NAME

ldd – list dynamic dependencies of executable files or shared objects

SYNOPSIS

ldd [**–d** | **–r**] [**–f**] [**–s**] [**–v**] *filename*. . .

AVAILABILITY

SUNWtoo

DESCRIPTION

ldd lists the dynamic dependencies of executable files or shared objects. If *filename* is an executable file, **ldd** lists the pathnames of all shared objects that would be loaded as a result of executing *filename*.

If *filename* is a shared object, **ldd** lists the pathnames of all shared objects that would be loaded as a result of loading *filename*. **ldd** expects shared objects to have execute permission, and if this is not the case will issue a warning before attempting to process the file.

ldd processes its input one file at a time. For each input file **ldd** performs one of the following:

- Lists the object dependencies if they exist.
- Succeeds quietly if dependencies do not exit.
- Prints an error message if processing fails.

OPTIONS

ldd can also check the compatibility of *filename* with the shared objects it uses. The following options indicate to **ldd** to print warnings for any unresolved symbol references that would occur if *filename* were executed.

–d Check references to data objects.

–r Check references to both data objects and functions.

Only one of the above options may be given during any single invocation of **ldd**.

–f Force the checking of an insecure executable file. By default, when **ldd** is invoked by the super-user it will not process an insecure executable. An executable is determined to be insecure if the interpretor it specifies does not reside under **/usr/lib**, or **/etc/lib**, or if the interpretor cannot be determined.

–s Displays the search path used to locate shared object dependencies.

–v Displays **all** dependency relationships incurred when processing *filename*. This options also displays any dependency version requirements (see **pvs**(1)).

A super-user should use the –**f** option only if the executable being examined is known to be trustworthy, as use of –**f** while super-user on an untrustworthy executable may compromise system security. If it is unknown if the executable being examined is trustworthy, it is suggested that a super-user temporarily become a regular user, and invoke **ldd** as that regular user. Untrustworthy objects can be safely examined with **dump**(1), and with **adb**(1) as long as the **:r** subcommand is not used. In addition, a non-super-user can use the **:r** subcommand of **adb**, and can also use **truss**(1), to examine an untrustworthy executable without too much risk of compromise. To minimize risk, it is recommended that the user id of "nobody" be used when using **ldd**, **adb :r**, or **truss** on an untrustworthy executable.

FILES

/usr/lib/lddstub Fake executable loaded to check the dependencies of shared objects.

SEE ALSO

adb(1), **dump**(1), **ld**(1), **pvs**(1), **truss**(1), **dlopen**(3X)

Linker and Libraries Guide

DIAGNOSTICS

ldd prints the record of shared object path names to **stdout**. The optional list of symbol resolution problems are printed to **stderr**. If *filename* is not an executable file or a shared object, or if it cannot be opened for reading, a non-zero exit status is returned.

NOTES

ldd does not list shared objects explicitly attached using **dlopen**(3X).

Using the −**d** or −**r** option with shared objects can give misleading results. **ldd** does a ''worst case'' analysis of the shared objects. However, in practice some or all of the symbols reported as unresolved can be resolved by the executable file referencing the shared object.

ldd uses the same algorithm as the runtime linker to locate shared objects.