

# 1 What is Alicia?

Alicia is an integrated dump analysis tool that was initially released in March 2005 as version 1.0. The users of former tools can easily migrate to the Alicia environment and benefit from the functional extensions in the new tool.

In Alicia, Linux Dump Analysis Scripts (LDAS) can be written in Perl, which simplifies scripting for analysis procedures.

For the Alicia starting procedure and application program interface (API) calls, see Appendix 5.1, "Alicia Execution."

## 1.1 Issues since Alicia 1.0

### 1.1.1 Dump Analysis Tool Integration

Alicia seeks to integrate dump analysis tools. Alicia 1.0 wrapped crash and Alicia 1.1 wrapped lcrash. These integration efforts have enabled dump analysts to use Alicia functions for both crash and lcrash. The analysts can always obtain the same results without being aware of which tool is wrapped in Alicia by using dump analysis scripts written to use the API calls.

### 1.1.2 Providing an usefull LDAS

One of the advantages of Alicia is that the user can create scripts and run them to analyze dumps. This means that Alicia is a dump analysis tool that has the potential for continued expansion with the support of many analysts. For example, when a user desires to perform a simple operation such as a loop while scanning a list of several structures chained with pointers or searching for and retrieving specific data from a large amount of information, a script reduces the time and effort compared to an interactive dump analysis in which the user manually enters commands.

Scripts are not just for one-time use, they can be reused in future analyses. This means that scripting corresponds to functional extension of Alicia.

This time, we create an usefull script for reporting information required at an initial stage of a dump analysis (initial LDAS) to enrich the LDAS set.

## 2 Implementation

We extended Alicia functions to resolve the "1.1 Issues since Alicia 1.0" by conducting the following implementations.

- Supporting lcrash
- Creating initial LDAS

### 2.1 Supporting lcrash

Alicia has supported lcrash since version 1.1.0, the latest release. This section checks that the various Alicia API calls provide the same LDAS output for both crash and lcrash. It also measures and compares the execution times of the API calls for crash and lcrash.

#### 2.1.1 Starting Alicia

The following listing shows how to start Alicia in the lcrash mode.

```
$alicia -lcrash [arguments to lcrash]
```

```
$ alicia -lcrash map dump kerntypes
```

The following sections will check that Alicia API calls return the same results for lcrash.

#### 2.1.2 kernel Function

We will use the kernel function to get the command image "task\_struct.comm" of a task. This example uses initial task "init\_task\_union." Use the lcrash "symbol" command to acquire the address of the Initial task task\_struct structure.

```
alicia> symbol init_task_union;
```

```
      ADDR  OFFSET SECTION      NAME      TYPE
=====
0xc049e000    0 GLOBAL_DATA  init_task_union  (unknown)
=====
1 symbol found
```

The following listing uses the kernel function to display the “comm” member of task\_struct.

```
alicia> print kernel('c049e000', 'task_struct', 'comm', 'char *')
```

The same result is provided for both crash and lcrash.

```
swapper
```

### 2.1.3 LDAS Execution (Using the get\_addr and get\_mem functions)

Using the shell escape function, start an external editor and edit LDAS.

```
alicia>!vi banner.ldas
```

Create a script using the Alicia command name as the subroutine name.

```
sub banner {  
    my $addr = get_addr 'banner';  
    print join ", map { pack "V", hex($_) } get_mem($addr, 15);  
}  
1;
```

Load the LDAS to Alicia and execute it.

```
alicia> load 'banner.ldas'  
alicia> banner
```

The same result is provided for both crash and lcrash.

```
<6>NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
```

### 2.1.4 Comparing API Call Execution Times

Using a Perl benchmark module, we created an LDAS for measuring the time to repeat each API call 10,000 times.

The following list shows the measuring environment for benchmark.ldas.

- Software

Kernel: MIRACLE LINUX V3.0 SP1 (2.4.21-20.19AX)

Perl: 5.8.0

Alicia: 1.1.0

- Hardware

CPU: Pentium M 1300MHz

Memory: 512 GB

- Target Dump

Kernel: MIRACLE LINUX V3.0 SP1 (2.4.21-20.19AX)

Table 2.1.4-1 Alicia API Call Execution Times: Alicia API Call Execution Times shows the result of execution (in seconds).

**Table 2.1.4-1 Alicia API Call Execution Times**

	get_addr	get_mem	kernel	ldas
crash	5	5	173	10
lcrash	1	4	2	5

The table indicates that lcrash executes faster than crash.

The kernel function is especially slow for crash. We think that this is due to the overhead of using the GDB function wrapped in crash instead of using the crash function on its own.

## 2.2 Initial LDAS

### 2.2.1 What is Initial LDAS?

When analyzing a dump generated upon failure, the procedure (e.g., which information should be checked first) depends on the failure. The analyst often investigates various pieces of information in the kernel by trial and error, relying on his or her experience and instinct.

In any case, it is important to get a rough sketch of the kernel state at an early stage. Using the Initial LDAS we developed, the user can easily obtain minimum information on the kernel state. It is also less time-consuming than each user developing a similar script.

### 2.2.2 Execution

In general, each LDAS is loaded into Alicia with the "load" command, but Initial LDAS does not require "load" execution because it is loaded when Alicia starts.

Initial LDAS writes dump analysis summary information in < specified

directory>/<specified file>, and detailed information under <specified directory>/MMDDhhmmss/ (where MM indicates month, DD day, hh hour, mm minute, and ss second). The following listing specifies an output to /demo/report.html.

```
alicia> report(output=>'/demo/report.html')
```

## 3 Consideration

### 3.1 Effectiveness of lcrash Support

The lcrash support has enabled the execution of lcrash commands from Alicia. Now users who have only used lcrash can use Alicia functions.

lcrash has the following advantages over crash, which are also inherited in Alicia.

- The kernel does not have to carry debug information.

crash needs a debug kernel image to obtain the information on symbol names and structures, but lcrash has a unique dump analysis mechanism that does not require a debug kernel image and can analyze the dump immediately after it is obtained by LKCD.

- It is possible to analyze dumps generated in a different architecture.

crash cannot analyze dumps that are generated in a different architecture. However, lcrash that was compiled in an IA-32 architecture environment can analyze a dump generated in an IA-64 architecture.

The comparison in Section 2.1.4 indicates that lcrash is faster in general. For the kernel function, lcrash is several tens of times faster than crash. We think this is because the kernel function uses the GDB function wrapped in crash.

The support of lcrash has boosted the Alicia API call execution speed. Dumps supported by lcrash can be opened in both crash and lcrash, but for a large amount of processing execution, running Alicia in the lcrash mode is more efficient.

### 3.2 Effectiveness of Initial LDAS

The advantages of Initial LDAS is that the user can obtain a failure analysis summary and a user with no knowledge of kernel internal or dump analysis can easily get a summary of what happened when dump was taken by running a predefined script.

When a failure occurs at a distant place, for example, it is necessary to send the dump

to the analysis team by email, but a raw dump file is sometimes too big to be attached to the mail. The file generated by Initial LDAS is 200 to 300 KB, which can be easily sent as an email attachment. (See Figure 3.2-1) Sending analysis information by email enables early starting of failure analysis and diagnosis.

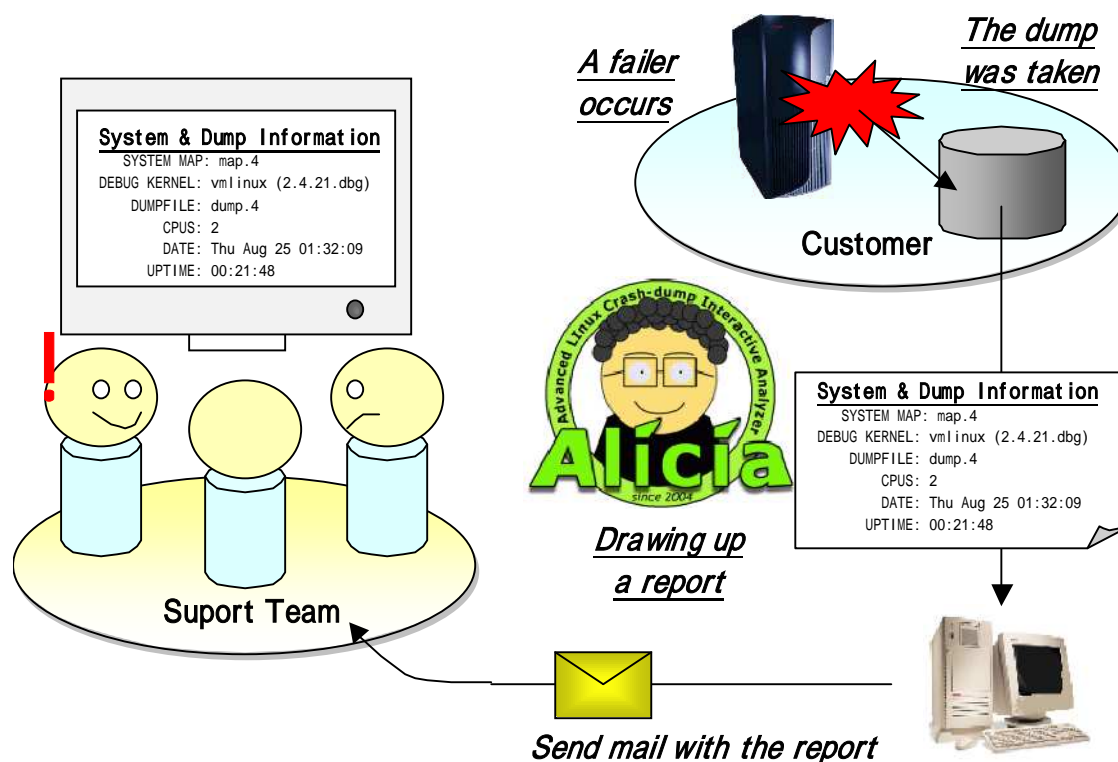


Figure 3.2-1 Initial LDAS directions for use

It is possible to determine whether the cause of a failure is a known problem by using the information listed in Initial LDAS. If it is known, early decision for the failure can be expected.

If Alicia and Initial LDAS have been installed in a Linux system, the above support is provided. By isolating the cause of a failure at an early stage, it is often possible to prevent another occurrence, or alleviate or localize the effect of the failure before it is resolved. This feature is very important for supporting mission critical systems.

As we mentioned, the Initial LDAS function for automatically generating the first analysis information is expected to greatly speed the analysis.

We developed the Initial LDAS that had been demanded since the advent of Alicia. The development task also had spillover effects. People who worked mainly in Linux support at customer sites and had not read the kernel source code or analyzed dumps joined LDAS development. This occasion developed their interest in the Linux source code and raised their support expertise. Conversely, this development has demonstrated that people who are unskilled in dump analysis can create LDAS.

In the design stage of Initial LDAS, staff working on dump analyses in different operating system platforms, not only Linux, exchanged opinions on "what information should be presented." They then investigated the correspondence of each proposed display item to those in Linux and added information specific to Linux. This task was answering the question "what information is required for analyzing Linux dumps?" We finally defined the current specifications after considering the importance of each item, degree of difficulty in developing each LDAS, and human resources.

By studying our Initial LDAS, it is possible to learn general know-how on creating LDAS and on the various data structures that are processed in the Linux kernel. Therefore, Initial LDAS will also play a significant role in serving as a material for teaching the Linux kernel.

## 4 Future Tasks and Plans

### 4.1 Future Tasks

It is necessary to further improve the Initial LDAS after this. To achieve this requires that the initialLDAS be used in many dump analyses to accumulate knowledge on the required types and amount of information to find a failure.

It is also necessary to review what information should be left in the kernel after a failure occurs. We intend to enrich dump information with an eye toward adding various tracing and recording functions to the kernel.

### 4.2 Future Plan

For sharing LDAS worldwide, we need a mechanism for retrieval by specifying purposes or kernel symbols, as well as sharing the LDAS source code. Initial LDAS has the capability to collect a wide variety of dump information. We plan to build an

information sharing and retrieval system following the current model.

## 5 Appendix

### 5.1 Running Alicia

This section describes a simple procedure for running Alicia in MIRACLE LINUX V3.0 SP1 (hereinafter called MIRACLE LINUX V3.0).

#### 5.1.1 Installing Perl Modules Required for Alicia

Download the Perl modules required for running Alicia from CPAN. CPAN is a general Perl code archive site.

```
# perl -MCPAN -e shell
cpan> install Term::ReadKey
cpan> install Term::ReadLine::Perl
```

#### 5.1.2 Installing crash and lcrash

Since MIRACLE LINUX V3.0SP1 includes both crash and lcrash, we will omit the procedure description. These programs can be downloaded from the following URL.

```
crash: ftp://people.redhat.com/anderson/
lcrash: http://lkcd.sourceforge.net/
```

#### 5.1.3 Installing Alicia

Alicia can be downloaded from the following URL.

<http://sourceforge.net/projects/alicia/>

It is necessary to download version 1.1.0 or later, which supports lcrash. Then install the program in the following procedure.

```
# tar zxvf Alicia-1.1.0.tar.gz
# cd Alicia-1.1.0
```

```
# make
# make install
```

#### 5.1.4 Alicia Settings

Set the crash and lcrash command paths in `/etc/alicia.conf`, which is created after installation.

The following listing shows their default settings.

```
Crash = /usr/bin/crash
Lcrash = /sbin/lcrash
```

#### 5.1.5 Dump Analysis Preparation

To analyze a dump, it is necessary to have a dump file and a kernel image (vmlinux) for resolving the symbol information. These files can be created easily in MIRACLE LINUX V3.0.

- Create a kernel image with debug information

Uncompress the kernel source code in `/usr/src/linux` and move to that directory. Perform a make specifying the CFLAGS make file with the `-g` option.

##### 1. Edit Makefile.

```
# vi Makefile
Add g option to CFLAGS
CFLAGS := -g -Wall -Wstrict-prototypes -Wno-trigraphs
```

##### 2. Perform make.

```
# make clean
# make
Vmlinux is made in current directory.
```

- Generate a dump

In MIRACLE LINUX V3.0, LKCD (Linux Kernel Crash Dump utility) is installed by default. Performing the following procedure generates a dump file under the `/var/log/dump` directory. The dump image is at first saved in the Swap partition, which

must be larger than the memory capacity.

1. Enable dump generation.

```
# echo 1 > /proc/sys/kernel/sysrq
```

2. Generate a dump.

```
# echo c > /proc/sysrq-trigger
```

The dump image is saved in `/var/log/dump/n/` (where 'n' indicates a sequence number that is incremented for each dump).

### 5.1.6 Start Alicia

Open the dump image in the above procedure. Alicia is started by using the following command syntax.

```
$ alicia [mode] [arguments in "mode"]
```

mode can be `-crash` or `-lcrash`.

The following listing opens a dump image in the crash mode.

```
$ cd /var/log/dump/0
$ alicia -crash map.0 /usr/src/linux/vmlinux dump.0
:
alicia>
```

When the "alicia" prompt appears, it is ready to start analysis.

### 5.1.7 Introduction of API Calls

- (1) `pass_through ('crash command')`

This function returns the result of `existing_dump_analysis_command(crash or lcrash execution)` in a character string. The following example acquires the list of `task_struct` structure addresses from the `ps` command in `crash`.

```
alicia> @ps = map { (split /[ \t>+]/)[4] } split /%n/, pass_through 'ps
```

- (2) `get_addr('kernel_symbol_name')`

This command returns the address of `kernel_symbol_name`.

```
alicia> $addr = get_addr('saved_command_line') # $addr is used at example (3) get_mem
```

(3) `get_mem('address', number_of_words)`

This function returns the contents of the memory location `number_of_words` from an `address`.

```
alicia> @outs = get_mem($addr, 256/4)
alicia> print join ", map { pack("V", hex($_)) } @outs # to ASCII
ro root=LABEL=/ hda=ide-scsi
```

(4) `kernel('address', 'structure', 'member_variable', 'type')`

This function returns the member variable of a specified kernel structure.

```
alicia> $utsname = get_addr('system_utsname');
alicia> print kernel($utsname, 'new_utsname', 'release', 'char *');
2.4.21-9.30AXsmp
```