

LN Kdebug Manual

Jochen
IBM T. J. Watson Research Center
jochen@watson.ibm.com

December 16, 1997
Under Construction

1 General Remarks

About LNKD

LN Kdebug (LNKD) is primarily a tool to test LN and to fix bugs in LN. However, it can sometimes also be useful for testing higher layers. Consequently,

- LNKD always freezes the entire machine state when invoked. All interrupts are disabled, even the clock is halted. No kernel or user process proceeds while LNKD is active.
- LNKD is a stand-alone debugger. It does neither use parts of LN nor other device drivers. LNKD includes simple device drivers for keyboard, display (CGA), and the serial interfaces COM1 and COM2. These device drivers are *not* interrupt driven.

Although developed together with the Lava Nucleus (LN) and traditionally packaged together with LN, the LN Kdebug (LNKD) is *not* part of the LN μ -kernel. LN can run without LNKD or with another kernel debugger. There is no other connection between LN and LNKD that LNKD intimately knows LN's data structures.

About Exception Handling

LNKD is invoked by exceptions, in particular by the debug exception (INT 1) and the breakpoint exception (INT 3). Some exceptions are handled normally by LN, for example a page fault. All other exceptions are distributed by LN either to LNKD or to the current user-level thread. The according algorithm is:

```
exception dispatch:
  if exception occurred in kernel
    then invoke LNKD with kernel error
  elif current thread defined an IDT AND corresponding IDT entry  $\neq 0$ 
    then invoke user-level exception handler by upcall
    else invoke LNKD with user error
  fi .
```

Summarizing: LNKD is used for kernel errors. It is used for user errors only as long as no debugger is installed at user level. (*LNKD's user-level features might disappear in production versions of LN.*)

2 LN-Specific Functions

2.1 Thread Control Block: t

Displays the *commented thread control block*, the thread's *kernel stack*, and the thread's *general purpose registers*.

t	displays the thread control block of the current thread.
txxx	displays the thread control block of thread xxx. xxx is the global thread number (hexadecimal); leading zeros can be omitted.
txxx.yy	displays the thread control block of thread yy of task xxx. xxx and yy are both hexadecimal; leading zeros may be omitted. For example, in LN version 2, thread 1 of task 4 can be denoted by 4.01 or by the global thread number 201.

thread control block	The fields of the thread control block are labeled.
kernel stack	<p>The kernel stack of the displayed thread is dumped from stack bottom (lower right corner) up to the current stack top. When entering kernel mode, the processor first pushes SS, ESP, EFLAGS, CS and EIP onto the kernel stack. So the lowest line looks always like</p> <pre> ... userEIP userCS userEFLAGS userESP userSS </pre>
registers	<p>are displayed if the <i>currently active</i> thread is displayed. For this thread, LNKD knows that the registers are pushed beyond the displayed stack top. Depending on their state, other threads might not have saved all their registers on the stack.</p> <p>With the cursor keys, the virtual top of the displayed kernel thread can be changed. When pressing the enter key, LNKD assumes that the current stack top holds a return address and all registers are pushed beyond and accordingly displays the register values. This feature can be used for kernel stacks of the currently active thread as well as other threads.</p>

2.2 Modules: a

Displays LN modules and helps to decode LN addresses to module-relative addresses.

a	displays a list of all LN modules, LN Kdebug modules, Sigma0, and Ktest. The list contains name, date and base address(es) of its kcode- and code-segments (for kernel modules), or dcode segments (for LNKD), or. scode segments for Sigma0. Icode segment and code16 segment addresses are not included. The reported base addresses can be used for setting breakpoints relatively to the beginning of modules (see bb).
axxxx	finds the kernel module that corresponds to physical address <i>xxxx</i> provided that <i>xxxx</i> lies in one of the kcode-, code-, dcode- or scode-segments mentioned above. It gives base address <i>B</i> of the according module segment and offset address <i>X</i> so that $B + X = xxxx$. <i>X</i> can be used for the identifying the code by means of a listing or setting breakpoints relative to the module. <i>xxxx</i> is a hexadecimal number. Leading zeros can be omitted.

2.3 Kernel Data: k

Displays some global LN kernel data.

k	displays some kernel data that is not thread specific.
----------	--

2.4 Mapping: m

Permits to parse the LN mapping database.

mxxx	starts parsing the mapping subtree corresponding to physical page frame <i>xxxx000</i> . The cursor keys can be used to proceed to parent-, child-, and sister-entries.
-------------	---

3 General Test Functions

3.1 Breakpoint: b

Sets/resets one global breakpoint for kernel-mode and user mode.

b	displays the current breakpoint.
bi ...	sets an instruction breakpoint.
bw {1,2,4} ...	sets a data-write breakpoint for a 1-, 2- or 4-byte variable.
ba {1,2,4} ...	sets a data-access breakpoint for a 1-, 2- or 4-byte variable.
bp {1,2,4} ...	sets a breakpoint for in/out to a 1-, 2- or 4-byte io port.
... <i>xxxxxxxx</i>	The breakpoint address can be absolute or relative. If no breakpoint base address was specified before (see below) or if the breakpoint base address <i>B</i> is 0, <i>xxxxxxxx</i> is the breakpoint address. Otherwise, <i>B + xxxxxxxx</i> is the breakpoint address. <i>xxxxxxxx</i> is a hexadecimal number where leading zeros can be omitted.
bb <i>xxxxxxxx</i>	sets the breakpoint base address <i>B</i> to <i>xxxxxxxx</i> . <i>xxxxxxxx</i> is a hexadecimal number where leading zeros can be omitted.
b-	resets the system-global breakpoint.

br ...	restricts the breakpoint. The breakpoint exception invokes LNKD only when all set restrictions are met. Otherwise, the breakpoint exception is ignored.
brt <i>xxx</i>	restricts breakpoints to thread <i>xxx</i> .
brT <i>xxx</i>	restricts breakpoints to threads \neq <i>xxx</i> .
br { <i>eax, ebx, ecx, edx, ebp, esi, edi, eip</i> } [<i>yyyyyyyy, zzzzzzzz</i>]	The specified register (or the instruction pointer <i>eip</i>) must have a value in the specified interval (if <i>yyyyyyyy</i> \leq <i>zzzzzzzz</i>) or outside the interval [<i>zzzzzzzz, yyyyyyyy</i>] if <i>yyyyyyyy</i> $>$ <i>zzzzzzzz</i> .
br {1,2,4} <i>xxxxxxxx</i> [<i>yyyyyyyy, zzzzzzzz</i>]	The specified 1-, 2- or 4-byte variable must have a value in the specified interval (if <i>yyyyyyyy</i> \leq <i>zzzzzzzz</i>) or outside the interval [<i>zzzzzzzz, yyyyyyyy</i>] if <i>yyyyyyyy</i> $>$ <i>zzzzzzzz</i> .
br-	All breakpoint restrictions are reset.

Note: Breakpoint base and breakpoint restrictions are *not* reset when the breakpoint address or type are changed or when the breakpoint is reset (b-). Breakpoint base can be explicitly reset to 0; breakpoint restrictions can be explicitly reset (br-). However, changes of the base address or restrictions do not require prior reset. brt, brT, and br{e.1,2,4} restrictions are logically anded. However, setting a register restriction overwrites a prior variable restriction, and setting a variable restriction overwrites a prior register restriction.

3.2 Dump Memory: d

Displays physical and virtual memory.

d*xxxxxxxx* displays memory beginning from address *xxxxxxxx*. The dump is 32-bit-word oriented so that addresses are always truncated to 4-byte aligned addresses. The cursor keys, PgUp, and PgDn can be used to move the dump cursor and display further memory. Since some terminal emulators do not support PgUp and PgDn, Ctl p and Ctl q can be used instead. This command always displays virtual memory of the current address space. Pages that are not mapped to physical memory are shown as

Memory can be displayed in various modes. The blank key switches between these modes:

d-mode	32-bit words are shown as dwords, uppermost nibble leftmost, e.g. 00000002 for the value 2. Values 0 and FFFFFFFF are displayed specially: 0 and -1.
b-mode	32-bit words are shown bitwise, uppermost byte righthmost, e.g. 02000000 for the value 2. Values 0 and FFFFFFFF are not treated differently.
c-mode	Bytes are shown as ASCII characters.
p-mode	32-bit words are shown as page-table entries (see below).

3.3 Dump Page Tables: p

Displays page tables and virtual memory.

p	displays the higher-level page table (page directory) of the current address space. The cursor keys, PgUp, PgDn, Ctl p, and Ctl q can be used like in the dump case. Pressing the enter key when the cursor points to a valid lower-level page table switches to this table. In the same way, the enter key there switches to the data page. The home key always “returns” to the lower- or higher-level page table respectively.
pxxx	displays the higher-level page table of task <i>xxx</i> .
pxxxx000	displays the higher-level page table beginning at address <i>xxxxxxx</i> .

	Page-table entry formats:
-	Nil entry.
<i>xxxx</i> --r	User-level, read-only, pointing to physical address <i>xxxx000</i> .
<i>xxxx</i> --w	User-level, read/write, pointing to physical address <i>xxxx000</i> .
<i>xxxx</i> --R	Kernel, read-only, pointing to physical address <i>xxxx000</i> .
<i>xxxx</i> --r	Kernel, read/write, pointing to physical address <i>xxxx000</i> .
<i>xx/4</i> --{r,w,R,W}	4M-page entry, pointing to physical address <i>xx00000</i> .
<i>xx*4</i> --{r,w,R,W}	4M-page entry, pointing to physical address <i>xx00000</i> , 4K pages are mapped out of this 4M entry.

3.4 Page Fault Monitoring: P

Monitors page faults. Page faults are monitored *before* they are handled or even seen by LN.

P+	switches page-fault monitoring on. Whenever a page fault occurs, page-fault address, instruction pointer and thread number are displayed and the system stops until a key is hit on the debug console. Pressing the i-key invokes the full LNKD menu; any other key resumes normal operation, i.e. starts normal page-fault handling.
P-	switches page-fault monitoring off, single as well as traced monitoring.
P*	switches <i>traced</i> page-fault monitoring on. Instead of presenting any single page fault, all page faults are monitored in a trace buffer. This buffer stores up to 1792 entries. When LNKD is invoked the next time, the last up to 22 new entries (that were entered after the previous invocation of LNKD) of this buffer are displayed. More entries, including all old entries, can be displayed by the <i>dump trace</i> command.

Pr...	restricts the monitored page fault. Page faults that do not meet all specified restrictions are ignored by the monitoring system.
Pr $xxxx$	restricts page faults to thread $xxxx$.
PrT $xxxx$	restricts page faults to threads $\neq xxxx$.
Prx [$yyyyyyyy$, $zzzzzzzz$]	Only page faults with fault addresses in the specified interval (if $yyyyyyyy \leq zzzzzzzz$) or outside the interval [$zzzzzzzz$, $yyyyyyyy$] (if $yyyyyyyy > zzzzzzzz$) are monitored.
Pr-	All page-fault restrictions are reset.

3.5 IPC Monitoring: |

Monitors IPC operations. IPCs are monitored *before* they are handled or even seen by LN.

+	switches IPC monitoring on. Whenever an IPC operation is invoked, sender thread number, operation type (call, send, wait, wait for, reply and wait), destination thread number (if specified), message type (normal or map), and message words 0 and 1 are displayed. The system stops until a key is hit on the debug console. Pressing the i-key invokes the full LNKD menu; any other key resumes normal operation, i.e. starts normal IPC handling.
-	switches IPC monitoring off, single as well as traced monitoring.
*	switches <i>traced</i> IPC monitoring on. Instead of presenting any single IPC, all IPCs are monitored in a trace buffer. This buffer stores up to 1792 entries. When LNKD is invoked the next time, the last up to 22 new entries (that were entered after the previous invocation of LNKD) of this buffer are displayed. More entries, including all old entries, can be displayed by the <i>dump trace</i> command.

r ...	restricts the monitored IPCs. IPCs that do not meet all specified restrictions are ignored by the monitoring system.
rtxxxx	restricts IPCs to thread <i>xxxx</i> .
rTxxxx	restricts IPCs to threads <i>≠xxxx</i> .
rs	Only IPCs containing a send part are monitored, i.e., call, send, reply and wait.
r-	All IPC restrictions are reset.

3.6 Exception Monitoring: X

Monitors Exceptions. IPCs are monitored *before* they are handled or even seen by LN.

X+ <i>xx</i>	switches exception monitoring on for exception <i>xx</i> . For description of the exception types, see the Pentium processor Reference manual, e.d. <i>xx=0d</i> denotes a General Exception. Whenever an exception is raised, this is displayed. The system stops until a key is hit on the debug console. Pressing the i-key invokes the full LNKD menu; any other key resumes normal operation, i.e. starts normal IPC handling.
X-	switches exception monitoring off.
X*	switches <i>traced</i> exception monitoring on. Instead of presenting any single exceptions, all are monitored in a trace buffer. This buffer stores up to 1792 entries. When LNKD is invoked the next time, the last up to 22 new entries (that were entered after the previous invocation of LNKD) of this buffer are displayed. More entries, including all old entries, can be displayed by the <i>dump trace</i> command.

Xr...	restricts the monitored Exceptions. Exceptions that do not meet all specified restrictions are ignored by the monitoring system.
Xrt <i>xxxx</i>	restricts exceptions to thread <i>xxxx</i> .
XrT <i>xxxx</i>	restricts exceptions to threads <i>≠xxxx</i> .
Xrx [<i>yyyy,zzzz</i>]	Only exceptions with an error code in the specified interval are monitored. This restriction applies only to those exceptions that are accompanied by hardware-generated error codes, i.e., 0A, 0B, 0C, 0D, 0E.
Xr-	All exception restrictions are reset.

3.7 Tracing: T

Shows the trace buffer and specifies trace mode. IPCs, page faults, exceptions and kernel events. The according messages are stored in a trace buffer, together with timing information and, optionally, performance-counter information. On 486, timing information is not available. On Pentium and all further processors, each trace-buffer entry is accomplished with a 64-bit *timestamp* which is read from the processor's internal time-stamp-counter register. When performance counting is activated, each trace-buffer entry is also augmented with the current values of the processor's two performance-count registers.

T	enters trace-buffer dump. The newest information is displayed at the bottom. The cursor keys, PgUp, and PgDn can be used to move the trace-buffer cursor and display further trace-buffer entries. Since some terminal emulators do not support PgUp and PgDn, Ctl p and Ctl q can be used instead.
---	---

	On all processors except 486, trace-buffer entries get timestamps when they are entered into the buffer. Accordingly, they can be displayed in various timing modes. The blank key switches between these modes. Since the 486 processor offers no timestamp-counter register, on 486, only index-mode is available.
index-mode	Entries are shown with their number in the buffer. Links to other entries are also shown as entry numbers.
delta-mode	Entries are shown with their time difference (in μ s, ms, or s) to their displayed predecessor entry. Links to other entries are shown as time differences.
page-mode	Entries are shown with their time difference relative to the uppermost entry of the displayed page. Links are shown like in the delta mode.

	Trace-buffer entries can be displayed with <i>links</i> to similar entries. Two entries are considered to be <i>similar</i> iff both are of the same type (kernel event, page fault, ipc-call, ipc-send, etc.) <i>and</i> entered by the same thread <i>and</i> the first word is identical in both messages. The cursor-right/left keys switch between these alternate modes:
no-links	No links are shown
forward-links	Links to the next similar entry are shown, depending on the mode either as index links or as delta times.
backward-links	Links to the previous similar entry are shown, depending on the mode either as index links or as delta times.
perf-counters	Performance-counter deltas are shown instead of links. This feature is not available on 486 processors. See below how to activate performance monitoring.

P	permits to activate, deactivate and change performance monitoring <i>while you are in the trace-buffer dump</i> . (This command does not work outside the trace-buffer dump. Performance monitoring is not available on 486 processors.
P+	turns performance monitoring on. Kernel and user-mode activities are measured.
Pu	turns performance monitoring on. Only user-mode activities are measured.
Pk	turns performance monitoring on. Only kernel-mode activities are measured.

The following performance-monitoring options are available:

i	Instructions, counts total (U+V pipe) and V-pipe instructions.
c	Cache misses, counts data-cache and instruction-cache misses.
t	TLB misses, counts data-TLB and instruction-TLB misses.
m	Memory stalls, counts read and write stall cycles.
a	Interlocks, counts AGI cycles (address generation interlock) and bank conflicts occurrences.
b	Bus utilization, counts bus cycles and total number of executed instructions.

3.8 IO: i/o

I/O to ports.

$i\{1,2,4\}xxxx$	Reads (<i>in</i>) from the specified 1-, 2-, or 4-byte port.
$iaxxxx$	Reads (<i>in</i>) from the specified register of the processor's local APIC.
$iixxxx$	Reads (<i>in</i>) from the specified register of the IO APIC.
$ipxxxxxxxx$	Reads (<i>in</i>) from the specified PCI configuration register. It is not required to set bit 32 of $xxxxxxxx$. PCI configuration registers are always 4-byte registers. Their address $xxxxxxxx$ must always be 4-byte aligned.

$o\{1,2,4\}xxxx$ { $yy, yyyy, yyyyyyyy$ }	Writes (<i>out</i>) $yy \dots$ to the specified 1-, 2-, or 4-byte port.
$oaxxxx yyyyyyyy$	Writes (<i>out</i>) $yyyyyyyy$ to the specified register of the processor's local APIC.
$oixxxx$	Writes (<i>out</i>) $yyyyyyyy$ to the specified register of the IO APIC.
$opxxxxxxxx$	Writes (<i>out</i>) $yyyyyyyy$ to the specified PCI configuration register. It is not required to set bit 32 of $xxxxxxxx$. PCI configuration registers are always 4-byte registers. Their address $xxxxxxxx$ must always be 4-byte aligned.

4 Miscellaneous

4.1 Go: g

4.2 Video Mode and Remote Console: V

4.3 Remote ESC: R