

Lcrash HOWTO

Andreas Herrman

<aherrman@de.ibm.com>

Copyright © 2001, 2002 IBM Deutschland Entwicklung GmbH, IBM Corporation

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](#).

Table of Contents

Chapter 1.

Table of Contents

<u>whatis</u>	38
<u>Chapter 5. Sample lcrash Sessions</u>	41
<u>Analyze Kernel Modules</u>	41
<u>0. PREAMBLE</u>	46
<u>1. _____</u>	

Chapter 1. Introduction

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](#)

Chapter 2. Installation

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](#).
-

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 now 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/`.

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`.

LKCD CVS Repository

The current code of LKCD and hence the newest lcrash sources are located at [sourceforge](http://sourceforge.net).

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.)

```
Dash$ cvs -d:pserver:anonymous@cvs.lkcd.sourceforge.net:/cvsroot/lkcd login  
(Logging in to anonymous@cvs.lkcd.sourceforge.net)
```

Chapter 3. General Usage

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**

Ctrl-R	redraw input line
Esc-F	forward one word
Esc-B	backward one word

Chapter 4. Lcrash Command Reference

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 lose 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

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
syntab	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

When using **lcrash_command -w *filename*, lcrash** 4s1 Tf (hds the output of the executed and)Tj to the/F4 -968 0

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

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 preceded by a pound sign (`#`), it will be treated as a page number (PFN).

Note

Output of `dump` command depends on endianness of the host platform. E.g. on i386 `lcrash` will show words, half-words and double-words in little endianness. 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: 00000000011417432074 00000000017035267151
           00000000016231273040 00000000015633664563
0xc02e4830: 00000000006213431040 00000000004016030456
           00000000015733671050 00000000014524040164

>> 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
```

findsym

Alias

fsym,symbol

Usage

Usage

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

Description

Note

To find out how the history mechanism works, please refer to [the Section called *User Interface* in Chapter 3](#).

Example 4–8. history

```
>> history
```

```
- gzip: gzip compression (default)
-x:  exclude memory regions to reduce dump size:
    - u: Exclude <U>ser pages
    - b: Exclude <B>uffer cache pages
    - p: Exclude <P>age cache pages
    - f: Exclude <F>ree (unused) pages
-t:  target directory where dump is written (default is current directory)
-i:  write info file which describes dump content
-v:  verbose output
-i:388Cseons to reduce dump size:
```

Alias

Usage

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	bffffd000	c0000000	-8192	177

=====

1 active mm_struct struct found

module

Usage

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

-f [modulename]

Lcrash HOWTO

```
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.

=====
```

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

report

Usage

```
report [-w outfile]
```

Description

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

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 liodsd.
- + If called with '-v', all debug views which are available to 'lcrash' are liodsd.
- + If called with the name of a debug log, all debug-views for which the debug-log has regiodsred are liodsd. It is possible that some 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 prindsd.

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 [...]

Printditionallr 4_lbasicata ty, of stru,ata tdefs) s for Td(memers of structypes in by


```
2968k data, 88k init, 0k bigmem)
...
```

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

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

>> dump 0x180 16
0x180: 00000000 000100e5 000100e5 00000001 : .....
```

Yes, `cat /sys/kernel/btf/symtab` is a table of exported kernel symbols.

Example 4–17. `symtab`

For a comprehensive example please refer to

[the Section called *Analyze Kernel Modules* in Chapter 5.](#)

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:

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

=====

1 active task struct found

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 (previously 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	bdfush
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

67e5a58:	00000000	00000000	00000000	00000000
67e5a68:	070dd000	c00e357a	00000000	400e3578
67e5a78:	ffffffff	7ffff7fc	00000002	00000000
67e5a88:	0048a668	00402c0c	00000001	00000000

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
=====
m e  cec99d01030ec99d01030  60569=====
===== (ATA  m*)0xec99d01)->name=====

>> vtop -m c02a0x8 Td6a26=====
      VADDR      KADDR      PADDR      PFN
=====
=====char*) ec96a26=====
```

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 -

```
>> print ((module*) d00e6000)->refs0xd0106b80>> walk -s module_ref next_ref 0xd0106b80struct modu
```

```

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
=====

```



```
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: 00000000 : @T...3241980:
```


Lcrash HOWTO

```
=====
long unsigned int      BASE      00010000000000005  0000000000000000  4
  ST_BIT_OFFSET=0, ST_BIT_SIZE=0
  ELEMENT_TYPE=0x0, INDEX_TYPE=0x100000000000005, 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
PiocnlData           STRUCT   0001004e00000049  0000000000000000  20
Unique_t             TYPEDEF  0001004e00000006  0001000900000017  0
...
loff_t               TYPEDEF  000100090000000d  0001000c00000013  0
long double          BASE     0001000000000000e  0000000000000000  12
long int             BASE     00010000000000003  0000000000000000  4
long long int        BASE     00010000000000006  0000000000000000  8
long long unsigned int BASE    00010000000000007  0000000000000000  8
long unsigned int    BASE     00010000000000005  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     00010000000000013  00010000000000013  -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
=====
49 founds found
```

Chapter 5. Sample lcrash Sessions

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 **syntab**.

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

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                                [ ]
```



```
TEXT:          0 DATA:          0 RODATA:          0 BSS:          0
#SYMS:         14 /tmp/my_dummy.map [my_dummy]
TEXT: d000006 14:  aHm4      0 RO14:  a1f      014:  41cdummy]
```


Appendix A. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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. Secondly, this License preserves for the author and publisher a

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, 0 –13.2 Td(titpr

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.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your

Bibliography