NTT DATA
Trusted Global Innovator

＜セキュリティ・キャンプ全国大会2018＞
Linux 故障解析入門

2018年8月16日
NTTデータ先端技術株式会社
半田哲夫

© 2018 NTT DATA INTELLILINK Corporation

自己紹介

高校1年生の時にプログラミングに出会い、今でもプログラムを書き続けている、Linux カーネル開発者です。「熊猫さくら」(@KumanekoSakura)として活動しています。

セキュリティ・キャンプでは毎年違う話をしています。資料は全て公開されていますので、興味のある方はダウンロードしてください。

今年は、サポートセンタで Linux システムの問合せ対応に携わった経験から、故障解析について考えます。今年の新入社員向け1日研修で使用した内容です。

© 2018 NTT DATA INTELLILINK Corporation

ITシステムに関する工程／領域

上流工程

実現したいシステムを考えて
方針を決める⇒企画

実現する仕様について決める⇒設計

仕様に沿った処理を
プログラムとして記述する⇒開発

想定通りに動作するかどうかを
確認する⇒試験

システムの中でプログラムを利用する⇒運用

（試験では見つけられなかった）想定外の
出来事に対処する⇒保守／故障解析

下流工程

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ先端技術株式会社

ITシステムに関する工程／領域

大まかな分類

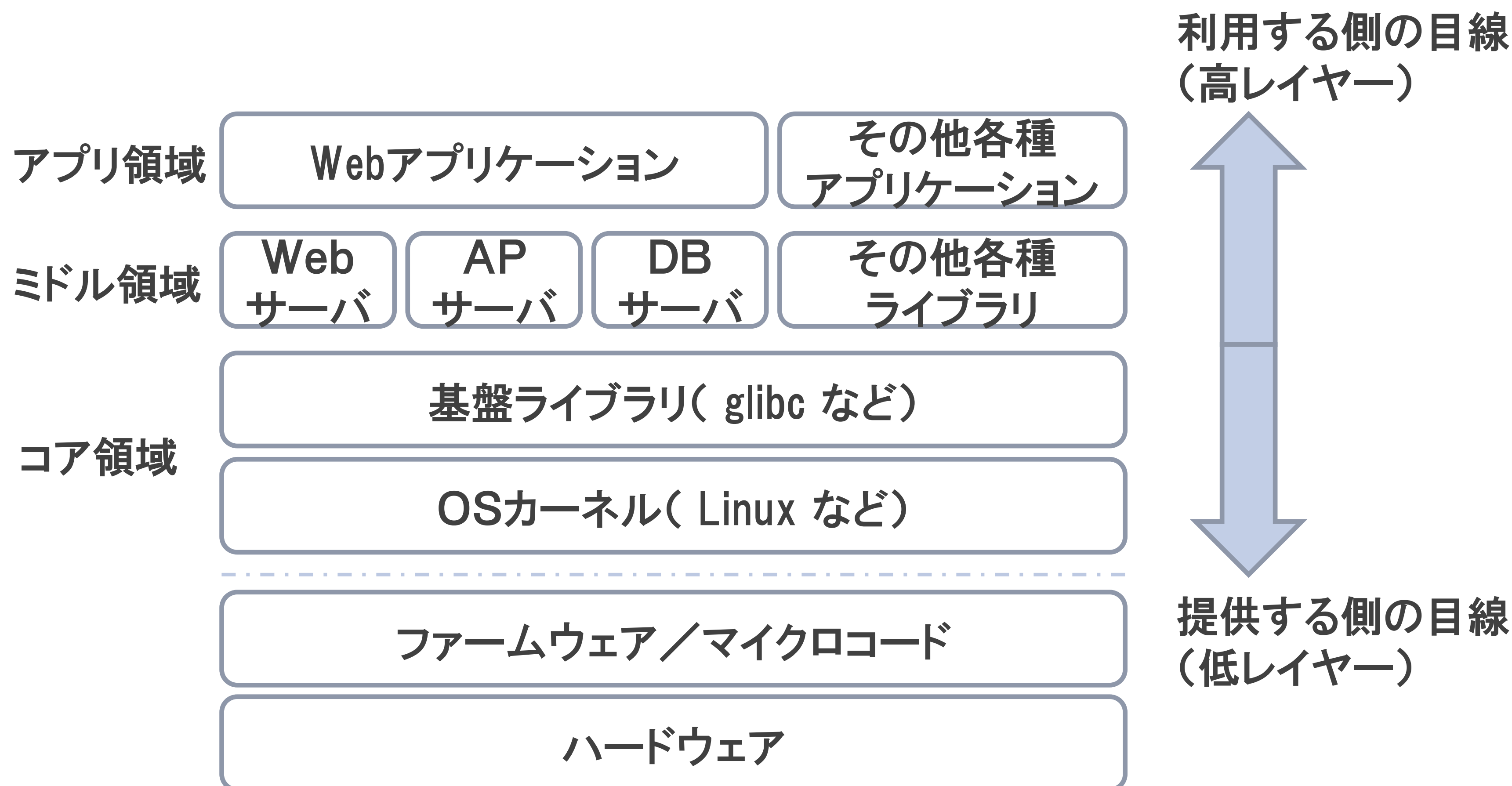
ソフトウェア

ハードウェア

© 2018 NTT DATA INTELLILINK Corporation

ITシステムに関する工程／領域

細かな分類



© 2018 NTT DATA INTELLILINK Corporation

ITシステムに関する工程／領域

余談：動作する環境による分類

コア領域とかミドル領域とかアプリ領域とかいう分類がされていますが、
動作環境という視点から見れば、カーネルモードとユーザモードの2種類
だけです。

OSが提供する保護機能が有効な
ユーザモードで動作するソフトウェア

トラブルの影響範囲は当該
プロセスのみに限定される
⇒複雑／大規模な処理を行うことができる

ユーザモードに保護機能を提供するために
カーネルモードで動作するソフトウェア

ハードウェア

トラブルの影響範囲が
システム全体に及ぶ
⇒必要最低限の処理だけを、慎重に行うことが求められる

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ先端技術株式会社

今日の講義内容

システムにはトラブルが付き物です。今日は、サポートセンタでもどかしい思いをした経験から、スムーズな問題解決を行えるようになるために、トラブルへの対処方法の基本について考えていきます。

トラブルは色々な領域で発生するものですが、

上流工程を経験する前に下流工程を経験しておく
高レイヤーを経験する前に低レイヤーを経験しておく

ことが、スムーズな問題解決を行えるようになるために有用だと考えています。

© 2018 NTT DATA INTELLILINK Corporation

今日の講義内容

そのため、私の講義では、馴染みの薄い人が多いであろう「OSレイヤー」を扱います。

環境としては、

　Windows PC にインストールした Oracle VM VirtualBox 上で動作する CentOS 7 ゲスト

を使用します。そして、

　OSレイヤーから Linux システムの動きを理解し、問題解決に必要なことについて知る

ことを目標とします。

NTT DATA
NTTデータ先端技術株式会社

講義で使用するネットワーク環境について

講義で使用するノートPCには、予め CentOS7-sc2018 という仮想マシンをインストールしてあります。この仮想マシンには、ポートフォワーディングの設定が行われています。

Windows から ssh アクセスしたい場合は TeraTerm から 127.0.0.1:22 へ接続してください。アカウントは root / sc2018 および penguin / sc2018 の2つを用意してあります。

Windows から http アクセスしたい場合はブラウザから http://127.0.0.1:80/ へ接続してください。

© 2018 NTT DATA INTELLILINK Corporation

コンピューター＝電子計算機

定められた手順に従って高速に計算することが、コンピュータの特技です。

計算処理をすることによって、目的を達成します。

何かの入力を与えて、何かの処理を行い、何かの出力を得るために、処理内容を記述したもの＝プログラムです。

ITシステムは、極論すると「入力＋処理＋出力」の組合せ＆繰り返しです。

© 2018 NTT DATA INTELLILINK Corporation

NTT Data
NTTデータ先端技術株式会社

アルゴリズム＝手順

電子計算機が高速に計算する能力があっても、何をどのように計算すれば良いのかを伝えてやらなければ役に立ちません。また、上手な計算方法を伝えてやらないと、力技では処理しきれない場合もあります。

NからMまでの連続する整数（ただし、N＜Mとする）の和を計算する場合の手順について考えてみましょう。

力技：

N＋（N＋1）＋（N＋2）＋中略＋（M－2）＋（M－1）＋M

公式：

（N＋M）＊（M－N＋1）／2

© 2018 NTT DATA INTELLILINK Corporation

アルゴリズム＝手順

計算例：

１＋２＋３＋４＝１０
（１＋４）＊（４－１＋１）／２＝１０

５＋６＋７＋８＝２６
（５＋８）＊（８－５＋１）／２＝２６

９＋１０＋１１＋１２＋１３＝５５
（９＋１３）＊（１３－９＋１）／２＝５５

手順の中で、どのような知恵／工夫をしているか、それが「価値」の源です。

© 2018 NTT DATA INTELLILINK Corporation

NTT DaTa
NTTデータ 先端技術株式会社

処理内容を記述するための様々な「言語」

C
Java
HTML
JavaScript
XML
SQL
シェル(Bashなど)
Perl
Python

ITシステムを提供する上で、「言語」に触れることは避けられません。
目的に合った言語を「理解して」使うことが大事です。

© 2018 NTT DATA INTELLILINK Corporation

演習：連続する整数の和を計算するプログラムを探る

コマンドライン引数でNとM(ただし、N≦Mとする)を渡すものとして、C言語で計算するプログラムの例を以下に示します。

```
---------- sum1.c start ----------
#include <stdio.h>

int main(int argc, char *argv[])
{
	short int n, m;
	if (argc != 3 || sscanf(argv[1], "%hd", &n) != 1 || sscanf(argv[2], "%hd", &m) != 1 || n > m) {
		printf("ERROR: Bad arguments¥n");
		return 1;
	}
	printf("Sum of integers between %hd and %hd is %hd¥n", n, m, (n + m) * (m - n + 1) / 2);
	return 0;
}
---------- sum1.c end ----------
```

コンパイルして実行してみましょう。

© 2018 NTT DATA INTELLILINK Corporation

演習：連続する整数の和を計算するプログラムを探る

コンパイル手順例：

```
$ gcc -Wall -O3 -o sum1 sum1.c
```

実行結果例：

```
$ ./sum1 1 10
Sum of integers between 1 and 10 is 55
$ ./sum1 1 100
Sum of integers between 1 and 100 is 5050
$ ./sum1 1 1000
Sum of integers between 1 and 1000 is -23788
```

あれれ？値が大きくなると想定外の結果が生じてしまいました。
実は、変数が扱える値の範囲には制限があり、その範囲を超えると想定外の結果になります。想定外の結果の使われ方によっては、誤動作／クラッシュ／脆弱性の原因となることもあります。

© 2018 NTT DATA INTELLILINK Corporation

NTT Data
NTTデータ先端技術株式会社

演習：連続する整数の和を計算するプログラムを探る

short int では扱えない範囲の値を扱う場合、int を使います。

```
---------- sum2.c start ----------
#include <stdio.h>

int main(int argc, char *argv[])
{
	int n, m;
	if (argc != 3 || sscanf(argv[1], "%d", &n) != 1 || sscanf(argv[2], "%d", &m) != 1 || n > m) {
		printf("ERROR: Bad arguments¥n");
		return 1;
	}
	printf("Sum of integers between %d and %d is %d¥n", n, m, (n + m) * (m - n + 1) / 2);
	return 0;
}
---------- sum2.c end ----------
```

演習：連続する整数の和を計算するプログラムを探る

実行結果例：

```
$ ./sum2 1 1000
Sum of integers between 1 and 1000 is 500500
$ ./sum2 1 10000
Sum of integers between 1 and 10000 is 50005000
$ ./sum2 1 100000
Sum of integers between 1 and 100000 is 705082704
```

また想定外の結果になってしまいました。

© 2018 NTT DATA INTELLILINK Corporation

演習：連続する整数の和を計算するプログラムを探る

int では扱えない範囲の値を扱う場合、long int を使います。

```
---------- sum3.c start ----------
#include <stdio.h>

int main(int argc, char *argv[])
{
	long int n, m;
	if (argc != 3 || sscanf(argv[1], "%ld", &n) != 1 || sscanf(argv[2], "%ld", &m) != 1 || n > m) {
		printf("ERROR: Bad arguments¥n");
		return 1;
	}
	printf("Sum of integers between %ld and %ld is %ld¥n", n, m, (n + m) * (m - n + 1) / 2);
	return 0;
}
---------- sum3.c end ----------
```

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ先端技術株式会社

演習：連続する整数の和を計算するプログラムを探る

実行結果例：

```
$ ./sum3 1 100000
Sum of integers between 1 and 100000 is 5000050000
$ ./sum3 1 1000000
Sum of integers between 1 and 1000000 is 500000500000
$ ./sum3 1 10000000
Sum of integers between 1 and 10000000 is 50000005000000
$ ./sum3 1 100000000
Sum of integers between 1 and 100000000 is 5000000050000000
$ ./sum3 1 1000000000
Sum of integers between 1 and 1000000000 is 500000000500000000
$ ./sum3 1 10000000000
Sum of integers between 1 and 10000000000 is 3883139820726120960
```

またまた想定外の結果になってしまいました。

NTT DATA
NTTデータ先端技術株式会社

演習：連続する整数の和を計算するプログラムを探る

long int では扱えない範囲の値を扱う場合、 long long int を使うことができます。さらに、それでも扱えない範囲の値を扱いたい場合、多倍長整数を扱うためのライブラリを使用することも可能です。

しかし、多倍長整数による計算処理はコストが高いため、多倍長整数を使わずに済むのなら、使わずに済ませたいものです。

そのため、どこまでの範囲を扱うかを想定し、その範囲を扱える型の変数を使用して処理を行い、その範囲を超えてしまう場合にはエラーとして扱う必要があります。

© 2018 NTT DATA INTELLILINK Corporation

演習：連続する整数の和を計算するプログラムを探る

```
---------- sum4.c start ----------
#include <stdio.h>
#define is_minus(x) ((x) < 0)

int main(int argc, char *argv[])
{
        long int n, m, tmp1, tmp2, tmp3;
        if (argc != 3 || sscanf(argv[1], "%ld", &n) != 1 || sscanf(argv[2], "%ld", &m) != 1 || n > m) {
                printf("ERROR: Bad arguments¥n");
                return 1;
        }
        tmp1 = n + m;
        tmp2 = m - n + 1;
        tmp3 = tmp1 * tmp2;
        if ((!is_minus(n) && !is_minus(m) && (is_minus(tmp1) || is_minus(tmp2))) ||
            (is_minus(n) && is_minus(m) && (!is_minus(tmp1) || !is_minus(tmp2))) ||
            ((tmp1 != 0 && tmp3 / tmp1 != tmp2) || (tmp2 != 0 && tmp3 / tmp2 != tmp1))) {
                printf("ERROR: Overflow¥n");
                return 1;
        }
        printf("Sum of integers between %ld and %ld is %ld¥n", n, m, tmp3 / 2);
        return 0;
}
---------- sum4.c end ----------
```

© 2018 NTT DATA INTELLILINK Corporation

演習：連続する整数の和を計算するプログラムを探る

実行結果例：

```
$ ./sum4 1 100000000
Sum of integers between 1 and 100000000 is 5000000050000000
$ ./sum4 1 1000000000
Sum of integers between 1 and 1000000000 is 500000000500000000
$ ./sum4 1 10000000000
ERROR: Overflow
$ ./sum4 1 3037000499
Sum of integers between 1 and 3037000499 is 4611686016981624750
$ ./sum4 1 3037000500
ERROR: Overflow
```

故障解析は、想定外の状況が発生した、あるいは、何かエラーが発生した場合に、何が間違っているのかを探し、修正方法や回避方法を探す作業です。

© 2018 NTT DATA INTELLILINK Corporation

演習：AKARI でシステム全体の動作を解析する。

システムの動作は連係プレーです。入力としてファイルや標準入力などを受け取り、何かの処理を行って、出力としてファイルや標準出力などに渡します。また、必要に応じて、他のプログラムを呼び出します。

そのため、プログラムの呼出しとファイルの読み書きを追跡することで、システム全体の動作をある程度知ることができます。

何処に何があるのか？
ファイルの場所と保存されている情報を探ってみましょう。

NTT DaTa
NTTデータ先端技術株式会社

演習：AKARI でシステム全体の動作を解析する。

カーネル起動時のコマンドライン引数に init=/sbin/akari-init というパラメータを追加して起動してみてください。

```
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  9b5c2db1-f\
b69-4265-a710-69d0adb14585
        else
          search --no-floppy --fs-uuid --set=root 9b5c2db1-fb69-4265-a710-69d0\
adb14585
        fi
        linux16 /boot/vmlinuz-3.10.0-693.17.1.el7.x86_64 root=UUID=9b5c2db1-fb\
69-4265-a710-69d0adb14585 ro crashkernel=auto sysrq_always_enabled LANG=en_US.\
UTF-8 init=/sbin/akari-init
        initrd16 /boot/initramfs-3.10.0-693.17.1.el7.x86_64.img


      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

起動が完了したら、ログインして、 /usr/sbin/ccs-editpolicy というコマンドを実行してみてください。

演習：AKARI でシステム全体の動作を解析する。

この画面における親子関係は、呼び出す側のプログラム／呼び出される側のプログラムを表しています。例えば、 systemd というプログラムは netconsole や network というプログラムを呼び出しています。そして、 netconsole というプログラムは、 cat や grep や sed や modprobe などのプログラムを呼び出しています。

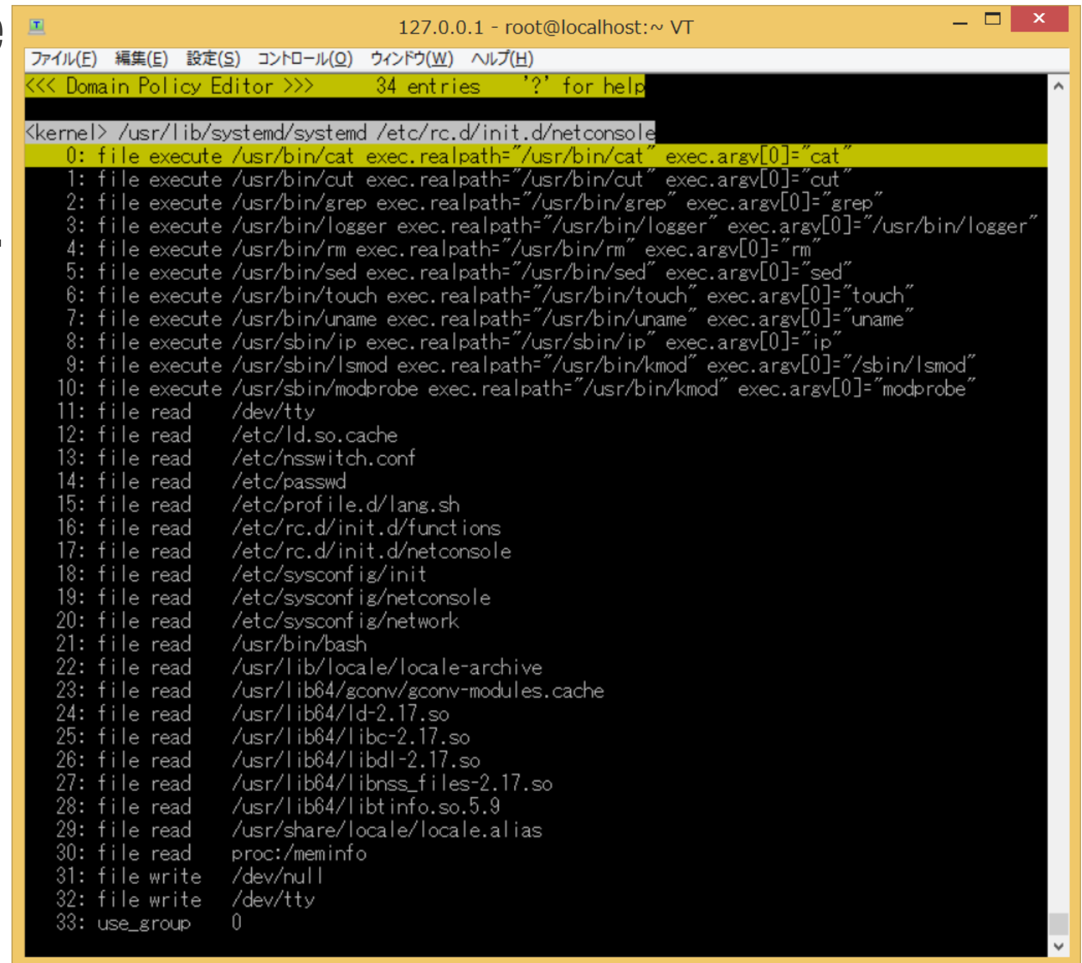
```
<<< Domain Transition Editor >>>       242 domains     '?' for help

<kernel>
    0:   1       <kernel>
    1:   1           /usr/lib/systemd/systemd
    2:   1               /etc/rc.d/init.d/netconsole
    3:   1                   /usr/bin/cat
    4:   1                   /usr/bin/cut
    5:   1                   /usr/bin/grep
    6:   1                   /usr/bin/logger
    7:   1                   /usr/bin/rm
    8:   1                   /usr/bin/sed
    9:   1                   /usr/bin/touch
   10:   1                   /usr/bin/uname
   11:   1                   /usr/sbin/ip
   12:   1                   /usr/sbin/lsmod
   13:   1                   /usr/sbin/modprobe
   14:   1               /etc/rc.d/init.d/network
   15:   1                   /etc/sysconfig/network-scripts/ifup
   16:   1                       /etc/sysconfig/network-scripts/ifup-eth
   17:   1                           /etc/sysconfig/network-scripts/ifup-ipv6
   18:   1                               /usr/bin/cat
   19:   1                               /usr/bin/dbus-send
   20:   1                               /usr/bin/grep
   21:   1                               /usr/bin/hostname
```

© 2018 NTT DATA INTELLILINK Corporation

演習：AKARI でシステム全体の動作を解析する。

この画面は、特定のプログラムが行っている操作（他のプログラムの呼び出しやファイルのオープンなど）の内容を示しています。

/etc/rc.d/init.d/netconsole というプログラムは、 cat や grep や sed や modprobe などのプログラムを呼び出す他に、書き込みモードで /dev/null をオープンしたり、読み込みモードで /etc/sysconfig/netconsole をオープンしたりしていることが確認できます。



© 2018 NTT DATA INTELLILINK Corporation

演習：Web サーバを例に、ファイルの種類について確認する。

ファイルには、それぞれ目的があります。

　プログラム・データ・設定ファイル・ログファイル・その他

その目的は、ファイルの存在する場所／ファイル名により決まります。

CentOS 7 における Apache Web Server のパス名は /usr/sbin/httpd です。
AKARI を用いて、 /usr/sbin/httpd というプログラムがどのようなファイルを操作しているかについて確認してみましょう。そして、それぞれのファイルについて、どのような目的のファイルなのかを確認してみましょう。

© 2018 NTT DATA INTELLILINK Corporation

OSSとは？

プログラムのソースコードが公開されているソフトウェアです。

誰でも開発に参加できるから、変化が速いです。

ソースコード公開／非公開

ソースコード公開

ソースコード非公開

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ 先端技術株式会社

OSSとは？

誰もが全ての仕様／内容を理解している訳ではありません。

使いこなすには、技術力／サポート力が必要です。

OSSの基本スタンス

「最新版にアップデートしない環境の面倒は見ない」

© 2018 NTT DATA INTELLILINK Corporation

Linux ディストリビューションとは？

一般の利用者がインストールして使いやすように、OSSをパッケージ化したものです。ディストリビューションの方針に沿って、

どのようなソフトウェア／どのバージョンを提供するか
どのようなサポートを提供するか

などが決まります。

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ 先端技術株式会社

Linux ディストリビューションとは？

ディストリビューションの基本スタンス

最新版を追いかけ続けるものや、互換性／安定性を重視するものなど、いろいろあります。

RHEL / CentOS などのエンタープライズ向けディストリビューションの場合、（サポート対象となる期間の間は）「互換性を維持する努力」をしています。しかし、人的リソースの制約があるので、基本的に「最新版にアップデートしない環境の面倒は見ない」という点では、OSSと同様です。

アップデートしないという選択肢は無い！？

他の人が作ったソフトウェアを共同利用させてもらう以上、自分が必要な修正だけを取り込んだバージョンを提供してもらうことはできません。そのため、常に最新版にアップデートすることが前提となります。つまり、アップデートすることを前提に、システムを設計していくことが必要になります。アップデートにより、予期せぬリグレッションに遭遇する可能性もあります。

でも、セキュリティ上の理由（システムへの侵入防止）により、アップデートしないという選択肢は困難になってきています。その時はアップデートしないで済んだとしても、いつかはアップデートしなければいけなくなる可能性もあります。

アップデートしないという選択肢は無い！？

実際、2018年1月、 Spectre / Meltdown という脆弱性への対応として、皆が一斉に Linux カーネルのアップデートを行いました。その結果、様々なリグレッションに遭遇し、サポートを提供する側は大変な思いをしました。

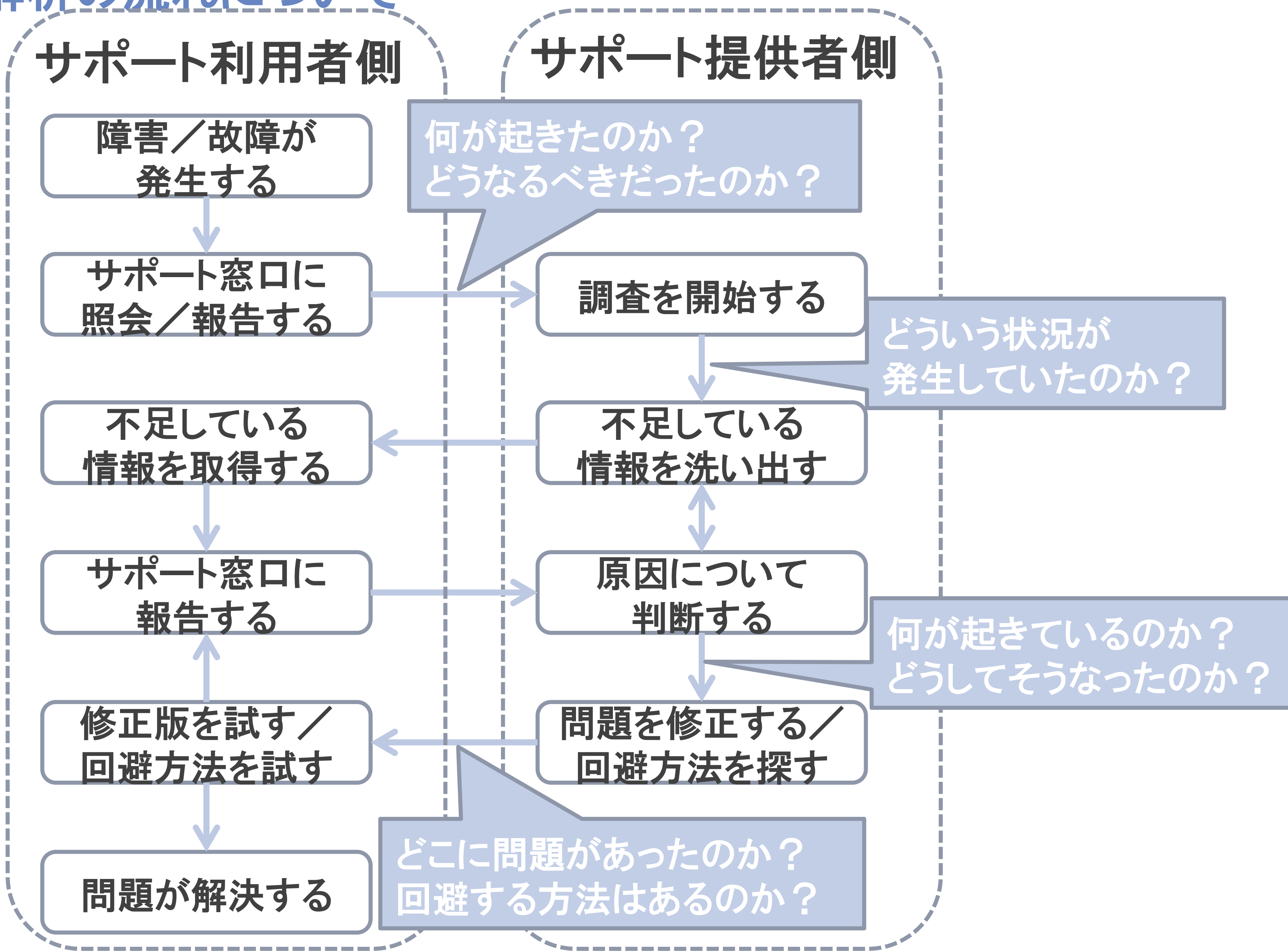
だから、システムの動きを理解して、いざという時に慌てないで済むように備えましょう。

© 2018 NTT DATA INTELLILINK Corporation

故障解析に必要な情報について

- 環境情報(ソフトウェアの構成情報)
  - プログラム自体はOSSなら入手可能なので不要
- 設定ファイル
- ログファイル

故障解析の流れについて



© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ 先端技術株式会社

故障解析のもどかしさ

- 実物を触ることができない辛さ／同じ環境を用意することの困難さ

システム管理者のパスワードを教えてもらって直接触る訳にはいきません。

どこで問題が起きているのかを把握できないと、再現環境を用意できません。

ハードウェア依存だと、環境を用意することも難しくなります。

© 2018 NTT DATA INTELLILINK Corporation

故障解析のもどかしさ

- 情報を取得することの難しさ

調査に必要になる情報を取得するための壁があります。何かのログを取得するのに5分でできる環境もあれば、各種組織と調整するなどで1か月かかる場合もありえます。

また、同じ故障が再発するまで情報を取得できないケースもあります。

セキュリティ上の理由(情報漏えい対策)により、情報管理の徹底が求められています。そのため、ファイルを共有することも容易ではありません。

© 2018 NTT DATA INTELLILINK Corporation

故障解析のもどかしさ

- 質問者の悩み：優先順位順／非同期処理なので進捗が予測できない

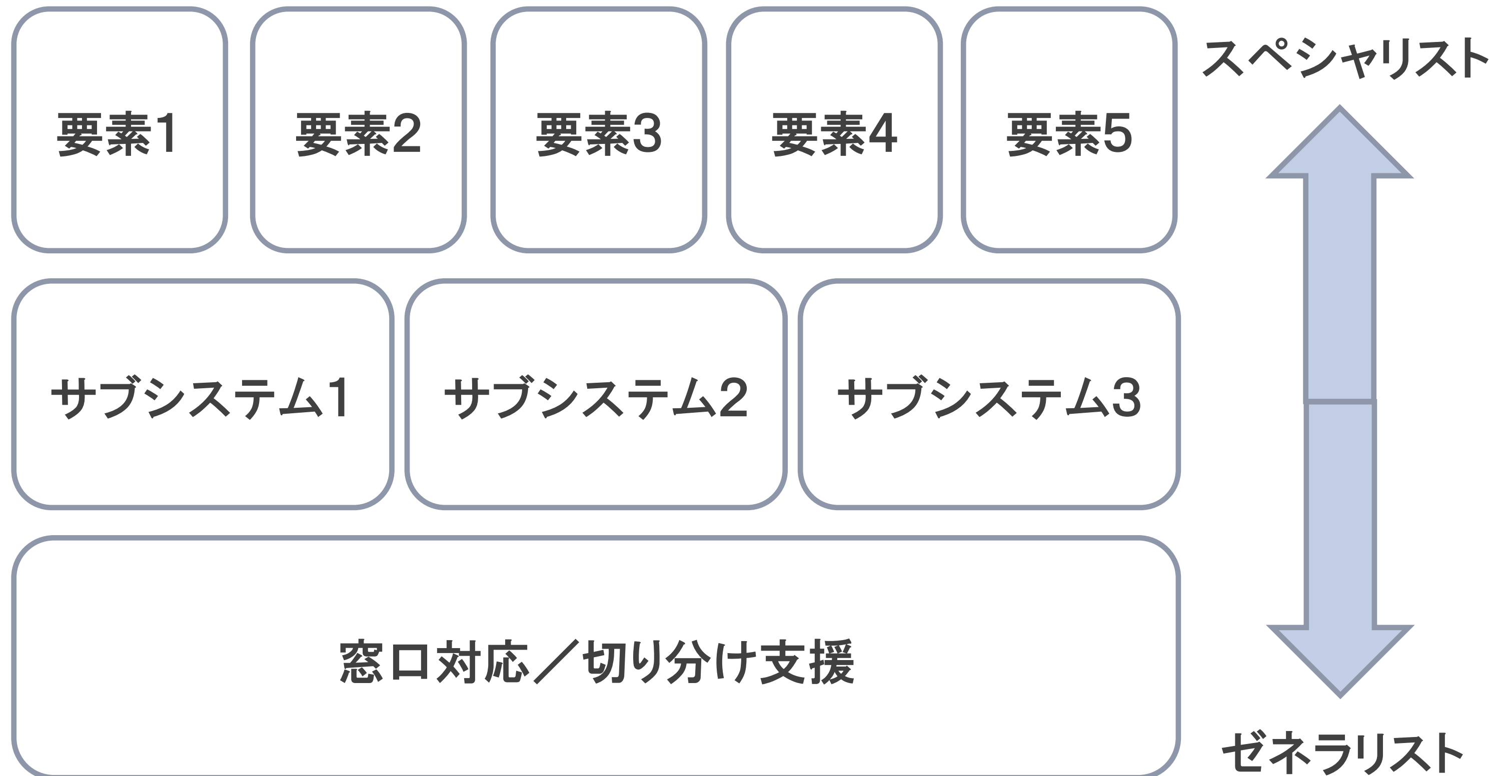
回答する（サポートを提供する）側は、捌ききれないくらいのサポート要求を抱えています。そのため、いかにサポートを提供する側の手を煩わせないようにできるかは、問題が解決するまでの期間に大きな影響を与えます。

- 回答者の悩み：質問者の状況やスキルレベルを把握できない

同様に、質問する（サポートを利用する）側にもいろいろな制約があります。そのため、いかにサポートを利用する側の手を煩わせないようにできるかは、利用者による評価に大きな影響を与えます。

故障解析のもどかしさ

余談：スペシャリストは担当の領域に関してはすごいスキルを持っていますが、超多忙であり、担当外の領域についても詳しくありません。そのため、ゼネラリストが切り分けを行い、必要に応じてスペシャリストの助けを借りるという運用が行われます。



© 2018 NTT DATA INTELLILINK Corporation

故障解析のもどかしさ

- 人が病気で医者に行く場合とは勝手が違うのです。

患者自らが医者に診てもらえる訳ではありません。
システム管理者自身が行う必要があります。
厳しい制約の中で如何に上手に伝えられるかが鍵となります。

- だからこそ、経験しておくことが重要です。

演習：Bugzilla などで公開されている対応事例を辿る

問題のインパクトを示す事例：

Bug 749909 - Inbox destroyed by incomplete rename error handling. If localized folder name is used, original special folder file(e.g. Inbox) is deleted when rename of a folder to "English special folder name of different case"(e.g inbox) is requested.
https://bugzilla.mozilla.org/show_bug.cgi?id=749909

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ先端技術株式会社

演習：Bugzilla などで公開されている対応事例を辿る

問題となる事象の再現手順を説明する事例：

Bug 1022845 - binutils: nm -D does not process subsequent files after "No symbols".
https://bugzilla.redhat.com/show_bug.cgi?id=1022845

Bug 1056828 - strace -cf java causes Segmentation fault
https://bugzilla.redhat.com/show_bug.cgi?id=1056828

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ先端技術株式会社

演習：Bugzilla などで公開されている対応事例を辿る

問題となる事象の再現手順を説明する事例：

Bug 1181649 - makedumpfile: User process data pages are not excluded appropriately.
https://bugzilla.redhat.com/show_bug.cgi?id=1181649

Bug 1116237 - find -perm +numeric does not work as expected
https://bugzilla.redhat.com/show_bug.cgi?id=1116237

Bug 1374495 - Firefox crashes when opening too large page.
https://bugzilla.mozilla.org/show_bug.cgi?id=1374495

© 2018 NTT DATA INTELLILINK Corporation

演習：Bugzilla などで公開されている対応事例を辿る

アップストリームに無いパッチによる不具合の事例：

BUG: unable to handle kernel paging request at ffffdf3cd60001a0
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1734686

脆弱性修正によるリグレッションと勘違いした事例：

System Hangs and General Protection Fault Occurs following NFS Access
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1742572

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ 先端技術株式会社

演習：Bugzilla などで公開されている対応事例を辿る

不具合だと思ったら想定通りの挙動だった事例：

Kernel segfault playing EVERSPACE
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1739928

簡単な説明でも的確に伝わる事例：

linux 3.13.0-140.189 - kernel panic after trying to launch any 32-bit application
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1744071

© 2018 NTT DATA INTELLILINK Corporation

NTT DATA
NTTデータ 先端技術株式会社

演習：Bugzilla などで公開されている対応事例を辿る

切り分けの難しさ（不具合だと思ったら想定通りの挙動だった＋本当に不具合かも）を示す事例：

Bug 1525356 - qemu quit silently after virtio-balloon of deadlock on OOM occurred
https://bugzilla.redhat.com/show_bug.cgi?id=1525356

切り分けの難しさ（タイミング依存）を示す事例：

[x86? mm? fs? 4.15-rc6] Random oopses by simple write under memory pressure.
http://lkml.kernel.org/r/201801052345.JBJ82317.tJVHFFOMOLFOQS@I-love.SAKURA.ne.jp

© 2018 NTT DATA INTELLILINK Corporation

演習: sosreport を解析する

一般的に、問題となる事象をシステム管理者が意図的に再現させることは、技術レベルの問題から困難な場合が多いです。だから、故障解析を行う側は、ログファイルなどに記録されたメッセージから、どのような事象が発生していたのかを追跡していくという調査手法を使用することになります。

RHEL/CentOS では、環境についての情報や、典型的な設定ファイル／ログファイルなどを収集するためのプログラムとして sosreport というコマンドが提供されています。ログインして、root ユーザとして以下のコマンドを実行してみてください。

```
# sosreport --batch -k rpm.rpmva=off
```


© 2018 NTT DATA INTELLILINK Corporation
NTT Data
NTTデータ先端技術株式会社

演習: sosreport を解析する

sosreport コマンドにより生成されたファイルの内容を確認し、どのような情報が収集されるのか、および、マスクすべき情報と共有すべき情報について考えてみましょう。

例:ドット区切りの整数の並び(例: 1.2.3.4)→マスクすべき情報?

NTT DATA
NTTデータ先端技術株式会社

スムーズな問題解決をするために

「何処で問題が発生しているのか」は「何処に問合せをすべきか」を判断するのに必要です。

業務固有(高レイヤー／応用)の処理なのか、共通プログラム(低レイヤー／基盤)の処理なのか？業務固有の処理は守秘義務などの制約により共有できないことが多いです。仮に共有できたとしても、プログラムの提供側が理解できる内容とは限りません。

プログラムの提供側が容易に理解できる再現プログラムを確立できれば、問題をより迅速に確実に解決できる可能性が高くなります。(場合によっては、サポート窓口を介さずに、プログラムの作者に対してダイレクトに照会することも可能です。)

© 2018 NTT DATA INTELLILINK Corporation

参考：Linux カーネルのデバッグ支援について

Linux カーネルは非常に大規模で複雑なプログラムであるため、いつも不具合と格闘しています。

そのため、不具合を検出するための様々なプロジェクトが存在しています。その中から、 syzkaller/syzbot について眺めてみましょう。

https://github.com/google/syzkaller/blob/master/docs/syzbot.md

© 2018 NTT DATA INTELLILINK Corporation

参考：Linux カーネルのデバッグ支援について

syzkaller は、システムコール(system call ／カーネルが提供する機能を呼び出すこと)に関して、様々なテストケースを生成してテストを行うことで不具合を見つけ出し、不具合についてのレポートを生成し、不具合を再現するためのプログラムの作成(≒問題事象の発生条件の特定)を試みるためのプログラムです。

syzbot は、 syzkaller の動作を制御し、 syzkaller の報告内容を整理し、不具合の状態を管理し、利用者に対してWebインタフェースを提供するためのプログラムです。

syzbot が「サポート利用者側」に、カーネル開発者が「サポート提供者側」に対応します。(syzbot がデバッグを支援してくれているという点では「サポート提供者側」でもあります。)

© 2018 NTT DATA INTELLILINK Corporation

演習：問題の発生個所／発生条件を切り分ける

最後に、既知の問題を利用して、発生個所／発生条件を切り分ける練習をしてみましょう。仮想マシンには、 Apache Web Server と、そこから実行される幾つかのCGIプログラムがインストールされています。この中に、特定の条件下で問題が発生するような処理が含まれていますので、それを見つけ出して、私に報告してください。

なお、仮想マシン（ゲスト）側で発生したカーネルメッセージは netconsole を経由して Windows （ホスト）側へと転送されるような設定が行われています。そのため、 Windows 側で udplogger-x86.exe を稼働させることで、カーネルメッセージを保存できるようにしておくと、何が起きているのかを知るためのヒントが得られるかもしれません。

© 2018 NTT DATA INTELLILINK Corporation

NTT DaTa
NTTデータ先端技術株式会社

まとめ

システムにはトラブルが付き物です。今日は、サポートセンタでもどかしい思いをした経験から、スムーズな問題解決を行えるようになるために、トラブルへの対処方法の基本について考えてみました。今日紹介しきれなかった内容については、OSSコラム「安らかな夜を迎えるために」を参照していただければと思います。
http://www.intellilink.co.jp/article/column/oss/index.html

今日やった内容は、一度聞いて全て理解できるようなものではありません。技術的な詳細は、理解できていなくて当然です。でも、問題が発生した際に、それを相手に伝えて、問題を解決する上で、どのようなことに注意すればよいのかについて、何かを感じていただけたのではないかと思います。それは、ちょっとした「想像力」と「気配り」かもしれません。

© 2018 NTT DATA INTELLILINK Corporation

© 2018 NTT DATA INTELLILINK Corporation