a format, follow the command with an optional repeat count, and the desired format letter or letters:

{ **?,/,=** } [[*r*] *f* . . .]]

where *r* is a repeat count, and *f* is one of the format letters listed below:

o	('**.**' increment: 2) Print 2 bytes in octal.
O	(4) Print 4 bytes in octal.
q	(2) Print in signed octal.
Q	(4) Print long signed octal.
d	(2) Print in decimal.
D	(4) Print long decimal.
x	(2) Print 2 bytes in hexadecimal.
X	(4) Print 4 bytes in hexadecimal.
u	(2) Print as an unsigned decimal number.
U	(4) Print long unsigned decimal.
f	(4) Print a single-precision floating-point number.
F	(8) Print a double-precision floating-point number.
b	(1) Print the addressed byte in octal.
c	(1) Print the addressed character.
C	(1) Print the addressed character using ^ escape convention.
s	(*n*) Print the addressed string.
S	(*n*) Print a string using the ^ escape convention.
Y	(4) Print 4 bytes in date format.
i	(*4* on SPARC; *n* on x86) Print as machine instructions.
a	(0) Print the value of '**.**' in symbolic form.
p	(4) Print the addressed value in symbolic form.
t	(0) Tab to the next appropriate TAB stop.
r	(0) Print a SPACE.
n	(0) Print a NEWLINE.
". . .**"**	(0) Print the enclosed string.
^	(0) Decrement '**.**'.
+	(0) Increment '**.**'.
−	(0) Decrement '**.**' by 1.

? and / Modifiers

l *value mask*	Apply *mask* and compare for *value*; move '**.**' to matching location.
L *value mask*	Apply *mask* and compare for 4-byte *value*; move '**.**' to matching location.
w *value*	Write the 2-byte *value* to address.
W *value*	Write the 4-byte *value* to address.
m *b1 e1 f1*[?]	Map new values for *b1, e1, f1*. If the **?** or **/** is followed by ∗ then the second segment (*b2* , *e2* , *f2*) of the address mapping is changed.
v	Like **w**, but writes only bytes at a time.

: *Modifiers*	**b** *commands*	Set breakpoint, execute *commands* when reached.
	r	Run *objectfile* as a subprocess.
	d	Delete breakpoint at *address*.
	z	Delete all breakpoints.
	c*s*	x86: The subprocess is continued with signal *s*.
	s*s*	Single-step the subprocess with signal *s*.
	i	Add the signal specified by *address* to the list of signals passed directly to the subprocess.
	t	Remove the signal specified by *address* from the list implicitly passed to the subprocess.
	k	Terminate (kill) the current subprocess, if any.
	A	Attach **adb** to an existing process id. (For example, **0t1234:A** would attach **adb** to decimal process number **1234**.)
	R	Release the previously attached process.
$ *Modifiers*	**<***filename*	Read commands from the file *filename*.
	<<*filename*	Similar to **<**, but can be used in a file of commands without closing the file.
	>*filename*	Append output to *filename*, which is created if it does not exist.
	l	x86: Show the current lightweight process (lwp) ID.
	L	x86: Show all the lwp IDs.
	P	Specify the **adb** prompt string.
	?	Print process ID, the signal which stopped the subprocess, and the registers.
	r	Print the names and contents of the general CPU registers, and the instruction addressed by **pc**.
	x or **X**	x86: Print the contents of floating point registers. **$x** and **$X** accept a "count" which determines the precision in which the floating point registers will be printed; the default is **25**. Using **$X** will produce more verbose output than using **$x**.
	x	SPARC: Print the names and contents of floating-point registers 0 through 15.
	X	SPARC: Print the names and contents of floating-point registers 16 through 31.
	b	Print all breakpoints and their associated counts and commands.
	c	C stack backtrace. On SPARC systems, it is impossible for **adb** to determine how many parameters were passed to a function. The default that **adb** chooses in a **$c** command is to show the six parameter registers. This can be overridden by appending a hexadecimal number to the **$c** command, specifying how many parameters to display. For example, the **$cf** command will print 15 parameters for each function in the stack trace.
	C	x86: Same as **$c**, but in addition it displays the frame pointer values.
	d	Set the default radix to *address* and report the new value. Note: *address* is interpreted in the (old) current radix. Thus '**10$d**' never changes the default radix.

e	Print the names and values of external variables.
w	Set the page width for output to *address* (default 80).
s	Set the limit for symbol matches to *address* (default 255).
o	All integers input are regarded as octal.
q	Exit from **adb**.
v	Print all non zero variables in octal.
m	Print the address map.
f	Print a list of known source filenames.
p	(*Kernel debugging*) Change the current kernel memory mapping to map the designated **user structure** to the address given by *u*; this is the address of the user's **proc** structure.
i	Show which signals are passed to the subprocess with the minimum of **adb** interference.
V	SPARC: Change the current disassembly and register display mode. Options are: 0 (v8), 1 (generic V9), and 2 (v9 plus Sun Ultra-SPARC specific instructions). Omitting the numeric parameter prints information on the current disassembly mode.
W	Reopen *objectfile* and *corefile* for writing, as though the −**w** command-line argument had been given.

EXAMPLES

To start **adb** on the running kernel, use (as **root**):

example# adb -k /dev/ksyms /dev/mem

/dev/ksyms is a special driver that provides an image of the kernel's symbol table. This can be used to examine kernel state and debug device drivers. Refer to the Debugging chapter in *Writing Device Drivers* for more information.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	The last command either failed or returned a nonzero status.

FILES

/usr/lib/adb and **/usr/platform/***platform-name***/lib/adb**
default directories in which files are to be read with **$<** and **$<<**. *platform-name* is the name of the platform implementation and can be found using **uname** −**i**.

a.out default name for *objectfile* operand.

core default name for *corefile* operand.

/dev/ksyms special driver to provide an image of the kernel's symbolic table.

SEE ALSO

uname(1), **ptrace**(2), **a.out**(4), **core**(4), **proc**(4), **ksyms**(7D)

Writing Device Drivers

DIAGNOSTICS

adb, when there is no current command or format, comments about inaccessible files, syntax errors, abnormal termination of commands, etc.

NOTES

adb should be changed to use the new format symbolic information generated by –**g**.

adb is platform and release dependent. Kernel core dumps should be examined on the same platform they were created on.

BUGS

Since no shell is invoked to interpret the arguments of the **:r** command, the customary wild-card and variable expansions cannot occur.

Since there is little type-checking on addresses, using a sourcefile address in an inappropriate context may lead to unexpected results.

The **$c***parameter-count* command is a workaround.