Lcrash HOWTO

Andreas Herrman

aherrman@de.ibm.com

This document describes **lcrash**, the Linux crash dump analyzer.

Most commercial UNIX systems have a feature that dumps the real storage to disk in case of a system crash. Afterwards a dump−analysis tool is used to analyze such dumps of the system's memory state at the time of the system crash.

A team at SGI has worked on extensions of the Linux Kernel to provide such a dump feature for GNU/Linux. They called their project Linux Kernel Crash Dumps (LKCD). The analysis tool **lcrash** (Linux Crash) is a part of LKCD.

Please refer to *the LKCD Project Home Page*. The LKCD code was released under the GNU General Public License (GPL) and it is available from *sourceforge*.

This is the first version of the document. It is written in DocBook 4.1. Please let me know if you find any markup and other errors.

Table of Contents

Table of Contents

Chapter 1. Introduction

1.1. About lcrash

When your Linux system completely crashes or hangs the last thing you can do is to take a system memory dump and afterwards inspect the dump to identify the problem. Inspecting the dump you can use lcrash − the Linux crash dump analyzer.

lcrash is part of the lkcd project which was initiated by SGI. Please refer to the *Project Home Page* for details regarding this project.

lcrash has a command line interface with simple command line editing, history mechanism and − in recent versions − command line completion. Even a graphical interface exists for lcrash. It is called **qlcrash** and resides also at *sourceforge*.

Some important features of lcrash are:

- kernel structures are displayed in C−like fashion,
- virtual to physical address translation is automatically performed,
- kernel modules are supported when analyzing a dump.

1.2. About this HOWTO

This documentation was written because there was no document describing the usage of **lcrash**. It was started in June 2001. After creation of first draft versions written in LaTeX it was decided to use sgml and DocBook 4.1 in order to be compliant with the LDP (Linux Documentation Project). At this step not only the conversion from TeX to DocBook was made but there were also added a couple of sections.

This HOWTO covers **lcrash** version as of LKCD version 4.0.

The documentation is split into several chapters. The next chapter gives information of where to get the code, and how to compile and install the program. In Chapter 3 the general usage of **lcrash** is described. Chapter 4 is a reference of lcrash commands. Besides descriptions of all lcrash commands there are also provided many examples for several commands.

To complete the practical benefit of the documentation a Chapter 5 was included, which describes how to use **lcrash** in special situations of analyzing Linux kernel dumps.

If you have any questions regarding this document, its copyright, or publishing this document in non−electronic form, please contact <aherrman@de.ibm.com>.

Chapter 2. Installation

2.1. Where to get the code

As mentioned earlier, lcrash is part of LKCD. You can download packages containing the lcrash version of LKCD 4.0 from sourceforge in form of:

- a source rpm package,
- a rpm package containing binaries for i386.

2.2. Install rpm packages

To install the binary package, you can use:

```
bash# rpm -ivh lkcdutils-4.0-1.i386.rpm
```

This should install lcrash properly. No further installation steps are required.

Installation of source rpm is done using:

```
bash# rpm -ihv lkcdutils-4.0-1.src.rpm
```

This should install `lkcdutils-4.0-1.tar.gz` and `lkcdutils.spec` somewhere under `/usr/src`. On my SuSE system the files are saved under `/usr/src/packages/SOURCES/` and `/usr/src/packages/SPECS/`.

Now you can build and install lkcdutils using:

```
bash# cd /usr/src/packages/SPECS/
bash# rpm -bi lkcdutils.spec
```

Lcrash should know be built and installed properly as `/sbin/lcrash`. The lkcdutils source tree, which contains the lcrash sources, can be found under `/usr/src/packages/BUILD/lkcdutils-4.0/`.

2.3. Compile and Install lcrash

If you have installed the lcrash sources, you can build lcrash using:

```
bash$ cd lkcdutils-4.0
bash$ ./configure
bash$ make
```

Installation of lcrash and all other programs of lkcdutils package is done with:

```
bash# make install
```

This installs lcrash as `/sbin/lcrash`.

2.4. LKCD CVS Repository

The current code of LKCD and hence the newest lcrash sources are located at sourceforge.

Of course you can receive lcrash source code directly from cvs. To do so you can run: (Simply press **Enter**, when asked for a password.)

```
bash$ cvs -d:pserver:anonymous@cvs.lkcd.sourceforge.net:/cvsroot/lkcd login
(Logging in to anonymous@cvs.lkcd.sourceforge.net)
CVS password:
bash$ cvs -z3 -d:pserver:anonymous@cvs.lkcd.sourceforge.net:/cvsroot/lkcd co -d lkcdutils_today l
```

From this point you can follow instructions given in Section 2.3 to compile and install lcrash.

When using recent lcrash versions from cvs, please keep in mind, that this documentation may not yet reflect latest changes of lcrash.

Chapter 3. General Usage

3.1. Invoking Lcrash

Three input files are needed for **lcrash**:

- a map file providing the symbol table of the Kernel,
- a dump file containing the image of a system's memory to be analyzed,
- an object file in "stabs" debug format providing information of Kernel data types. [1]

Currently **lcrash** uses positional arguments. To invoke **lcrash** you can use the following command line:
lcrash *`symbol-table`* *`dump-file`* *`kern-types`*

Lcrash knows defaults for its arguments. They are given in table Table 3−1.

Table 3−1. Default values

Parameter	Default
`symbol-table`	`/boot/System.map`
`dump-file`	`/dev/mem`
`kern-types`	`/boot/Kerntypes`

If you are happy with all default values you can call **lcrash** without any arguments − as shown in the following example.

Example 3−1. Starting Lcrash

```
bash# lcrash
map = /boot/System.map, vmdump = /dev/mem, outfile = stdout, kerntypes = /boot/Kerntypes

Please wait...
    Loading system map ..................... Done.
    Loading type info (Kerntypes) ... Done.
    Loading ksyms from dump ...... Done.
>>
```

Lcrash only works correctly if *`symbol-table`*, *`kern-types`* and *`dump-file`* are from the same Kernel.

The `System.map` file is generated automatically when the Kernel is built. It contains symbol names of the Kernel and their corresponding Kernel addresses. Normally it is installed under `/boot/System.map`.

The file `/dev/mem` is used for analyzing the running Linux system. For parameter *`dump-file`* you can specify a file containing a dump that was generated with dump tools (see chapter \ref{chapter:DumpTools}).

The `Kerntypes` file is also generated in the Kernel build. But since `Kerntypes` is not in the standard Linux tree it is necessary to apply a specific "Kerntypes patch" before. The `Kerntypes` file is compiled with the `-gstabs` compile option which generates type information for all types defined in the Kerntypes source file. In the Kerntypes source file there are several includes for Kernel header files with important Kernel

structures.

The mentioned "Kerntypes patch" and the s390 dump tools can be downloaded from http://oss.software.ibm.com/developerworks/opensource/linux390/exp_src.html

3.2. User Interface

Lcrash provides a command line interface. This comes with basic command line editing and history mechanism, which will be described here.

3.2.1. History Mechanism

The default history size is 100 command lines and the maximum history size is 1000. Command line length is restricted to 1024 characters. To view history list or to change number of lines in history use the lcrash command \htmlref{'history'}{cmd:history}. An explanation of the history mechanism is given in Table 3−2.

Table 3−2. Command Line History

!!	Refer to the previous command. By itself, this substitution repeats the previous command.
!*n*	Refer to command line *n*.
!−*n*	Refer to the current command line minus *n*.
!*str*	Refer to the most recent command starting with *str*.

3.2.2. Command Line Editing

Supported keys for line editing are given in Table 3−3

Table 3−3. Command Line Keys

Ctrl−W	delete to previous word
Ctrl−D	delete current character
Ctrl−A	goto start of line
Ctrl−E	goto end of line
Ctrl−F	forward one character
Ctrl−B	backward one character
Ctrl−H	delete previous character
Ctrl−N	down history
Ctrl−K	erase to end of line (from cursor)
Ctrl−L	clear screen and redisplay prompt
Ctrl−P	up history
Ctrl−U	erase to beginning of line (from cursor)

Ctrl−R	redraw input line
Esc−F	forward one word
Esc−B	backward one word
Esc−D	delete next work
Esc−Del	delete previous word

Chapter 4. Lcrash Command Reference

4.1. Command Overview

Lcrash provides a whole bunch of commands. For some commands synonyms are provided. Furthermore the behavior of commands may be platform dependent or even a command is not available on a platform. A short overview of lcrash commands is given in table Table 4−1.

The following subsections explain lcrash commands in more detail. The commands can be grouped as shown in table Table 4−2 − hopefully this helps not to loose the overall view of the commands.

Table 4−1. Overview of lcrash commands

Command	Description	Aliases	alpha	i386	ia64	s390(x)
base	Display a number in binary, octal, decimal, and hex.		x	x	x	x
deftask	Set/display the default task.	**dt**	x	x	x	x
dis	Display the disassembled code.	**id**	x	x	x	x
dump	Display dump.	**md**, **od**	x	x	x	x
findsym	Display symbol information for given symbol addresses and names.	**fsym**, **symbol**	x	x	x	x
help	Display command help.	**?**	x	x	x	x
history	Set/display command history of lcrash.	**h**	x	x	x	x
ldcmds	Dynamically load a library of lcrash commands.		x	x	x	x
livedump	Create a system dump from live system memory.		x	x	x	x
load	Load a sial macro.		x	x	x	x
mktrace	Construct a stack backtrace from scratch.			x		
mmap	Display information for mm_struct structs.		x	x	x	x
module	Display information for module structs.		x	x	x	x
namelist	Add type information from namelist, list opened namelists.	**nmlist**, **addtypes**	x	x	x	x
page	Display information for page structs.		x	x	x	x
print	Evaluate and print expressions.	**p**,**pb**, **pd**,**po**, **px**	x	x	x	x
quit	Exit lcrash.	**q**, **q!**	x	x	x	x
report	Display a crash dump report.		x	x	x	x

s390dbf	Display Debug logs.					x
sizeof	Determine size of types. Display offset of struct members.	**offset**	x	x	x	x
stat	Display system statistics and the log_buf array.		x	x	x	x
strace	Displays all complete and unique stack traces.		x	x	x	x
symtab	Add/remove/list symbol table information.		x	x	x	x
task	Display information for task_struct structs.	**ps**	x	x	x	x
trace	Display stack trace for task_struct.	**t**	x	x	x	x
unload	Unload sial macros.		x	x	x	x
vi	Start a vi session of a sial file/function.		x	x	x	x
vtop	Determine the physical address of an virtual one.		x	x	x	x
walk	Walk a linked list of kernel structures or memory blocks.		x	x	x	x
whatis	Display type information and symbol information.		x	x	x	x

Table 4−2. Classification of lcrash commands

General Purpose	**base**, **help**, **history**, **ldcmds**, **quit**
Data Inspection	**dis**, **dump**, **print**, **vtop**, **walk**
Accessing Symbol and Type Information	**findsym**, **namelist**, **sizeof**, **symtab**, **whatis**
Support for Special Structures	**deftask**, **mmap**, **module**, **page**, **task**
Stack Tracing	**mktrace**, **strace**, **trace**,
Sial Support	**load**, **unload**, **vi**
Other Commands	**livedump**, **report**, **s390dbf**, **stat**

4.2. Common Options

Most lcrash commands have two things in common:

1. Command output can be piped to normal shell commands like **less** or **grep**.
2. They support the option `−w` to write output to a file.

To pipe the output of a command to **less**, just specify **lcrash_command | less**. Take care to use a blank before the pipe symbol, otherwise it could be misinterpreted by lcrash.

When using **lcrash_command** **-w** ***filename***, **lcrash** appends the output of the executed command to the file *filename*.

4.3. base

Usage

```
base [-w outfile] numeric_values[s]
```

Description

```
Display a number in binary, octal, decimal, and hexadecimal. A number in
a radix other then decimal should be preceded by a prefix that indicates
its radix as follows:

              0x      hexadecimal
              0       octal
              0b      binary
```

Example 4−1. base

```
>> base 4711 0x4711 04711 0b1000
--------------------------------------------------
    hex: 0x1267
decimal: 4711
  octal: 011147
 binary: 0b1001001100111
--------------------------------------------------
    hex: 0x4711
decimal: 18193
  octal: 043421
 binary: 0b100011100010001
--------------------------------------------------
    hex: 0x9c9
decimal: 2505
  octal: 04711
 binary: 0b100111001001
--------------------------------------------------
    hex: 0x8
decimal: 8
  octal: 010
 binary: 0b1000
--------------------------------------------------
```

4.4. deftask

Alias

dt

Usage

```
deftask [−w outfile] [task]
```

Description

```
Set the default task if one is indicated.  Otherwise, display the current
value of deftask. When 'lcrash' is run on a system core dump, deftask
gets set automatically to the task that was active when the system PANIC
occurred.When 'lcrash' is run on a live system, deftask is not set by
default.

The deftask value is used by 'lcrash' in a number of ways. The trace
command will display a trace for the default task if one is set. Also,
the translation of certain virtual addresses (user space) depends upon
deftask being set.
```

Note

Currently there is no possibility to reset the default task.

Example 4−2. deftask

```
>> task
ACTIVE TASKS:

    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
==========================================================================
  18e000      0      0      0      0         0  swapper
  5b0000      0      1      0      1       100  init
  5a8000      0      2      1      1        40  kmcheck
  59a000      0      3      1      1        40  keventd
  57c000      0      4      1      1       840  kswapd
  57a000      0      5      1      1       840  kreclaimd
  578000      0      6      1      1        40  bdflush
  576000      0      7      1      1        40  kupdated
 6edc000      0    231      3      1        40  keventd
 6ed0000      1    287      1      1       140  portmap
 6e60000      0    349      1      1        40  syslogd
 779a000      0    363      1      1       140  klogd
 6d54000      0    401      1      1       140  inetd
 6a0a000    100    448      1      1        40  xfs
 7ac0000      0    467      1      1         0  sulogin
 6948000      0    468    401      1       100  in.telnetd
 68f8000      0    469    468      1       100  login
 67e4000      0    470    469      1       100  bash
 61c8000      0    522    470      0       100  lcrash
==========================================================================
19 active task structs found

>> trace
System is ACTIVE. Set deftask.

>> deftask
No default task set

>> deftask 68f8000
```

```
Default task is 0x68f8000

>> trace
============================================================
STACK TRACE FOR TASK: 0x68f8000 (login)

 STACK:
 0 schedule+1076 [0x1c590]
 1 sys_wait4+1050 [0x23fc6]
 2 pgm_system_call+34 [0x130d0]
============================================================

>> deftask
Default task is 0x68f8000
```

4.5. dis

Usage

```
dis [−f] [−w outfile] [−F funcname]|addr[count|[bcount acount]]
```

Description

```
Display the disassembled code for addr for count instructions (the
default count is 1). Alternately, display the disassembled code for addr
with bcount instructions before and acount instructions after. If bcount
or acount is zero, then no instructions will be displayed before or after
respectively. If the dis command is issued with the −f command line
option, additional information will be displayed (opcode and byte size).
If the dis command is issued with the −F option followed by funcname,
disassembled code will be displayed for all instructions in the function.
```

Example 4−3. dis (i386)

```
>> dis −F memcmp
0xc0251878 <memcmp>:       pushl  %esi
0xc0251879 <memcmp+1>:     pushl  %ebx
0xc025187a <memcmp+2>:     movb   $0x0,%al
0xc025187c <memcmp+4>:     movl   0x14(%esp,1),%esi
0xc0251880 <memcmp+8>:     movl   0xc(%esp,1),%ecx
0xc0251884 <memcmp+12>:    movl   0x10(%esp,1),%edx
0xc0251888 <memcmp+16>:    testl  %esi,%esi
0xc025188a <memcmp+18>:    je     0xc02518a1 <memcmp+41>
0xc025188c <memcmp+20>:    jmp    0xc0251895 <memcmp+29>
0xc025188e <memcmp+22>:    movl   %esi,%esi
0xc0251890 <memcmp+24>:    incl   %ecx
0xc0251891 <memcmp+25>:    incl   %edx
0xc0251892 <memcmp+26>:    decl   %esi
0xc0251893 <memcmp+27>:    je     0xc02518a1 <memcmp+41>
0xc0251895 <memcmp+29>:    movb   (%edx),%al
0xc0251897 <memcmp+31>:    movb   (%ecx),%bl
0xc0251899 <memcmp+33>:    subb   %al,%bl
0xc025189b <memcmp+35>:    movb   %bl,%al
0xc025189d <memcmp+37>:    testb  %al,%al
0xc025189f <memcmp+39>:    je     0xc0251890 <memcmp+24>
0xc02518a1 <memcmp+41>:    movsbl %al,%eax
```

```
0xc02518a4 <memcmp+44>:    popl   %ebx
0xc02518a5 <memcmp+45>:    popl   %esi
0xc02518a6 <memcmp+46>:    ret
0xc02518a7 <memcmp+47>:    nop

>> dis 0xc025188e 10 -f
0xc025188e <memcmp+22>:    0x0089  movl   %esi,%esi  (2 bytes)
0xc0251890 <memcmp+24>:    0x0041  incl   %ecx  (1 byte)
0xc0251891 <memcmp+25>:    0x0042  incl   %edx  (1 byte)
0xc0251892 <memcmp+26>:    0x004e  decl   %esi  (1 byte)
0xc0251893 <memcmp+27>:    0x0074  je     0xc02518a1 <memcmp+41>  (2 bytes)
0xc0251895 <memcmp+29>:    0x008a  movb   (%edx),%al  (2 bytes)
0xc0251897 <memcmp+31>:    0x008a  movb   (%ecx),%bl  (2 bytes)
0xc0251899 <memcmp+33>:    0x0028  subb   %al,%bl  (2 bytes)
0xc025189b <memcmp+35>:    0x0088  movb   %bl,%al  (2 bytes)
0xc025189d <memcmp+37>:    0x0084  testb  %al,%al  (2 bytes)
```

Example 4-4. dis (s390)

```
>> idis 00154d8c 19
0x154d8c <memcmp>:       lhi    %r0,0
0x154d90 <memcmp+4>:     lr     %r5,%r2
0x154d92 <memcmp+6>:     j      0x154da2 <memcmp+22>
0x154d96 <memcmp+10>:    ahi    %r5,1
0x154d9a <memcmp+14>:    ahi    %r3,1
0x154d9e <memcmp+18>:    ahi    %r4,-1
0x154da2 <memcmp+22>:    ltr    %r4,%r4
0x154da4 <memcmp+24>:    je     0x154dc0 <memcmp+52>
0x154da8 <memcmp+28>:    ic     %r0,0(%r5)
0x154dac <memcmp+32>:    ic     %r1,0(%r3)
0x154db0 <memcmp+36>:    sr     %r0,%r1
0x154db2 <memcmp+38>:    lr     %r2,%r0
0x154db4 <memcmp+40>:    sll    %r2,24
0x154db8 <memcmp+44>:    sra    %r2,24
0x154dbc <memcmp+48>:    je     0x154d96 <memcmp+10>
0x154dc0 <memcmp+52>:    lr     %r2,%r0
0x154dc2 <memcmp+54>:    sll    %r2,24
0x154dc6 <memcmp+58>:    sra    %r2,24
0x154dca <memcmp+62>:    br     %r14
```

4.6. dump

Alias

md,od

Usage

```
dump [-d] [-o] [-x] [-B] [-D] [-H] [-W] [-w outfile] addr [count]
```

Description

```
Display count values starting at kernel virtual address addr in one of
the following formats: decimal (−d), octal (−o), or hexadecimal (−x).
The default format is hexadecimal, and the default count is 1. If addr is
preceeded by a pound sign ('#'), it will be treated as a page number
(PFN).
```

Note

Output of dump command depends on endianess of the host platform. E.g. on i386 lcrash will show words, half−words and double−words in little endianess. In conclusion on little endian platforms only the option −B will force lcrash to show you the bytes in the order as they really occur in the dump.

Example 4−5. dump

```
>> dump c02e4820 8 −o
0xc02e4820: 0000000011417432074 0000000017035267151
            0000000016231273040 0000000015633664563
0xc02e4830: 0000000006213431040 0000000004016030456
            0000000015733671050 0000000014524040164

>> dump c02e4820 8 −d
0xc02e4820: 01279145020 02020961897 01919252000 01852795251
0xc02e4830: 00841888288 00540553518 01869574696 01699758196

>> dump c02e4820 8 −x
0xc02e4820: 4c3e343c 78756e69 72657620 6e6f6973 : <4>Linux version
0xc02e4830: 322e3220 2038312e 6f6f7228 65504074 :  2.2.18 (root@Pe

>> dump c02e4820 8 −W
0xc02e4820: 4c3e343c 78756e69 72657620 6e6f6973 : <4>Linux version
0xc02e4830: 322e3220 2038312e 6f6f7228 65504074 :  2.2.18 (root@Pe

>> dump c02e4820 8 −B
0xc02e4820: 3c 34 3e 4c 69 6e 75 78                         : <4>Linux

>> dump c02e4820 8 −H
0xc02e4820: 343c 4c3e 6e69 7875 7620 7265 6973 6e6f : <4>Linux version

>> dump c02e4820 8 −D
0xc02e4820: 78756e694c3e343c 6e6f697372657620 : <4>Linux version
0xc02e4830: 2038312e322e3220 655040746f6f7228 :  2.2.18 (root@Pe
0xc02e4840: 75732e6d7569746e 28202965642e6573 : ntium.suse.de) (
0xc02e4850: 7372657620636367 35392e32206e6f69 : gcc version 2.95
```

4.7. findsym

Alias

fsym,**symbol**

Usage

```
findsym
        symname | symaddr [symname | symaddr [...] ]
        −f string [...]
        [−w outfile]
```

Description

```
Display relevant information for each requested symbol name and/or symbol
address.

OPTIONS:
 symname | symaddr [symname | symaddr [...] ]
       Search symbol information for given symbol names and addresses.
 −f string [...]
       Search symbol information for symbols which names start with given
       strings. Use this version if you don't know the full symbol name.
```

Example 4−6. findsym

```
>> findsym 0xc0150000
    ADDR  OFFSET  TYPE         NAME
============================================================
c0150000     144  GLOBAL_TEXT  ext2_truncate
============================================================
1 symbol found

>> findsym ext2_truncate
    ADDR  OFFSET  TYPE         NAME
============================================================
c014ff70       0  GLOBAL_TEXT  ext2_truncate
============================================================
1 symbol found

>> findsym 0xc0300000 init_mm module_list 0xc02f0000 memcmp
    ADDR  OFFSET  TYPE         NAME
============================================================
c0300000     480  GLOBAL_DATA  ip_masq_d_table
c02a90a0       0  GLOBAL_DATA  init_mm
c02ad128       0  GLOBAL_DATA  module_list
c02f0000     800  LOCAL_DATA   ro_bits
c0251878       0  GLOBAL_TEXT  memcmp
============================================================
5 symbols found
```

4.8. help

Alias

?

Usage

```
help [−w outfile] [all | command_list]
```

Description

```
Display a description of the named functions, including syntax.  The
'all' option displays help information for every command.
```

Example 4−7. help

```
>> help
?               id              p               sizeof
base            ldcmds          page            stat
bt              livedump        pb              strace
deftask         load            pd              symtab
dis             md              po              t
dt              mktrace         print           task
dump            mmap            ps              trace
findsym         module          px              unload
fsym            mt              q               vi
h               namelist        q!              vtop
help            nmlist          quit            walk
history         od              report          whatis

>> ? h
command: history [n]

    Without the optional parameter, displays the current history. Optional
    argument 'n' specifies the number of commands that are kept in the
    history list.
```

4.9. history

Alias

h

Usage

```
history [n]
```

Description

```
Without the optional parameter, displays the current history. Optional
argument 'n' specifies the number of commands that are kept in the
history list.
```

Note

To find out how the history mechanism works, please refer to Section 3.2.

Example 4−8. history

```
>> history
  1: base 4711 0x4711 04711 0b1000
  2: help
  3: ? h

>> h 2

>> h
  2: help
  3: ? h
```

4.10. ldcmds

Usage

```
ldcmds cmd_library
```

Description

```
Dynamically load a library of lcrash commands.
```

4.11. livedump

Usage

```
livedump [−l level]
```

Description

```
Create a system dump from live system memory.
```

4.12. load

Alias

Usage

```
load filename|directory
```

Description

```
Load a sial macro from a file or a directory. In the case of a directory,
all files in that directory will be loaded.
```

4.13. mktrace

Platform Dependency

i386

Alias

mt

Usage

```
mktrace [-l] [-w outfile] [stack_addr SP PC FP RA] | [-F [-a] [free_list]]
```

Description

```
Construct a stack backtrace from scratch using an arbitrary stack_addr,
SP, PC, FP, and RA. Alternately, free a trace record that was previously
allocated, list currently allocated trace records, and delete selected or
all active trace records.
```

4.14. mmap

Alias

Usage

```
mmap [-f] [-n] [-w outfile] mmap_list
```

Description

```
Display relevant information for each entry in mmap_list.
```

Example 4-9. mmap

```
>> task ce4ac000
    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
==========================================================================
ce4ac000   4640   1966   1951      1         0  netscape
==========================================================================
1 active task struct found

>> print ((task_struct*)ce4ac000)->mm
0xc97e7540
```

```
>> mmap 0xc97e7540
    ADDR  MM_COUNT  MAP_COUNT      MMAP
========================================
c97e7540         1         40  c571fa60
========================================
1 active mm_struct struct found

>> mmap -f 0xc97e7540
    ADDR  MM_COUNT  MAP_COUNT      MMAP
========================================
c97e7540         1         40  c571fa60

  START_CODE:0x8048000, END_CODE:0x8b5d422
  START_DATA:0x0, END_DATA:0x8d4be68
  START_BRK:0x8d99664, START_STACK:0xbffff210
  ARG_START:0xbffff3a6, ARG_END:0xbffff3b3
  TOTAL_VM:0x10ba

========================================
1 active mm_struct struct found

>> mmap -n 0xc97e7540
    ADDR  MM_COUNT  MAP_COUNT      MMAP
========================================
c97e7540         1         40  c571fa60

    ADDR  VM_START    VM_END  VM_PGOFF VM_FLAGS
------------------------------------------------
c571fa60   8048000   8b5e000         0     1875
c571f220   8b5e000   8d4c000  11620352     1873
c571f320   8d4c000   8dce000         0       77
c571f2a0  40000000  40016000         0      875
c571f160  40016000  40017000     86016      873
c571f9e0  4002a000  4002b000         0       73
c571ff60  4002b000  4002c000         0       75
c59dfd20  4002c000  4002d000         0       73
c571f0a0  4002d000  40076000         0       75
c59dfee0  40076000  4007a000    294912       73
c6582b20  4007a000  4007b000         0       73
c59df120  4007b000  40083000         0       75
c274d7e0  40083000  40085000     28672       73
c274d120  40085000  4009a000         0       75
c274dee0  4009a000  4009c000     81920       73
c274dd60  4009c000  4009d000         0       73
c274d9a0  4009d000  400b1000         0       75
c274da60  400b1000  400b2000     77824       73
c274dc60  400b2000  400b3000         0       73
c274daa0  400b3000  400c0000         0       75
c274dc20  400c0000  400c2000     49152       73
c274de20  400c2000  400cf000         0       75
c274dbe0  400cf000  400d0000     49152       73
c274d560  400d0000  401ad000         0       75
c274d660  401ad000  401b3000    901120       73
c274d5a0  401b3000  401b4000         0       73
c274dd20  401b4000  401b6000         0       75
c274db60  401b6000  401b7000      4096       73
c274da20  401b7000  401f0000         0       75
c274d8a0  401f0000  401fc000    229376       73
c274dea0  401fc000  401ff000         0       73
c274d860  401ff000  4021c000         0       75
c274db20  4021c000  4021d000    114688       73
c274dba0  4021d000  40326000         0       75
```

```
c274d760  40326000  4032c000   1081344       73
c274d060  4032c000  40330000         0       73
c274d2a0  50000000  50002000         0       70
c571f060  50002000  50012000      8192       77
c274d8e0  50012000  50014000     73728       70
c571fc60  bfffd000  c0000000     -8192      177
-----------------------------------------------

=======================================
1 active mm_struct struct found
```

4.15. module

Usage

```
module
        [modulename]
        [-f [modulename]]
        [-i iteration_threshold]
        [-w outfile]
```

Description

```
Display list of loaded modules and module symbols.

OPTIONS:
 modulename
       Display information of (all) module structure(s) in linked list
       module_list of the kernel.
       Shows address of module structure, and size, usecount, name of
       module, and modules that depend on the module.
       Equals "cat /proc/modules" in a running Linux system.
 -f [modulename]
       Show list of exported module symbols of (all) module structure(s)
       in linked list module_list of the kernel.
       Equals "cat /proc/ksyms" in a running Linux system.
 -i iteration_threshold
       By default certain loops are interrupted after 10'000 iterations
       to avoid endless loops while following invalid pointers. Using
       this option you can change the threshold for the current command.
       A value '0' means infinite iteration threshold, i.e. no
       interruption of the loop is caused by reaching the threshold.

 The kernel_module can be accessed by using "kernel_module" as
 modulename.
```

Example 4-10. module

```
>> module
    ADDR      SIZE USED NAME                                  REFS
============================================================================
d0103000     17928    1 ibmtr_cs                          []
d00fe000      6608    2 ds                                [ibmtr_cs]
d00f3000     23408    2 i82365                            []
d00e6000     46848    0 pcmcia_core                       [ibmtr_cs
                                                          ds
```

```
                                                        i82365]
c02ad0e0         0    1 kernel_module                 []
==============================================================================

>> module pcmcia_core
    ADDR      SIZE USED NAME                                REFS
==============================================================================
d00e6000     46848    0 pcmcia_core                   [ibmtr_cs
                                                        ds
                                                        i82365]
==============================================================================

>> module pcmcia_core -f
EXPORTED MODULE SYMBOLS:
==============================================================================
        Module: pcmcia_core
        Number of exported symbols: 15

            ADDR NAME [MODULE]

        d00e6120 register_ss_entry                   [pcmcia_core]
        d00e6290 unregister_ss_entry                 [pcmcia_core]
        d00e8d30 CardServices                        [pcmcia_core]
        d00ecb50 MTDHelperEntry                      [pcmcia_core]
        d00f0788 proc_pccard                         [pcmcia_core]
        d00eb800 request_mem_region                  [pcmcia_core]
        d00eb820 release_mem_region                  [pcmcia_core]
        d00f1618 pci_irq_mask                        [pcmcia_core]
        d00ef090 pci_enable_device                   [pcmcia_core]
        d00ef100 pci_set_power_state                 [pcmcia_core]
        d00e6000 __insmod_pcmcia_core_O/lib/modules/2.2.18/pcmcia/pcmcia_
core.o_M3A6ED7D0_V131602 [pcmcia_core]
        d00e6060 __insmod_pcmcia_core_S.text_L37383  [pcmcia_core]
        d00ef280 __insmod_pcmcia_core_S.rodata_L4779 [pcmcia_core]
        d00f0740 __insmod_pcmcia_core_S.data_L3996   [pcmcia_core]
        d00f16e0 __insmod_pcmcia_core_S.bss_L32      [pcmcia_core]

==============================================================================

>> module kernel_module -f -i 10
EXPORTED MODULE SYMBOLS:
==============================================================================
        Module: kernel_module
        Number of exported symbols: 825

      ADDR NAME [MODULE]
------------------------------------------------------------------------------

0xc027a640 drive_info                           [kernel_module]
0xc023e7c0 boot_cpu_data                        [kernel_module]
0xc023e840 EISA_bus                             [kernel_module]
0xc023e844 MCA_bus                              [kernel_module]
0xc010f224 __verify_write                       [kernel_module]
0xc0107680 dump_thread                          [kernel_module]
0xc010e40c dump_fpu                             [kernel_module]
0xc010e4b8 dump_extended_fpu                    [kernel_module]
0xc010fa1c __ioremap                            [kernel_module]
0xc010fafc iounmap                              [kernel_module]
WARNING: Iteration threshold reached. Current threshold: 10.
         Use "-i" to change threshold.

==============================================================================
```

4.16. namelist

Alias

addtypes,**nmlist**

Usage

```
namelist
        [−a namelist]
        [index_number]
```

Description

```
Add/list opened namelists, i.e. files with type information.

OPTIONS:
 −a namelist
       Add type information of new namelist.
 index_number
       Current namelist is set to given index_number.

If no arguments are given, display all currently opened namelists.
"addtypes" is an alias for "namelist −a".
```

Example 4−11. namelist

For a comprehensive example please refer to Section 5.1.

```
>> namelist
INDEX  NAMELIST
================================================
    0  /boot/Kerntypes
================================================

The current namelist is /boot/Kerntypes (0)

>> namelist −a /tmp/snd.o
/tmp/snd.o is not an object file
The current namelist is /tmp/snd.o (1)

>> namelist
INDEX  NAMELIST
================================================
    0  /boot/Kerntypes
    1  /tmp/snd.o
================================================

The current namelist is /tmp/snd.o (1)

>> namelist 0
The current namelist is /boot/Kerntypes (0)
```

4.17. page

Usage

```
page [−f] [−n] [−w outfile] [page_list]
```

Description

```
Display relevant information from the page struct for each entry in
page_list. Entries in page_list can take the form of a page number
(following a '#') or a virtual address of a page struct in memory. If no
entries are specified, an entry for every page of physical memory will be
displayed.
```

4.18. print

Aliases

p,**pb**, **pd**,**po**,**px**

Usage

```
print [−d] [−o] [−x] [−b] [−w outfile] expression
```

Description

```
Evaluate an expression and print the result. An expression can consist of
numeric values, operators, typedefs, struct/union members, symbols, or a
combination of the above. Following are some examples of valid
expressions:

    (((2*3+4/2)*2+(2/6))/2)

    ((struct task_struct *)0xc5c14000)−>comm

    (*((struct task_struct *)0xc5c14000)−>files.fd).f_flags & 0x8000

The pd command is the same as the print command except that it forces all
integers to be displayed as decimal values.

The px command is the same as the print command except that it forces all
integers to be displayed as hexadecimal values.

The po command is the same as the print command except that it forces all
integers to be displayed as octal values.

The pb command is the same as the print command except that it forces all
integer values to be displayed as binary values. Note that only single
values (numbers, members of structures, etc.) will be displayed in binary
form. Integer values in complex data types such as structures will be
displayed as decimal values.
```

4.19. quit

Aliases

q, **q!**

Usage

```
quit
```

Description

```
Exit lcrash.  Note that q will prompt for confirmation unless a '!' is
appended to the command line.
```

Example 4−12. quit

```
>> q
Do you really want to quit (y to quit) ? n

>> q!
```

4.20. report

Usage

```
report [−w outfile]
```

Description

```
Display a crash dump report. The report contains information about the
system state when the kernel failure occurred.
```

4.21. s390dbf

Platform Dependency

s390, s390x

Usage

```
s390dbf [−w outfile] [−v] [debug_log] [debug_log view]
```

Description

```
Display Debug logs:

+ If called without parameters, all active debug logs are listed.
```

```
+ If called with '−v', all debug views which are available to'lcrash' are
listed.

+ If called with the name of a debug log, all debug−views for whichthe
debug−log has registered are listed. It is possible thatsome of the debug
views are not available to 'lcrash' (see '−v'option).

+ If called with the name of a debug−log and an available viewname,the
specified view is printed.
```

4.22. sizeof

Alias

offset

Usage

```
sizeof
        type | structure.field [...]
        −o structure.field [...]
        [−w outfile]
```

Description

```
Display size of data types in bytes. Additionally display offsets for
members of structs.

OPTIONS:
 type | structure.field [...]
       Print size of types (basic types, structs, typedefs) or
       member of structures in bytes.
 −o structure.field [...]
       Determine the member offset. Only arguments of the form
       'structure.field' are allowed.

To request size for multi−worded types (e.g. "short int") specify the
type whithin "".

Note: An alias "offset" exists for the calling sequence "sizeof −o".
```

Example 4−13. sizeof

```
>> sizeof task_struct module_ref int double
Size of "task_struct": 1152 bytes
Size of "module_ref": 12 bytes
Size of "int": 4 bytes
Size of "double": 8 bytes

>> sizeof mem_map_t page pgd_t
Size of "mem_map_t": 40 bytes
Size of "page": 40 bytes
Size of "pgd_t": 4 bytes
```

```
>> sizeof page.next mem_map_t.index thread_struct.trace −o
Offset: 0 bytes.
Offset: 8 bytes.
Offset: 100 bytes.

>> sizeof "long long unsigned int" "short int" "long double"
Size of "long long unsigned int": 8 bytes
Size of "short int": 2 bytes
Size of "long double": 12 bytes

>> sizeof "short unsigned int" mm_struct.count task_struct −o
ERROR: Could not determine offset for short unsigned int.
Offset: 16 bytes.
ERROR: Could not determine offset for task_struct.
```

4.23. stat

Usage

```
stat [−w outfile]
```

Description

```
Display system statistics and the log_buf array, which contains the
latest messages printed via the kernel printf/cmn_err routines.
```

Example 4−14. stat (s390)

```
>> stat

   sysname : Linux
  nodename : (none)
   release : 2.4.2−0tape−dasd
   version : #7 SMP Mon Apr 30 15:47:23 CEST 2001
   machine : s390
domainname : (none)

LOG_BUF:

    <4>Linux version 2.4.2−0tape−dasd (root@gfree16) (gcc version 2.95.2
 19991024 (release)) #7 SMP Mon Apr 30 15:47:23 CEST 2001
    <4>Command line is: root=/dev/dasda1 ro noinitrd dasd=3e04,3e05,3e00
 cio_msg=yes
    <4>
    <4>We are running native
    <4>This machine has an IEEE fpu
    <4>On node 0 totalpages: 24576
    <4>zone(0): 24576 pages.
    <4>zone(1): 0 pages.
    <4>zone(2): 0 pages.
    <4>Kernel command line: root=/dev/dasda1 ro noinitrd dasd=3e04,3e05,3e00
 cio_msg=yes
    <4>
 ...
```

Example 4−15. stat (i386)

```
>> stat

   sysname : Linux
  nodename : lion28
   release : 2.2.18
   version : #1 Wed Jan 24 12:28:55 GMT 2001
   machine : i686
domainname :

LOG_BUF:

    <4>Linux version 2.2.18 (root@Pentium.suse.de) (gcc version 2.95.2
19991024 (release)) #1 Wed Jan 24 12:28:55 GMT 2001
    <4>BIOS−provided physical RAM map:
    <4> BIOS−e820: 0009f000 @ 00000000 (usable)
    <4> BIOS−e820: 0fef0000 @ 00100000 (usable)
    <4>Detected 696981 kHz processor.
    <4>Console: colour VGA+ 80x25
    <4>Calibrating delay loop... 1389.36 BogoMIPS
    <4>Memory: 256508k/262080k available (1668k kernel code, 408k reserved,
 2968k data, 88k init, 0k bigmem)
...
```

4.24. strace

Platform Dependency

Platform dependent usage and functionality.

Usage on i386

```
strace [−a] [−l] [−f] [−w outfile] [pc sp] stack_addr [level]
```

Description (i386)

```
Displays all complete and unique stack traces (containing level or more
stack frames) from the stack starting at stack_addr. If a level isn't
specified, then each stack trace must have at least three frames to be
considered valid. Alternately, use a specific PC and SP to generate a
stack trace from the stack starting at stack_addr. Or, when the −l
command line option is specified, displays a list of all saved return
addresses contained in the stack starting at stack_addr, along with their
location in the stack and possibly the name of the function called. Or,
if the −a option is specified, display ALL traces of level or more
frames, including invalid traces and duplicate (sub) traces.
```

Usage on s390(x)

```
strace [−f] [−w outfile] stack_addr [level]
```

Description (s390)

```
Displays all complete and unique stack traces (containing level or more
stack frames) from the stack starting at stack_addr. If a level isn't
Specified, then each stack trace must have at least three frames to be
considered valid.
```

Example (s390)

Example 4−16. strace (s390)

```
>> task
    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
================================================================================
  184000      0      0      0      0         0  swapper
================================================================================
1 active task struct found

>> whatis lowcore_ptr
    ADDR  OFFSET  TYPE         NAME
============================================================
  25c484       0  GLOBAL_DATA  lowcore_ptr

>> dump 25c484 10
0x25c484: 00000000 00000000 00000000 00000000 : ................
0x25c494: 00000000 00000000 00000000 00000000 : ................
0x25c4a4: 00000000 00000000                   : ........

>> dump 0x180 16
0x180: 00000000 000100e5 000100e5 00000001 : ................
0x190: 0042ce60 00010000 00000066 00000003 : .B.`.......f....
0x1a0: 00000394 000000e5 ffc4ea0a 0018cc80 : ................
0x1b0: 00000002 800b7f70 800b80ee 00185cd8 : .......p......\.

>> strace 00185cd8
================================================================
TRACE FOR STACK PTR: 0x185cd8

 0 disable_cpu_sync_isc+390 [0xb80ee]
 1 s390_device_recognition_irq+240 [0xb8f80]
 2 s390_device_recognition_all+42 [0xb8fc2]
 3 s390_init_IRQ+192 [0xb5fc0]
 4 init_IRQ+28 [0x1d50ac]
 5 start_kernel+322 [0x1d47d6]
 6 _stext+98 [0x10862]
 7 <back chain invalid>+<ERROR> [0x65bec0]
================================================================
```

4.25. symtab

Usage

```
symtab
        [−l [−f] [symtable]]
        [−r symtable]
```

```
        [-a symtable modulename]
        [-a symtable offset size]
        [-a symtable t_off d_off rd_off b_off t_len d_len rd_len b_len]
        [-w outfile]
```

Description

```
Add/remove/list symbol table information.

OPTIONS:
 -l [symtable]
       List information of (all) symbol table(s).
 -l -f [symtable]
       Show full list of symbols of (all) symbol table(s).
 -a symtable modulename
       Add new symbol table belonging to module modulename.
 -a symtable t_off d_off rd_off b_off t_len d_len rd_len b_len
       Add new symbol table using given segment offsets and lengths
       (off=offset, len=length, t=text, d=data, rd=rodata, b=bss).
 -a symtable offset size
       Add new symbol table using given offset and size.
       Regard size as size of object file corresponding to symtable.
 -r symtable
       Remove symbol table.
 -a __ksymtab__
 -r __ksymtab__
 -l [-f] __ksymtab__
       Add, remove or list table of exported kernel symbols.

You can use only one of the above command lines at the same time.
```

Example 4-17. symtab

For a comprehensive example please refer to Section 5.1.

4.26. task

Alias

ps

Usage

```
task [-f] [-n] [-w outfile] [task list]
```

Description

```
Display relevant information for each entry in task_list. If no entries
are specified, display information for all active tasks. Entries in
task_list can take the form of a virtual address or a PID (following a
'#').
```

Example 4−18. task

```
>> task
ACTIVE TASKS:

    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
================================================================================
c02ca000      0      0      0      0         0  swapper
cff3c000      0      1      0      1       100  init
cff28000      0      2      1      1        40  kflushd
cff26000      0      3      1      1        40  kupdate
cff24000      0      4      1      1       840  kswapd
cfd7a000      0      5      1      1        40  mdrecoveryd
cecea000      0    170      1      1       140  cardmgr
cc15c000      0    229      1      1       140  syslogd
cbfa6000      0    231      1      1       140  sshd
cc1b0000      0    234      1      1       140  klogd
cb9b6000      0    245      1      1       140  lpd
    ...
ca6b0000   4640   3306    433      1         0  xosview.bin
c9810000   4640   3309      1      1         0  xeyes
c02e0000   4640   3312      1      1         0  xclock
c5e6c000   4640   3314    433      1         0  FvwmPager
c657e000      0   3321    356      4        44  cron
================================================================================
57 active task structs found

>> task −f c02e0000
    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
================================================================================
c02e0000   4640   3312      1      1         0  xclock

  MM:0xc97e7cc0

THREAD:
  ESP0:0xc02e2000, ESP:0xc02e1ea8, EIP:0xc0113286
  FS:0x0, GS:0x0

================================================================================
1 active task struct found
```

4.27. trace

Alias

t

Usage

```
trace [−a] [−f] [−w outfile] [[task_list] | [−t tracerec_list]
```

Description

```
Displays a stack trace for each task included in task_list. If task_list
is empty and deftask is set, then a stack trace for the default task is
displayed. If deftask is not set, then a trace will be displayed for the
task running at the time of a system PANIC. If the command is issued with
```

```
the −t command line option, additional items on the command line will be
treated as pointers to lcrash stack trace records (prevously allocated
using the mktrace command).
```

Example 4−19. trace

```
>> task
ACTIVE TASKS:

    ADDR    UID    PID   PPID  STATE     FLAGS  NAME
============================================================================
  18e000      0      0      0      0         0  swapper
  5b0000      0      1      0      1       100  init
  5a8000      0      2      1      1        40  kmcheck
  59a000      0      3      1      1        40  keventd
  57c000      0      4      1      1       840  kswapd
  57a000      0      5      1      1       840  kreclaimd
  578000      0      6      1      1        40  bdflush
  576000      0      7      1      1        40  kupdated
 6edc000      0    231      3      1        40  keventd
 6ed0000      1    287      1      1       140  portmap
 6e60000      0    349      1      1        40  syslogd
 779a000      0    363      1      1       140  klogd
 6d54000      0    401      1      1       140  inetd
 6a0a000    100    448      1      1        40  xfs
 7ac0000      0    467      1      1         0  sulogin
 6948000      0    468    401      1       100  in.telnetd
 68f8000      0    469    468      1       100  login
 67e4000      0    470    469      1       100  bash
 61c8000      0    534    470      0       100  lcrash
============================================================================
19 active task structs found

>> trace 67e4000
================================================================
STACK TRACE FOR TASK: 0x67e4000 (bash)

 STACK:
 0 schedule+1076 [0x1c590]
 1 sys_wait4+1050 [0x23fc6]
 2 pgm_system_call+34 [0x130d0]
================================================================

>> trace 67e4000 −f
================================================================
STACK TRACE FOR TASK: 0x67e4000 (bash)

 STACK:
 0 schedule+1076 [0x1c590]

   SP=0x67e5de8, FP=0x67e5e48, SIZE=144

   67e5de8: 067e5e78  00525164  077e4000  077e4000
   67e5df8: 067e5ea8  07000000  067e4000  00000001
   67e5e08: 005bc000  067e4000  00000000  067e5de8
   67e5e18: 00525120  8001c164  8001c590  067e5de8
   67e5e28: 00525120  067e5f68  00000004  070de000
   67e5e38: 00479000  067e4000  0052513c  80011038
   67e5e48: 070112cc  077e5e10  0401d000  0690a000
```

```
   67e5e58: 067e5f68  00000001  00000000  00000000
   67e5e68: 00000000  067e5e6c  001ea030  00000004

 1 sys_wait4+1050 [0x23fc6]

   SP=0x67e5e78, FP=0x67e5ed8, SIZE=144

   67e5e78: 067e5f08  800f8da0  067e5f08  067e5e48
   67e5e88: 067e5e6c  00000215  00000002  001ea010
   67e5e98: 7ffff7fc  00000000  ffffffff  fffffe00
   67e5ea8: 00000000  80023bb4  80023fc6  067e5e78
   67e5eb8: ffffffea  00000020  0048a668  067e4000
   67e5ec8: 00000000  000000ff  7ffff6c8  06df1ae0
   67e5ed8: 00000000  067e4000  067e40b0  067e40b0
   67e5ee8: 00000000  067e4000  00000000  00000000
   67e5ef8: 067e4000  00000000  7ffff800  067e4000

 2 pgm_system_call+34 [0x130d0]

   SP=0x67e5f08, FP=0x67e5f68, SIZE=248

   67e5f08: 00000000  00000000  00000000  00000000
   67e5f18: 00000000  00000000  0048a668  00402c0c
   67e5f28: 00023bac  067e4000  00422e74  7ffff798
   67e5f38: c015f2d4  00013000  800130d0  067e5f08
   67e5f48: 00000000  00000000  00000000  00000000
   67e5f58: 00000000  00000000  00000000  00000000
   67e5f68: 070dd000  c00e357a  00000000  400e3578
   67e5f78: ffffffff  7ffff7fc  00000002  00000000
   67e5f88: 0048a668  00402c0c  00000001  00000000
   67e5f98: 00422e74  7ffff798  c015f2d4  c00e3504
   67e5fa8: c00e352a  7ffff738  00000000  00000000
   67e5fb8: 00000000  00000000  00000001  00000000
   67e5fc8: 00000000  00000000  00000000  00000000
   67e5fd8: 00000000  00000000  00000000  00000000
   67e5fe8: 00000000  00000000  ffffffff  00000020
   67e5ff8: 01ffffff  00000000

================================================================

>> trace 61c8000
================================================================
TASK HAS CPU (1): 0x61c8000 (lcrash):
No valid lowcore info available ?
 LOWCORE INFO:
  -psw      : 0x07080000 0x8001b0de
  -function : do_machine_power_off+142
  -prefix   : 0x005bf000
  -cpu timer: 0xffff879f 0x5a597b00
  -clock cmp: 0xb5eabdb9 0x7a80ea00
  -general registers:
     00190654 00000000 00000000 00000000
     0026350c 00000009 00000004 00000001
     0002cab8 04674000 00400af0 04674000
     04674000 8001b058 8001b080 04675ce8
  -access registers:
     00000000 00000000 00000000 00000000
     00000001 00000000 00000000 00000000
     00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000
  -control registers:
     14b52a02 0026107f 00000000 00000000
```

```
     00000000 00000000 11000000 846661ff
     00000000 00000000 00000000 00000000
     00000000 846661ff d0000000 00000000
  −floating point registers 0,2,4,6:
     0000000000000000 0000000000000000
     0000000000000000 0000000000000000

================================================================
```

4.28. unload

Usage

```
unload filename|directory
```

Description

```
Unload a file or a directory. In the case of a directory, all files in
that directory will be unloaded.
```

4.29. vi

Usage

```
vi function_name | −f sial_file_name
```

Description

```
Start a vi session of a sial file or a sial function in particular.
```

4.30. vtop

Usage

```
vtop [−m map_pointer] [−w outfile] vaddr_list
```

Description

```
Display the virtual to physical memory mapping for each entry in
vaddr_list. Entries in addr_list can in the form of a physical address,
virtual address, or page number (following a '#'). When the −m command
line option is specified, treat the accompanying parameter as an
mm_struct pointer to use when determining memory mapping.
```

Example 4−20. vtop

```
>> dump 0xd002cfe0 −B 60
0xd002cfe0: 73 6e 64 5f 70 63 69 5f 63 6f 6d 70 61 74 5f 66 : snd_pci_compat_f
```

```
0xd002cff0: 69 6e 64 5f 63 61 70 61 62 69 6c 69 74 79 00 73 : ind_capability.s
0xd002d000: 6e 64 5f 70 63 69 5f 63 6f 6d 70 61 74 5f 64 6d : nd_pci_compat_dm
0xd002d010: 61 5f 73 75 70 70 6f 72 74 65 64 00             : a_supported.

>> vtop 0xd002cfe0
   VADDR     KADDR     PADDR       PFN
============================================================
d002cfe0  cef42fe0   ef42fe0     61250
============================================================

>> dump ef42fe0 -B 60
0xef42fe0: 73 6e 64 5f 70 63 69 5f 63 6f 6d 70 61 74 5f 66 : snd_pci_compat_f
0xef42ff0: 69 6e 64 5f 63 61 70 61 62 69 6c 69 74 79 00 73 : ind_capability.s
0xef43000: c4 20 83 c4 fc 6a 01 68 73 c8 02 d0 53 e8 a6 fe : . ...j.hs...S...
0xef43010: ff ff 83 c4 fc 6a 02 68 81 c8 02 d0             : .....j.h....

>> vtop 0xd002d000
   VADDR     KADDR     PADDR       PFN
============================================================
d002d000  cef41000   ef41000     61249
============================================================

>> dump ef41000 -B 28
0xef41000: 6e 64 5f 70 63 69 5f 63 6f 6d 70 61 74 5f 64 6d : nd_pci_compat_dm
0xef41010: 61 5f 73 75 70 70 6f 72 74 65 64 00             : a_supported.
```

Example 4-21. vtop

```
>> whatis init_mm
    ADDR  OFFSET  TYPE         NAME
============================================================
c02a90a0       0  GLOBAL_DATA  init_mm

>> whatis module_list
    ADDR  OFFSET  TYPE         NAME
============================================================
c02ad128       0  GLOBAL_DATA  module_list

>> dump c02ad128
0xc02ad128: d0103000                            : .0..

>> vtop -m c02a90a0 d0103000
   VADDR     KADDR     PADDR       PFN
============================================================
d0103000  cec99000   ec99000     60569
============================================================

>> print ((module*)0xec99000)->name
0xd0106a26

>> vtop -m c02a90a0 0xd0106a26
   VADDR     KADDR     PADDR       PFN
============================================================
d0106a26  cec96a26   ec96a26     60566
============================================================

>> print (char*) ec96a26
0xec96a26 "ibmtr_cs"
```

4.31. walk

Usage

```
walk
        [−l]
        struct field|offset addr [−f] [−n] [−h n|p]
        struct field|offset addr −s [−h n|p]
        struct field|offset addr −h n|p −t
        address offset size
        [−i iteration_threshold]
        [−w outfile]
```

Description

```
Walk a linked list of kernel structures or memory blocks.

OPTIONS:
 −l
       Show a list of special structures, which can be displayed in a
       predefined formatted manner.
       Currently there is support for a handful special structures.
 struct field|offset addr [−f] [−n] [−h n|p]
       Display  each entry of a linked list of special structures in
       a predefined formatted way.
       By default, the output consists of one line for each structure.
       Using '−f' and/or '−n' a more detailed output is given.
       '−f' can be used for all special structures. '−n' works for
       special structures "mm_struct" and "task_struct".
 struct field|offset addr −s [−h n|p]
       Each structure of a linked list is displayed in its entirety −
       in a C−like format. All structures for which type information is
       available can be displayed in this manner.
 −h n|p
       A linked list is constructed by following "list_head" structures
       instead of next pointers. The argument specifies wether to follow
       the next pointers of struct list_head (using 'n') or to follow
       the prev pointers of struct list_head (by using 'p').
       'field' or 'offset' is regarded as a member of type "list_head"
       instead of a next pointer within the 'struct'. 'addr' is
       interpreted as a pointer to an anchor of a linked list of
       "struct list_head" structures.
 struct field|offset addr −h n|p −t
       Display each entry of a linked "list_head"−list in one line.
       For each entry the address to the 'struct' structure, the
       address to the "list_head" member within 'struct', and previous
       and next pointer of the embedded "list_head" are given.
 address offset size
       Do a hex memory dump of each structure in a list.
       A start address ('address') of a structure, a byte offset
       ('offset') for the next pointer in the structure, and a
       structure size ('size') are required. 'size' bytes will be
       dumped for each entry in the constructed list.
 −i iteration_threshold
       By default, certain loops are interrupted after 10'000 iterations
       to avoid endless loops while following invalid pointers. Using
       this option you can change the threshold for the current command.
       A value '0' means infinite iteration threshold, i.e. no
```

```
       interruption of the loop is caused by reaching any threshold.

While using "struct field|offset addr" without '−h', a structure name
('struct'), a field name ('field') or byte offset ('offset') for the next
pointer within the structure, and a pointer ('addr') to the first entry
of the linked list must be given.

Note: Using '−h' the anchor is not displayed as a structure 'struct'.
```

Example 4−22. walk

```
>> module
    ADDR      SIZE USED NAME                                    REFS
============================================================================
d0103000     17928    1 ibmtr_cs                             []
d00fe000      6608    2 ds                                   [ibmtr_cs]
d00f3000     23408    2 i82365                               []
d00e6000     46848    0 pcmcia_core                          [ibmtr_cs
                                                             ds
                                                             i82365]
c02ad0e0         0    1 kernel_module                        []
============================================================================

>> print ((module*) d00e6000)−>refs
0xd0106b80

>> walk −s module_ref next_ref 0xd0106b80
struct module_ref {
        dep = 0xd00e6000
        ref = 0xd0103000
        next_ref = 0xd00ff9bc
}
struct module_ref {
        dep = 0xd00e6000
        ref = 0xd00fe000
        next_ref = 0xd00f8b38
}
struct module_ref {
        dep = 0xd00e6000
        ref = 0xd00f3000
        next_ref = (nil)
}
```

Example 4−23. walk

```
>> findsym inode_unused
      ADDR  OFFSET  TYPE         NAME
============================================================
0xc0243e48       0  LOCAL_DATA   inode_unused
============================================================
1 symbol found

>> walk list_head next 0xc0243e48 −h n −t
STRUCT ADDR       PREV   LISTHEAD       NEXT
============================================
          0 0xc2faca48 0xc0243e48 0xc4d8d340
 0xc4d8d340 0xc0243e48 0xc4d8d340 0xc416ef68
```

```
 0xc416ef68 0xc4d8d340 0xc416ef68 0xc7ab55d0
 0xc7ab55d0 0xc416ef68 0xc7ab55d0 0xc3244298
 0xc3244298 0xc7ab55d0 0xc3244298 0xc328c3e0
 0xc328c3e0 0xc3244298 0xc328c3e0 0xc3baf0b0
   ...
 0xc32767b8 0xc3276cd8 0xc32767b8 0xc7ab50b0
 0xc7ab50b0 0xc32767b8 0xc7ab50b0 0xc79e7af0
 0xc79e7af0 0xc7ab50b0 0xc79e7af0 0xc3289af0
 0xc3289af0 0xc79e7af0 0xc3289af0 0xc32623e0
 0xc32623e0 0xc3289af0 0xc32623e0 0xc31f2150
 0xc31f2150 0xc32623e0 0xc31f2150 0xc314b0b0
 0xc314b0b0 0xc31f2150 0xc314b0b0 0xc2ff3c38
 0xc2ff3c38 0xc314b0b0 0xc2ff3c38 0xc2fd2528
 0xc2fd2528 0xc2ff3c38 0xc2fd2528 0xc2faca48
 0xc2faca48 0xc2fd2528 0xc2faca48 0xc0243e48
==========================================

>> walk inode i_list 0xc0243e48 -h n -t
STRUCT ADDR       PREV   LISTHEAD       NEXT
==========================================
          0 0xc2faca48 0xc0243e48 0xc4d8d340
 0xc4d8d338 0xc0243e48 0xc4d8d340 0xc416ef68
 0xc416ef60 0xc4d8d340 0xc416ef68 0xc7ab55d0
 0xc7ab55c8 0xc416ef68 0xc7ab55d0 0xc3244298
 0xc3244290 0xc7ab55d0 0xc3244298 0xc328c3e0
 0xc328c3d8 0xc3244298 0xc328c3e0 0xc3baf0b0
   ...
 0xc32767b0 0xc3276cd8 0xc32767b8 0xc7ab50b0
 0xc7ab50a8 0xc32767b8 0xc7ab50b0 0xc79e7af0
 0xc79e7ae8 0xc7ab50b0 0xc79e7af0 0xc3289af0
 0xc3289ae8 0xc79e7af0 0xc3289af0 0xc32623e0
 0xc32623d8 0xc3289af0 0xc32623e0 0xc31f2150
 0xc31f2148 0xc32623e0 0xc31f2150 0xc314b0b0
 0xc314b0a8 0xc31f2150 0xc314b0b0 0xc2ff3c38
 0xc2ff3c30 0xc314b0b0 0xc2ff3c38 0xc2fd2528
 0xc2fd2520 0xc2ff3c38 0xc2fd2528 0xc2faca48
 0xc2faca40 0xc2fd2528 0xc2faca48 0xc0243e48
==========================================

>> findsym inode_in_use
      ADDR  OFFSET  TYPE         NAME
============================================================
0xc0243e40       0  GLOBAL_DATA  inode_in_use
============================================================
1 symbol found

>> walk inode i_list 0xc0243e40 -h n -t -i 5
STRUCT ADDR       PREV   LISTHEAD       NEXT
==========================================
          0 0xcff38008 0xc0243e40 0xc5501c38

WARNING: Previous pointer broken. PREV: 0xc579c3e0, SHOULD BE: 0xc0243e40
 0xc5501c30 0xc579c3e0 0xc5501c38 0xc6314f68
 0xc6314f60 0xc5501c38 0xc6314f68 0xc2c44e20
 0xc2c44e18 0xc6314f68 0xc2c44e20 0xc8671340
 0xc8671338 0xc2c44e20 0xc8671340 0xc54da528
 0xc54da520 0xc8671340 0xc54da528 0xcbde6528

WARNING: Iteration threshold reached. Current threshold: 5.
         Use "-i" to change threshold.
```

Example 4−24. walk

```
>> module
      ADDR      SIZE USED NAME                            REFS
========================================================================
0xd00f6000     17928    1 ibmtr_cs                        []
0xd00f1000      6608    2 ds                              [ibmtr_cs]
0xd00e6000     23408    2 i82365                          []
   ...
0xd002b000     27168    0 snd−ac97−codec                  [snd−cs461x]
0xd0023000     28624    0 snd−mixer                       [snd−ac97−codec]
0xd0017000     43632    1 snd                             [snd−seq−midi
                                                          snd−seq−midi−event
                                                          snd−seq
                                                          snd−card−cs461x
                                                          snd−cs461x
                                                          snd−pcm
                                                          snd−timer
                                                          snd−rawmidi
                                                          snd−seq−device
                                                          snd−ac97−codec
                                                          snd−mixer]
0xd0015000      2576    2 soundcore                       [snd]
0xc0241980         0    1 kernel_module                   []
========================================================================

>> sizeof module
Size of "module": 72 bytes

>> offset module.next
Offset: 4 bytes.

>> walk 0xd002b000 4 72
Dumping 72 byte block at 0xd002b000:

0xd002b000: 00000060 d0023000 d00314c9 00006a20 : `....0...... j..
0xd002b010: 00000000 00000011 0000000a 00000002 : ................
0xd002b020: d00315a0 d0031a08 d0058134 d0030350 : ........4...P...
0xd002b030: d003035c 00000000 00000000 00000000 : \................
0xd002b040: 00000000 00000000                   : ........

Dumping 72 byte block at 0xd0023000:

0xd0023000: 00000060 d0017000 d0029cc1 00006fd0 : `....p.......o..
0xd0023010: 00000000 00000019 00000035 00000001 : ........5.......
0xd0023020: d0029d78 d0029fc4 d0031a08 d00266b4 : x............f..
0xd0023030: d00266c0 d00296e0 d00297e8 00000000 : .f..............
0xd0023040: 00000000 00000000                   : ........

Dumping 72 byte block at 0xd0017000:

0xd0017000: 00000060 d0015000 d00200c1 0000aa70 : `....P......p...
0xd0017010: 00000001 00000019 0000005f 00000001 : ........_.......
0xd0017020: d0020170 d0021a60 d0080fd0 d0017ba4 : p...`........{..
0xd0017030: d0017bb0 d001f8a4 d001f8fc 00000000 : .{..............
0xd0017040: 00000000 00000000                   : ........

Dumping 72 byte block at 0xd0015000:

0xd0015000: 00000060 c0241980 d0015825 00000a10 : `.....$.%X......
0xd0015010: 00000002 00000019 00000010 00000000 : ................
0xd0015020: d00158f8 00000000 d0021a60 d001545c : .X......`...\T..
```

```
0xd0015030: d0015440 00000000 00000000 00000000 : @T..............
0xd0015040: 00000000 00000000                   : ........

Dumping 72 byte block at 0xc0241980:

0xc0241980: 00000048 00000000 c0205380 00000000 : H........S .....
0xc0241990: 00000001 00000001 00000339 00000000 : ........9.......
0xc02419a0: c0233958 00000000 00000000 00000000 : X9#.............
0xc02419b0: 00000000 c0232aa0 c0233958 00000000 : .....*#.X9#.....
0xc02419c0: 00000000 00000000                   : ........

5 blocks in linked list
```

4.32. whatis

Usage

```
whatis [−a] [−f] [−l] [−n] [−w outfile] expression
```

Description

```
Display, in C−like fashion, detailed information about kernel types
(structs, unions, typedefs, base types, etc.) If the −a option is
specified, display a list of all types. If the −l option is specified,
display type information in tabular form. When the −f option is
specified, along with the −l option, display additional information about
the type. If the −n option is specified for a struct or union, along with
the −l option, display information about each member.
```

Note

For display of information for multi−worded types (e.g. "short int") you have to use parenthesis around the type.

Example 4−25. whatis

```
>> whatis mem_map
    ADDR  OFFSET  TYPE         NAME
============================================================
c02addec       0  GLOBAL_DATA  mem_map

>> whatis (short unsigned int) −l
NAME                      TYPE             TYPE_NUM         REAL_TYPE  SIZE
=================================================================================
short unsigned int        BASE     0001000000000009  0000000000000000    2
=================================================================================
1 type found

>> whatis page
struct page {
        struct page *next;
        struct page *prev;
        pgoff_t index;
        struct inode *inode;
        struct page *next_hash;
```

```
        atomic_t count;
        long unsigned int flags;
        struct wait_queue *wait;
        struct page **pprev_hash;
        struct buffer_head *buffers;
};

>> whatis page.index
pgoff_t

>> whatis pgoff_t
long unsigned int

>> whatis -l
NAME                        TYPE             TYPE_NUM         REAL_TYPE  SIZE
================================================================================
page                        STRUCT   0001002300000014  0000000000000000   40
================================================================================
1 type found

>> whatis page.index -l -f
NAME                        TYPE             TYPE_NUM         REAL_TYPE  SIZE
================================================================================
long unsigned int           BASE     0001000000000005  0000000000000000    4
  ST_BIT_OFFSET=0, ST_BIT_SIZE=0
  ELEMENT_TYPE=0x0, INDEX_TYPE=0x1000000000005, VALUE=0
  FLAGS=0x2, OFFSET=0
  TYPESTR="long unsigned int "
  LOW_BOUNDS=0, HIGH_BOUNDS=-1, MEMBER=0x0, NEXT=0x0

================================================================================
1 type found

>> whatis -a -l
FileVersion                 TYPEDEF  0001004e00000007  0001000900000017    0
PioctlData                  STRUCT   0001004e00000049  0000000000000000   20
Unique_t                    TYPEDEF  0001004e00000006  0001000900000017    0
...
loff_t                      TYPEDEF  000100090000000d  0001000c00000013    0
long double                 BASE     000100000000000e  0000000000000000   12
long int                    BASE     0001000000000003  0000000000000000    4
long long int               BASE     0001000000000006  0000000000000000    8
long long unsigned int      BASE     0001000000000007  0000000000000000    8
long unsigned int           BASE     0001000000000005  0000000000000000    4
machine_type                ENUM     0001004900000001  0000000000000000    0
mem_map_t                   TYPEDEF  0001000200000016  0001002300000014    0
...
task_struct                 STRUCT   0001002500000002  0000000000000000 1424
task_union                  UNION    0001000300000014  0000000000000000 8192
tcflag_t                    TYPEDEF  0001007b00000003  0001000000000004    0
termio                      STRUCT   0001007a00000002  0000000000000000   18
...
void                        BASE     0001000000000013  0001000000000013   -1
vuid_t                      TYPEDEF  0001004e0000000a  0001000900000020    0
wait_queue                  STRUCT   0001001c00000003  0000000000000000   12
wait_queue_head_t           TYPEDEF  0001002500000004  0001001c00000002    0
wait_queue_t                TYPEDEF  0001002500000003  0001001c00000003    0
winsize                     STRUCT   0001007a00000001  0000000000000000    8
================================================================================
491 types found
```

Chapter 5. Sample lcrash Sessions

5.1. Analyze Kernel Modules

This session should describe how to use lcrash in analyzing kernel modules. First of all we make use of lcrash commands **namelist** and **symtab**.

We have a kernel module `my_dummy.o` containing a locale variable *`DUMMY`* of type dummy_t. The corresponding code fragment is as follows:

```
typedef struct dummy_s{
        int member1;
        char *member2;
        struct dummy_s *member3;
} dummy_t;

static dummy_t DUMMY={0, "just a demonstration", &DUMMY};
```

Our intention will be to examine this local data with lcrash. To make it little more tricky we analyze a live dump and the module will be loaded while lcrash is running.

Our module was compiled using **gcc** option `-gstabs` to create type information. The symbol table of the module was generated using a command line like **nm `my_dummy.o` > `/tmp/my_dummy.map`**.

The file `my_dummy.o` was also copied to `/tmp`.

1. Start lcrash.

```
bash# lcrash /boot/System.map-2.2.18 /dev/mem /boot/Kerntypes
map = /boot/System.map-2.2.18, vmdump = /dev/mem, outfile = stdout, kerntypes =
 /boot/Kerntypes

Please wait...
        Loading system map ..... Done.
        Loading type info (Kerntypes) ... Done.
        Loading ksyms from dump ........ Done.
>>
```

2. Look what modules are loaded.

```
>> module
    ADDR      SIZE USED NAME                                    REFS
=============================================================================
d0103000     17928    1 ibmtr_cs                          []
d00fe000      6608    2 ds                                [ibmtr_cs]
d00f3000     23408    2 i82365                            []
d00e6000     46848    0 pcmcia_core                       [ibmtr_cs
                                                          ds
                                                          i82365]
c02ad0e0         0    1 kernel_module                     []
=============================================================================
```

3. From another shell, load module *my_dummy*.

```
bash# insmod my_dummy.o
bash#
```

4. Verify the former action with lcrash.

```
>> module
    ADDR      SIZE USED NAME                                  REFS
==========================================================================
d0000000      1120    0 my_dummy                           []
d0103000     17928    1 ibmtr_cs                           []
d00fe000      6608    2 ds                                 [ibmtr_cs]
d00f3000     23408    2 i82365                             []
d00e6000     46848    0 pcmcia_core                        [ibmtr_cs
                                                           ds
                                                           i82365]
c02ad0e0         0    1 kernel_module                      []
==========================================================================
```

5. Look which symbols of the new module are exported.

```
>> module -f my_dummy
EXPORTED MODULE SYMBOLS:
==========================================================================
        Module: my_dummy
        Number of exported symbols: 6

            ADDR NAME [MODULE]

        d0000000 __insmod_my_dummy_O/home/aherrman/CPP/crash_ex/my_dummy.o_
M3B1CDF3B_V131602 [my_dummy]
        d0000060 dummy_init                           [my_dummy]
        d0000060 __insmod_my_dummy_S.text_L447        [my_dummy]
        d000021f __insmod_my_dummy_S.rodata_L29       [my_dummy]
        d000041c __insmod_my_dummy_S.bss_L16          [my_dummy]
        d0000240 __insmod_my_dummy_S.data_L260        [my_dummy]

==========================================================================
```

6. Load type information of the module.

```
>> namelist -a /tmp/my_dummy.o
.The current namelist is /tmp/my_dummy.o (1)
>> namelist
INDEX  NAMELIST
================================================
    0  /boot/Kerntypes
    1  /tmp/my_dummy.o
================================================

The current namelist is /tmp/my_dummy.o (1)
```

7. Load symbol table of the module.

```
>> symtab -a /tmp/my_dummy.map my_dummy
```

```
Adding symbol table.
           filename: /tmp/my_dummy.map
        text_offset:        0
        data_offset:        0
      rodata_offset:        0
         bss_offset:        0
        module size:     1120
..Done.
```

Something went wrong, offsets of text and data sections of the module should not be zero. This is caused by the fact, that we added our module after lcrash was started. We have to remove the loaded symbol table and we have to recreate the table *__ksymtab__*.

8. Remove our new symbol table and *__ksymtab__*.

```
>> symtab -l
Loaded symbol tables:
========================================================================
#SYMS:     7803 /boot/System.map-2.2.18
 TEXT:        0 DATA:        0 RODATA:        0 BSS:        0
#SYMS:     1163 __ksymtab__
 TEXT:        0 DATA:        0 RODATA:        0 BSS:        0
#SYMS:       14 /tmp/my_dummy.map [my_dummy]
 TEXT:        0 DATA:        0 RODATA:        0 BSS:        0
========================================================================
>> symtab -r /tmp/my_dummy.map
Removing symbol table.
Done.
>> symtab -r __ksymtab__
Removing symbol table.
Done.
```

9. Recreate symbol table *__ksymtab__*.

```
 >> symtab -a __ksymtab__
Adding symbol table.

        Loading ksyms from dump ........
Done.
```

10. Load our new symbol table again.

```
>> symtab -a /tmp/my_dummy.map my_dummy
Adding symbol table.
           filename: /tmp/my_dummy.map
        text_offset: d0000060
        data_offset: d0000240
      rodata_offset: d000021f
         bss_offset: d000041c
        module size:     1120
..Done.
>> symtab -l
Loaded symbol tables:
========================================================================
#SYMS:     7803 /boot/System.map-2.2.18
 TEXT:        0 DATA:        0 RODATA:        0 BSS:        0
#SYMS:     1169 __ksymtab__
```

```
 TEXT:        0 DATA:        0 RODATA:        0 BSS:        0
#SYMS:       14 /tmp/my_dummy.map [my_dummy]
 TEXT: d0000060 DATA: d0000240 RODATA: d000021f BSS: d000041c
================================================================================
```

11. Look which symbols are available in module *my_dummy*.

```
>> symtab -l -f /tmp/my_dummy.map
================================================================================
#SYMS:       14 /tmp/my_dummy.map [my_dummy]
 TEXT: d0000060 DATA: d0000240 RODATA: d000021f BSS: d000041c

    ADDR  OFFSET  TYPE         NAME
----------------------------------------------------------
d0000060       0  GLOBAL_TEXT  dummy_init
d00000f0       0  LOCAL_TEXT   dummy_xmit
d0000130       0  LOCAL_TEXT   dummy_get_stats
d0000140       0  LOCAL_TEXT   dummy_open
d0000160       0  LOCAL_TEXT   dummy_close
d0000180       0  LOCAL_TEXT   set_multicast_list
d0000190       0  LOCAL_TEXT   dummy_probe
d00001b0       0  GLOBAL_TEXT  init_module
d00001f0       0  GLOBAL_TEXT  cleanup_module
d000021f       0  LOCAL_TEXT   Letext
d0000240       0  LOCAL_DATA   DUMMY
d0000260       0  LOCAL_DATA   dev_dummy
d000041c       0  LOCAL_DATA   dummy_name
d00004c0       0  ABS          /tmp/my_dummy.map_END
----------------------------------------------------------
================================================================================
```

12. Try to examine the local variable *DUMMY* of our module.

```
>> whatis DUMMY
    ADDR  OFFSET  TYPE         NAME
============================================================
d0000240       0  LOCAL_DATA   DUMMY
>> whatis dummy_t
struct dummy_s
struct dummy_s {
        int member1;
        char *member2;
        struct dummy_s *member3;
};
>> print *(dummy_t*) d0000240
struct dummy_s {
        member1 = 0
        member2 = 0xd000021f
        member3 = 0xd0000240
}

>> whatis dummy_s.member2
char *
>> print (char*) 0xd000021f
0xd000021f "just a demonstration"
```

Furthermore an additional symbol table of a kernel module provides you function names when setting up stack back−traces with **trace** or **strace** and when using disassembling routine **dis**.

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Sect1" is a named appendix or a front−matter sect1 of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Sect1 may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sect1s" are certain Secondary Sect1s whose titles are designated, as being those of Invariant Sect1s, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front−Cover Texts or Back−Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard−conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine−generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in sect1 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front−Cover Texts on the front cover, and Back−Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine−readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly−accessible computer−network location containing a complete Transparent copy of the Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. If you use the latter option, you must

take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sect1s 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History sect1 of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sect1s and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the sect1 entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no sect1 entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" sect1. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. In any sect1 entitled "Acknowledgements" or "Dedications", preserve the sect1's title, and preserve in the sect1 all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sect1s of the Document, unaltered in their text and in their titles. Sect1 numbers or the equivalent are not considered part of the sect1 titles.
M. Delete any sect1 entitled "Endorsements". Such a sect1 may not be included in the Modified Version.
N. Do not retitle any existing sect1 as "Endorsements" or to conflict in title with any Invariant Sect1.

If the Modified Version includes new front−matter sect1s or appendices that qualify as Secondary Sect1s and contain no material copied from the Document, you may at your option designate some or all of these sect1s as invariant. To do this, add their titles to the list of Invariant Sect1s in the Modified Version's license notice.

These titles must be distinct from any other sect1 titles.

You may add a sect1 entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties−−for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front−Cover Text, and a passage of up to 25 words as a Back−Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front−Cover Text and one of Back−Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in sect1 4 above for modified versions, provided that you include in the combination all of the Invariant Sect1s of all of the original documents, unmodified, and list them all as Invariant Sect1s of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sect1s may be replaced with a single copy. If there are multiple Invariant Sect1s with the same name but different contents, make the title of each such sect1 unique by adding at the end of it, in parentheses, the name of the original author or publisher of that sect1 if known, or else a unique number. Make the same adjustment to the sect1 titles in the list of Invariant Sect1s in the license notice of the combined work.

In the combination, you must combine any sect1s entitled "History" in the various original documents, forming one sect1 entitled "History"; likewise combine any sect1s entitled "Acknowledgements", and any sect1s entitled "Dedications". You must delete all sect1s entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an

"aggregate", and this License does not apply to the other self−contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of sect1 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of sect1 4. Replacing Invariant Sect1s with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sect1s in addition to the original versions of these Invariant Sect1s. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sect1s being

> LIST THEIR TITLES, with the Front−Cover Texts being LIST, and with the Back−Cover Texts being LIST. A copy of the license is included in the sect1 entitled "GNU Free Documentation License".

If you have no Invariant Sect1s, write "with no Invariant Sect1s" instead of saying which ones are invariant. If you have no Front−Cover Texts, write "no Front−Cover Texts" instead of "Front−Cover Texts being LIST"; likewise for Back−Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliography

[stabs1997] Julia Menapace, Jim Kingdon, and David MacKenzie, 1992−2001, Cygnus Support, *The "stabs" debug format* .

Notes

[1] In fact **lcrash** uses only type information contained in the "stabs" format. Further debug information of this format is not used. For more information about the "stabs" format please refer to [stabs1997].