Linuxカーネルのクラッシュダンプ解析

Analyzing Linux Kernel Crash Dump

前 原 志 好

要 約 Linuxをミッションクリティカル領域で使うためには，カーネル本体の安定性や拡張性の他に，障害が発生した際のトラブル追及の迅速化・容易化が必須である．本稿では，Linuxカーネルのダンプ解析について，現状を分析し，不足している部分を指摘する．さらに，メインフレームとの比較を行い，その不足分を補うダンプ解析ツールAliciaの作成までを取り上げる．

Abstract To advance the Linux into mission-critical areas, the quickness and easiness of troubleshooting is required. In this paper, first, we give a brief description of current dump analysis tools and point out what is required to analyze dump and then dump analysis of mainframe and Linux are compared for dump analysis tool. We introduce and explain new dump analysis tool Alicia to fill in the gap in these tools.

1. はじめに

2004年頃からオープンソースソフトウェア（以下，OSS）[*1]というキーワードが世間を賑わせるようになってきた．その様々なOSSを活用するための基盤となるオペレーティングシステム（以下，OS）として，Linuxが採用されるケースが多い．そのためLinuxの信頼性向上は，OSSを用いたシステムでは，非常に重要である．

Linuxカーネルもバージョンが2.6となり，ミッションクリティカルな環境でも耐えられるように，スケーラビリティの確保やダンプ[*2]を採取する機能も取り込まれるようになってきたが，性能情報を詳細に見るための機能や，ダンプを解析するためのツールがまだまだ不足している．つまり，メインフレームのOSには当たり前のように備えている機能やツールが十分ではないため，Linuxはメインフレームと比較すると，信頼性が低いと言わざるを得ない．

本稿では，Linuxカーネルのダンプ解析部分に焦点を合わせ，現在のダンプ解析ツールの紹介と問題点，また，それを克服したツールAliciaについて説明する．

2. ダンプ解析ツール

ここでは，現在，開発や保守の現場で使われている主なLinuxカーネルのダンプ解析ツールについて紹介する．また，Unisys製メインフレームClearPath Server CMP CSシリーズ・HMP IXシリーズのカーネルであるEXECのダンプ解析ツールについても少し触れ，それらの違いについて分析する．

2.1 Linuxカーネルとは

http：//www.linux.or.jp/general/linux.html[1]から一部引用すると，Linuxとは

「自由に再配布することのできる，独立したUnix系オペレーティングシステム（OS）のことです．Linuxを動かすことのできるアーキテクチャはx86, Motorola 68k,

Digital Alpha，SPARC, Mips, Motorola PowerPC など実に多岐に渡ります．Linux は本来 OS の中核となるカーネル（kernel）だけを指す名称ですが，Linux カーネルベースのシステム全体をさして「Linux」と表現することもあります．」
（一部引用：Linux とは，Webmasters of www.linux.or.jp. 1998–2006）

と定義されている．本稿では，本来の意味である OS の中核となるカーネルのことを単に Linux と表記し，Linux カーネルを利用したシステム全体のことを Linux システムと表記する．

また，自由に再配布できるということについて，同サイトでは以下のように述べている．

「Linux と他の OS との違いとしてひとついえることは，値段---つまりフリー（free）であることです．つまり，誰にもロイヤリティやお金を払うことなく，コピーや再配布をすることができるのです．しかしながら，値段よりもむしろフリーであることに関して重要なことがあります．Free Software Foundation の General Public License により，Linux のソースコードは誰でも入手することができます．」
（一部引用：Linux とは，Webmasters of www.linux.or.jp. 1998–2006）

この通り，Linux カーネルは値段も無償な上，フリーにソースコードも閲覧できる．また，閲覧できるだけではなく，誰でもカスタマイズやパッチの作成が可能であり，これが非常に重要なポイントである．なぜなら，ダンプを解析するためには，ソースコードからデータ構造を把握したり，カーネルがパニック[*3]に陥ったところから遡って，ソースコードを見ながらトレースしたりする必要があるからである．

2.2 Linux のダンプ解析ツール

Linux はソースコードを閲覧できるため，採取されたダンプがどのような形式を持っており，どのようなデータがダンプに保存されるのか，また，そのダンプのデータがソースコード中の何を意味するのかが分かる．

ソースコードが公開されていない OS（例えば，Windows 等）で採取されたダンプは単にデータの塊であるだけで，そのデータが指し示す「意味」までは理解できない．つまり，ダンプ解析ツールを作るには，OS のソースコードにアクセスできなければならない．逆に，Linux はソースコードが公開されているので，誰でもダンプ解析ツールを作成できるのである．

2.2.1 主なダンプ解析ツール

Linux のダンプ解析では，crash と lcrash というツールが，主に現場で使用されている．両者とも，ダンプファイルのデータに仮想アドレス（virtual address）を用いてアクセスできる．つまり，物理メモリ（physical address）を意識することなく，通常のカーネルコーディングで扱っているようにアドレッシングできる．

crash[2]は，UNIX の crash ツールのインターフェースをベースにしており，Red Hat の社員により作成されている．構造体へのアクセスに GDB[*4] を用いていることが大きな特徴である．また，ダンプ編集のスピードアップのためによく利用されるデータのキャッシング機能や，動的ライブラリによる拡張コマンドの追加機能も備えている．

lcrash[3]は，LKCD[*5]で採取されたダンプを編集するために作成されたツールである．crashのように外部のモジュールは利用しておらず，すべて自前の機能で実装している．また，sialと呼ばれるC言語に似た記述方法で作成したスクリプトをインタプリタとして動かせるエンジンを持っている．これにより，カーネルの変数や構造体にアクセスするスクリプトを作成することができる．

2.3 メインフレームのダンプ解析ツール

この節では，Unisys ClearPath Server CMP CSシリーズ・HMP IXシリーズのOSであるEXECのダンプ解析ツールを紹介する．EXECのダンプ解析ツールは，DMPLIBSというパッケージで提供されている．DMPLIBSは以下の内容で構成されている．

- 運用関連ツール・スクリプト群
  ダンプ採取，ダンプデータのテープへの保存，ダンプデータのネットワーク転送，必要のなくなったダンプデータの削除，スワップファイルと主メモリダンプの連結
- ダンプ解析用ライブラリ群
  ダンプ解析プログラム

これに加え，ダンプデータと実行ファイルへのアクセスを提供するツールであるDAP（Dump Analysis Processor）というプロダクトがある．DAPはOSのソースコードの外部参照名を解決し，また，FLIT（Fault Location Interpretive Testing）という軽量言語（PASCALに似た文法で簡単な構造化プログラミングができる言語）で，プログラムを作成することを可能にする．

2.4 ダンプ解析ツールの比較

Linuxのダンプ解析ツールには，ダンプ解析用ライブラリ群というものが存在しない．ダンプを解析するためのコマンドは，すべてダンプ解析ツールに埋め込まれており，外側で簡単に修正したり，追加したりという構成にはなっていない．

また，lcrash, crashともに，拡張コマンドを作成可能であるが，簡単なサンプルがあるだけである．一方，メインフレームのDMPLIBSのソースコードは簡単に閲覧可能であり，かつライブラリとして充実しているため，それらを参考に自分用のコマンドを簡単に作成できる．

つまり，メインフレーム並みのダンプ解析環境を整えるには，ユーザが簡単にコマンドを追加できる仕組みと，参考になるライブラリ群を作成することであると考えられる．

メインフレームのダンプ解析ツールの視点から，Linuxのダンプ解析ツールにない点を表1にまとめた．

3. 新ダンプ解析ツール Alicia

本章では，まず，Alicia（Advanced LInux Crash-dump Analyzer）がどんなツールであるかを簡単に説明する．次章にて，Aliciaが現在のLinuxカーネルダンプ解析ツールが抱えている問題をどのように解決しているかを検証する．

3.1 基本方針

ダンプデータにアクセスするための機能を一から作成するのは非効率的であり時間もかかる

表 1　ダンプ解析ツールの機能比較

項目	内容
インタラクティブ	両者ともコマンドプロンプトで，実行結果を逐次見ながら解析を続けることができる．しかし，Linux 側は，実行結果を変数に格納することができない(パイプを利用して外部のプログラムで編集は可能).
スクリプト	Linux 側には C 言語やそのインタプリタ用言語 sial で記述可能であるが，変数の型などを気にすることなく，簡易に記述できる軽量言語で記述できる機能はない.
解析の共有化	Linux 側では，ほとんどのコマンドを本体のツールに組み込んでしまっており，それらを簡単にカスタマイズして実行するという利用方法がない．一方，DMPLIBS には，解析のためのロジックのみが入っており，解析者は先人のノウハウに簡単にアクセスできる

ため，crash, lcrash を利用する．実は，こうすることでメリットもある．開発した新ダンプ解析ツール Alicia から，それらが持っているコマンドも実行できるように設計することで，既存のユーザからも Alicia を利用しやすくなる．

次に，メインフレームでは，ダンプデータへアクセスする API を DAP 経由で発行することができ，また DAP は FLIT 言語によって，ダンプの解析コマンド（スクリプト）を記述する機能を持っている．Linux 側では，軽量な高級言語でダンプデータにアクセスする手段がない．そこで Alicia は，そのような言語でダンプデータへのアクセス機能を備える必要があった．

しかし，FLIT のように Unisys 独自の言語を Linux 側で実装しても，多くの人がまずその使い方の習得に時間をかける必要が出てくる．よってここでも，既にいくつか存在している軽量言語を利用することとし，実績があって利用者も多く，まだ進化も続けている Perl を採用した．

3.2　Alicia を加えた Linux ダンプ解析環境

今までの Linux ダンプ解析環境に Alicia を加えた場合，ダンプ解析環境は，図 1 のようになる．この図の右側の網掛け部分が，Alicia 本体と Alicia 上で利用できるダンプ解析スクリプトである．この解析スクリプトのことを LDAS：Linux Dump Analysis Script と呼ぶ．

3.3　Alicia の機能

Alicia は，大きく分けて四つの機能を持っている．ダンプ解析者は，それらの機能を利用して，ダンプ解析環境の向上，解析時間の短縮を手に入れることができる．以下に，それぞれの機能について，簡単に説明する．

1）　Wrapper 機能

既存ダンプ解析ツールをラッピングする．単にラッピングして，Alicia 独自のコマンド体系にしてしまうのではなく，既存ツールが持っているコマンドセットもそのまま使用できるようにしている（一部のコマンドはサポートしない）.

2）　Interactive Perl

対話的に解析を進めることができる．そのときに，Perl の構文が利用できるため，Alicia に入力したコマンドの結果を変数に格納し，それをさらに別のコマンドへの入力にするという使い方が可能である．

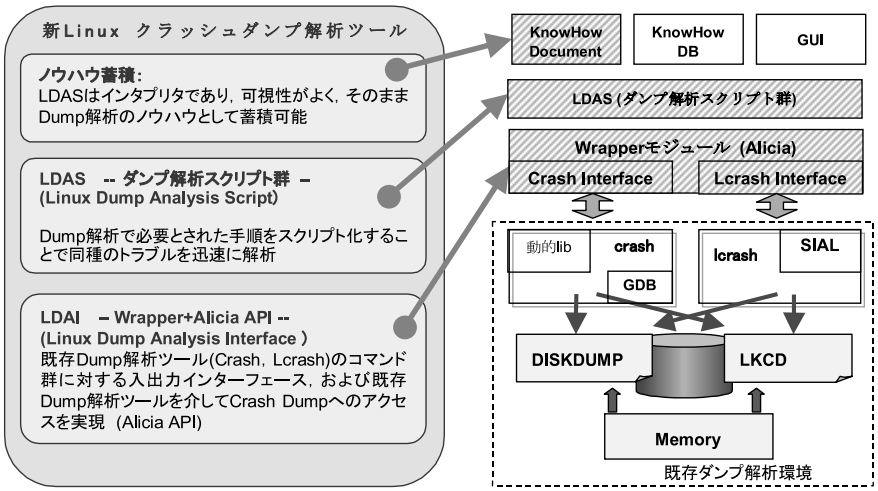

図 1　Alicia の全体構成図

3）　Scripting 機能

作成しておいた LDAS をロードし，コマンドとして対話的に実行することができる．また，その LDAS を関数として利用することで，別の LDAS の中に組み込むことも可能である．

4）　Alicia API

スクリプトの中で利用しやすいいくつかの独自の関数（コマンド）を持つ．それらは，ラッピングするツールに依存することなく同じ結果を返す．これにより Alicia API を利用した LDAS もラッピングするツールに依存しない．

4.　Alicia による課題解決

Alicia は，2.4 節で挙げた課題に対して，スクリプト言語としてメジャーである Perl を解析環境に導入することで解決をはかっている．具体的に，どのように解決したかについて，Alicia を実際に使用してみることで，順に報告していく．

4.1　Alicia での解析方法

Alicia はインタプリタの記述言語として，Perl を採用している．そのため，sial や FLIT のような独自言語を使用することなく，解析スクリプト（LDAS）を記述できるようになっている．つまり，Perl 言語に習熟した者にとっては，新しい知識を習得することなく，ダンプ解析を行えるようになるのである．また，Perl 未経験者でも，簡単な LDAS であれば数日で作成できるようになったという実績がある．

4.1.1　インタラクティブ

Alicia は，Perl を一行ずつ実行するインタプリタ機能を持っているので，それを利用した Alicia の API と Perl を組み合わせた例を示す．ここでは，Linux カーネルの起動オプションを表示させる．

```
alicia> $addr＝get_addr（'saved_command_line'）　　・・・・・①
alicia> print map｛pack（"V", hex（$__)）｝get_mem（$addr, 64)・・・・・②
ro root＝LABEL＝/profile＝2 nmi_watchdog＝1
```

①Alicia の API：get_addr を利用して外部参照変数：saved_command_line のアドレスを，Perl の変数：$addr に格納する．
②Alicia の API：get_mem で $addr のアドレスから 64 語分取得し，Perl の pack 文で文字列に変換したものを Perl の print 文で画面に出力する．

4.1.2 解析スクリプト

システムの情報を表示するシェルコマンド uname と同じ結果を出力する LDAS を作成する．以下のプログラムを uname.ldas という名前で保存する．

```
sub uname ｛　　・・・・・①
    my $uts＝get_addr（'system_utsname'）；　　・・・・・②
    print join ''，map｛kernel（$uts，'new_utsname'，$__，'char＊'）｝
        qw/sysname nodename release version machine/；　　・・・・・③
｝
1；
```

①LDAS は一つ以上のサブルーチンから構成される．サブルーチン名がそのまま Alicia のコマンドとなる．記述方法は，Perl の文法と同等である．また，Alicia の API も Perl の関数として扱うことができる．
②Alicia の API：get_addr を利用してカーネルの外部参照変数：system_utsname のアドレスを Perl の変数：$uts に格納する．
③Alicia の API：kernel を利用して，構造体 new_utsname の指定したメンバ変数の値を表示する．

作成した LDAS は，Alicia の load コマンドを利用して登録し，作成した関数（コマンド）を実行する．

```
alicia> load 'uname.ldas'
alicia> uname
Linux localhost.localdomain 2.6.9-11.19 AX #1 Fri Aug 5 05：12：07 EDT
2005 i 686
```

4.1.3 解析手順の共有化

前項の uname.ldas のような解析スクリプトをライブラリ化することによって，解析手順の共有化を図ることができる．そのため，作成したスクリプトを集めて，Alicia と一緒にリリー

スすることで，EXEC の DMPLIBS と同じようなライブラリ環境が実現できるはずである．

しかし，EXEC の DMPLIBS の充実度は，Alicia のライブラリと比較にならないほど高い．なぜなら EXEC では，OS の新機能を作成する時に，コードを作成した人間がダンプ解析スクリプトも作成する義務があるので，ライブラリが自然と充実していくからである．一方，Linux では，コード作成者にはそのような義務はないし，障害解析がダンプを使って行われるという認識がほとんどないため，Linux でのダンプ解析ライブラリの充実化は，Alicia のユーザのみによってしかなされていない．

4.2 Alicia を使ったダンプ解析例

あるプロセスが大量に CPU を使用していないか調査する．ここでは，Linux カーネル 2.6.9 のダンプに対して，Alicia-1.1.4 を使用して解析する．

```
class 'task_struct', qw{tasks stime utime comm};
sub pstime {
    $ts=task_struct(get_addr('init_task'));
    $ts->list('tasks');
    $ts->each(sub {
        my $ts=shift;
        printf("%9s %9lu %9lu", $ts->{addr}, $ts->stime, $ts->
                utime);
        print $ts->get('comm', 'char*'), "¥n";
    });
}
1;
```

上記の LDAS は，ダンプ採取時に実行されていたプロセス全てに関して，プロセスの情報を保持している構造体 task_struct から，システムで使用された実行時間とユーザプログラムの実行時間を表す stime と utime というメンバ変数を出力させている．実行結果を以下に示す．

```
alicia> cputime
0xc035bbc0      698109              0 swapper
0xc1491770         258              1 init
0xc14911a0           3              0 ksoftirqd/0
0xc1490bd0          11              0 events/0
0xc1490600          11              0 khelper
（途中，省略）
0xc84f7830         214           3089 stress
0xc84f7260        3289             12 stress
0xc84f6c90         242           3095 stress
```

左側から，task_struct の仮想アドレス，stime, utime が出力される．このように，簡単なス

クリプトを書くことで，カーネルのデータ構造に簡単にアクセスでき，また，繰り返しの処理のためのコーディング量を減らすことができる．

5. Alicia の活用方法

ダンプの一次解析では，解析のスピードが求められる．なぜなら，障害の切り分けができないと，問題の再発への対応策を考えることができないからである．もし，障害の情報が全くない場合，再起動後のシステムの信頼性はゼロといってもよい．

また，ダンプは生き物と呼ばれるように，ダンプが採取された瞬間毎に違う顔を見せる．ここで必要なことは，素早く簡単に解析スクリプトをその場で作成できる環境である．そのような環境を，Alicia は用意する．

このように，どのようなダンプが吐き出されるか分からない状況で，さらに，その大量の情報が詰まったダンプから，いかに手軽に必要な情報を抽出できるかという場面で，Alicia は効果を発揮する．しかも，ダンプ解析後は，解析に使用したスクリプトを保存しておくことで，そのノウハウを蓄積することができる．

しかし，4.1.3 項で述べたように，ノウハウのぎっしり詰まったライブラリが現在は存在しない．今後は，Alicia 本体とは別にライブラリをリリースするようにしたり，ライブラリをオンラインで参照・更新できるようなシステムを作ったりすることが必要となる．また，カーネル作成者にもスクリプトを書いてもらえるように，様々な言語でスクリプトが書けるようにしていかなければならない．

6. お わ り に

Alicia によって，Unisys のメインフレームにあるようなダンプ解析の基盤は完成したと考えるが，Linux をミッションクリティカル領域で使用していくにあたっては，ダンプ解析ライブラリの充実，つまり Linux でのダンプ解析のノウハウを充実させることが必須である．メインフレームは長年の積み重ねでライブラリが充実しているが，Linux はまだダンプ解析に対する経験が浅いため，ノウハウが蓄積していない．

そのためには，Alicia が OSS であることの強みを生かし，日本ユニシスグループだけでなくコミュニティからもノウハウを集める必要がある．しかし現在のところ，十分なフィードバックを得られるほどコミュニティが活性化できていない．コミュニティを活性化させるには，Linux がビジネスでも利用できる品質であることを，実際に示していくことである．

また，コミュニティを活性化させる別の方法として，まず，ダンプ解析というものを広める活動を行う予定である．その第一弾として，日経 BP 社の Web サイト IT Pro の「オープンソースのトラブル解決」というコーナーに記事「第 3 回カーネル・パニック」[4]を執筆した．ダンプ解析に取り掛かるのはそれほど難しいものでないという内容である．このような活動を続けることによっても，Alicia の存在をアピールしていく所存である．

＊ 1　OSS：ソースコードを公開したソフトウェア．使用・修正・配布等には，OSS を公開した組織により，何らかの条件・ライセンスが適用される．詳細は，http：//www.opensource.gr.jp/osd/osd-japanese.html を参照のこと．

＊ 2　ダンプ：ある瞬間のメモリの内容をなんらかのデバイスに吐き出したもの．人間だと CT スキャンされた情報がそれに近い．

＊3　カーネルがパニック：カーネルが予期しない現象を発見し，システムを停止させること．
＊4　GDB（The GNU source-level Debugger）：GNU プロジェクトで作成されているデバッガ．実行ファイルのシンボル情報やダンプ（コア）ファイルのデータ構造へのアクセスを提供する．http：//www.gnu.org/software/gdb/を参照のこと．
＊5　LKCD（Linux Kernel Crash Dump）：ダンプ採取・解析の統合プロジェクト．米 SGI 社によって開発され，sourceforge.net で開発が継続されている．

参考文献

［1］www. linux. or. jp の Webmaster, Linux とは（訳），http：//www.linux.or.jp/general/linux.html（原 典：http：//www.linuxjournal.com/xstatic/community/whatislinux）
［2］Red Hat Software, Inc., http：//people.redhat.com/anderson/crash_whitepaper/, 2003
［3］LKCD Team, http：//lkcd.sourceforge.net/，2005
［4］前原志好,「止まる」「遅い」を元から絶つ！オープンソースのトラブル解決第 3 回カーネル・パニック，http：//itpro.nikkeibp.co.jp/article/COLUMN/20060629/242051/, 2006
［5］独立行政法人情報処理機構,「OSS 性能・信頼性評価/障害解析ツール開発」ダンプデータ解析ツール（Alicia）の評価と考察，2004
［6］OSS 技術開発・評価コンソーシアム,「OSS 性能・信頼性評価/障害解析ツール開発」OS 層～Alicia 編～，2005

執筆者紹介

前　原　志　好（Shiko Maehara）
1997 年日本ユニシス(株)入社．米国 UNISYS 製メインフレームのカーネルを担当する．2004 年 4 月ユニアデックス(株)に転籍．その後 OSS 関連作業に着手．現在，ユニアデックス(株)ソフトウェアプロダクト統括部 OSS 推進室所属．