NAME
lksh – Legacy Korn shell built on mksh

SYNOPSIS
lksh [-+abCefhiklmnprUuvXx] [-o opt][ -c string | -s | file [args ...]]

DESCRIPTION
lksh is a command interpreter intended exclusively for running legacy shell scripts. It is built on mksh; refer to its manual page for details on the scripting language. It is recommended to port scripts to mksh instead of relying on legacy or objectionable POSIX-mandated behaviour, since the MirBSD Korn Shell scripting language is much more consistent.

Do not use lksh as an interactive or login shell; use mksh instead.

Note that it’s strongly recommended to invoke lksh with -o posix to fully enjoy better compatibility to the POSIX standard (which is probably why you use lksh over mksh in the first place); -o sh (possibly additionally to the above) may be needed for some legacy scripts.

LEGACY MODE
lksh currently has the following differences from mksh:

• The KSH_VERSION string identifies lksh as “LEGACY KSH” instead of “MIRBSD KSH”. Note that the rest of the version string is identical between the two shell flavours, and the behaviour and differences can change between versions; see the accompanying manual page mksh(1) for the versions this document applies to.

• lksh uses POSIX arithmetic, which has quite a few implications: The data type for arithmetic operations is the host ISO C long data type. Signed integer wraparound is Undefined Behaviour; this means that...

    $ echo $(2147483647 + 1)

... is permitted to, e.g. delete all files on your system (the figure differs for non-32-bit systems, the rule doesn’t). The sign of the result of a modulo operation with at least one negative operand is unspecified. Shift operations on negative numbers are unspecified. Division of the largest negative number by -1 is Undefined Behaviour. The compiler is permitted to delete all data and crash the system if Undefined Behaviour occurs (see above for an example).

• The rotation arithmetic operators are not available.

• The shift arithmetic operators take all bits of the second operand into account; if they exceed permitted precision, the result is unspecified.

• Unless set -o posix is active, lksh always uses traditional mode for constructs like:

    $ set -- $(getopt ab:c "$@")
    $ echo $?

    POSIX mandates this to show 0, but traditional mode passes through the errorlevel from the getopt(1) command.

• Functions defined with the function reserved word share the shell options (set -o) instead of locally scoping them.

SEE ALSO
mksh(1)
CAVEATS

To use lksh as /bin/sh, compilation to enable set -o posix by default if called as sh (adding -DMKSH_BINSHPOSIX to CPPFLAGS) is highly recommended for better standards compliance.

For better compatibility with legacy scripts, such as many Debian maintainer scripts, Upstart and SYSV init scripts, and other unfixed scripts, also adding the -DMKSH_BINSHREDUCED compile-time option to enable both set -o posix -o sh when the shell is run as sh, as well as integrating the optional disrecommended printf(1) builtin, might be necessary.

lksh tries to make a cross between a legacy bourne/posix compatibl-ish shell and a legacy pdksh-alike but “legacy” is not exactly specified.

Talk to the MirBSD development team and users using the mailing list at <miros-mksh@mirbsd.org> (please note the EU-DSGVO/GDPR notice on http://www.mirbsd.org/rss.htm#lists and in the SMTP banner!) or the #!/bin/mksh (or #ksh) IRC channel at irc.freenode.net (Port 6697 SSL, 6667 unencrypted) if you need any further quirks or assistance, and consider migrating your legacy scripts to work with mksh instead of requiring lksh.