クラウド基盤構築演習

第一部
クラウド基盤を支えるインフラ技術
～ 第5回 Linux問題判別と内部構造入門

ver1.1 2012/05/01

目次

- Linux問題判別の概要
- システムログの収集
- Linuxのプロセス管理
- Linuxのメモリ管理
- 参考: 問題判別の基本コマンド

Linux問題判別の概要

問題判別の心構え

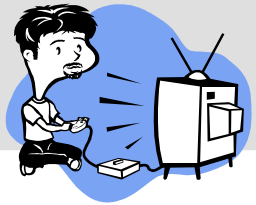
- ITシステムで発生する事象には、すべて原因があります。問題発生時は、まずは深呼吸をして、冷静な判断を心がけてください。

  - 不可解な問題に直面した際は、思い込みを排除して、論理的に原因を追求していきます。
  - ほとんどの問題は、すでに一度は誰かが直面して解決しているはずです。広く情報収集に努めてください。
  - 「あなた自身の力で問題を解決すること」は重要ではありません。適切な専門家に依頼して問題解決に導くこともシステム管理者の大切な役割です。

ここは暗闇に包まれた自席のパーティションです。北は部長の部屋で、西はチームリーダーのパーティションです。東には開いた窓があり、5階からの景色が見えています。南は給湯室です。(コーヒーメーカーには出来立てのコーヒーが入っています。)あなたが端末のスクリーンをにらみつけている間に、5度目の電話の呼び出し音が鳴っています。(恐らく、サーバーに接続できないユーザーからの苦情です。)
コマンド?>見る_

机の上には書類と飲み終えたコーヒーの紙コップが積み上げられています。電話機のメッセージ保存ランプは3ヶ月前から点灯したままです。Eメールの受信箱は未読メッセージで溢れています。
机の上の書類の隙間から、来週に受講予定の新しい研修テキストが見えます。
-『Linux問題判別と内部構造入門』
コマンド?>読む
何を?>Linux問題判別と内部構造入門_

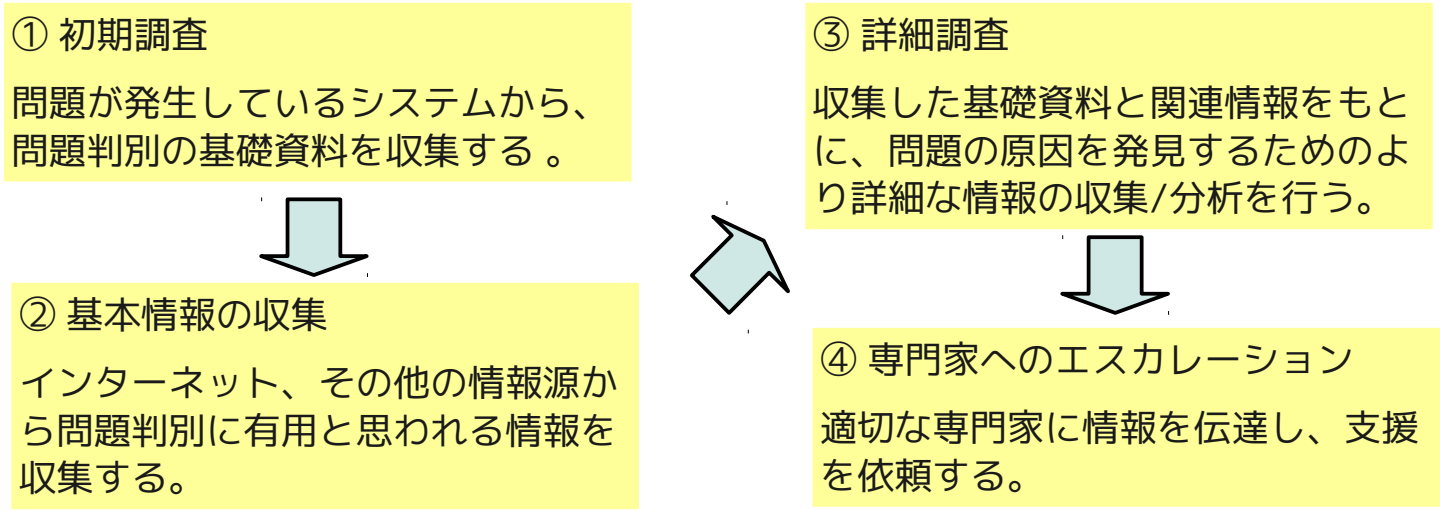
帰宅して仮眠した方がよさそうです。
コマンド?>_



問題判別の流れ

- 問題判別を実施する際は、次の点を常に意識しておきます。
  - 早急な問題の解決　→ その作業は、問題の解決に関係がありますか?
  - 次に活かせる経験の習得　→ 問題判別は 1 回限りの仕事ではありません。
  - 適切な専門家の有効活用　→ 専門家を『道具』として活用するのもスキルが必要です。

- 問題判別は大きく 4 つの段階に分けて実施します。
  - 専門家に対応を依頼する場合でも、①~③を適切に実施した上で依頼することで、問題解決のまでの時間が短縮できます。本テキストでは、専門家へのエスカレーションを効率的に実施して、迅速な問題解決を行うための基礎知識を提供します。



『初期調査』の進め方 (1)

- まずは、「ユーザからの情報収集」と「システムの基礎資料の収集」を行います。
  - この段階では、思い込みを排除して、広く、客観的な情報の収集に努めます。断片的な情報で拙速な判断をしないことが大切です。

- 「ユーザからの情報収集」では、問題を発見した利用者の立場から現状を把握します。

① なぜ問題に気付いたのか。本来はどのような動作をするべきなのか。以前はどのように動作していたのか。

  - エラー表示画面のスナップショットや、入力したコマンドと出力結果のログファイルなど、編集の入らない生の情報を集めるように心がけます。
    - 文章による表現は情報の欠落が発生します。下はどちらも『fdisk でディスクが見えません』と言えますが、状況はまったく異なります。

```
# fdisk -l /dev/sdb
Cannot open /dev/sdb
```

```
# fdisk -l /dev/sdb
（何も表示されない）
```

  - ユーザが正常だと思っている動作と、システムとして本来あるべき、正常な動作は往々にして異なります。過去の時点から、本来の正常動作が行われていなかった可能性もあります。

『初期調査』の進め方 (2)

② 問題が発生した時は何をしていたのか。問題に気付いた後で何をしたのか。

- 直前の操作が、問題の発生に直接関係する事がよくあります。
- ユーザは自分が問題に関係すると思った情報だけを伝えてきます。できるだけ客観的な情報を引き出すことが大切です。
- システムログやアプリケーションログを解析する際に、問題の発生部分を特定するためにも必要な情報です。

③ その問題はいつから発生しているのか。他のシステムを含め、関連すると思われる他の問題はあるのか。

- 過去のシステム構成の変化が、問題に関係する可能性を確認するため、システム構成、システム負荷、システム利用方法などの変化を次に説明する基礎資料として収集します。
- 特定の環境のみで発生する問題かどうか、という事実も貴重な情報です。

『初期調査』の進め方 (3)

- 「システムの基礎資料の収集」では、システム全体の構成を把握する資料を集めます。

① HW 構成、SW 構成、OS 設定ファイル、ミドルウェア/アプリケーション設定ファイル

- 既存の設計資料の入手、および、ツールによる現状データの収集を行います。
  - この後の「OS構成情報の収集ツール」も参照。

② 問題の証拠となるエラーメッセージ、エラーログファイル、システムログファイル

- OS のリソース情報については、ツールによる定期的な収集も効果的です。
  - この後の「sysstatによるシステム稼働情報の収集」も参照。
- ミドルウェアなどは独自の情報収集コマンドを持っている場合があります。

③ 問題管理記録、変更管理記録

- 過去の問題やシステム変更が、今回の問題と関連することはよくあります。
- 日頃からの問題管理プロセス/変更管理プロセスの整備が大切です。

『基本情報の収集』の進め方

- この段階では、次の『詳細調査』で、問題の原因についての仮説を立てるために有用な情報を収集します。 Linux の問題に関する情報は、インターネット、その他の情報源から広く入手することができます。
  - 『初期調査』で入手した情報に基づいて、同一の問題の事例について検索します。
    - 原因の仮説を立てるための情報を収集する段階ですので、何らかの類似性がある問題の情報は一通り集めておくのが得策です。
  - サポート契約がある場合は、契約に基づいてベンダーに調査、情報提供を依頼します。
    - 思い込みや仮説を排除して、『初期調査』で入手した客観的な情報を元に、調査を依頼します。
- FAQ や Bugzilla、HowTo 文書から関連する情報が得られることもあります。
  - 日頃からこれらの情報源のリンクを収集、整理しておくことをお勧めします。
  - 製品の公式マニュアルを参照することも忘れないでください。

『詳細調査』の進め方

- この段階で必ず問題を解決しないといけないわけではありません。『専門家へのエスカレーション』に向けた最善の準備をする段階でもあることを意識してください。
  - 問題に関連する、より本質的な情報の収集を試みます。
    - 問題に関する仮説を証明するには、システムログに加えて、コマンドや専用のツールを利用した情報収集が必要となります。
  - すべての行為の正確な記録を残します。
    - 問題の原因の予想、試みた変更、その結果など、問題判別に関連するすべての作業を記録に残します。これは、『専門家へのエスカレーション』の際に必要な情報となります。
    - なるべく定性的な表現を避け、場合によっては、すべての入力コマンドと結果の画面出力のレベルで記録をします。
  - コンポーネントや階層に分けて考えます。
    - 問題の範囲をコンポーネントや階層に分けて考え、問題の原因についての仮説を立てます。その仮説が『正しい』、もしくは、『正しくない』事を証明することで、問題の範囲を狭めていきます。 多くの場合、『情報収集』の段階で得られた情報を元に仮説を立てることになります。
  - 事実と仮説の区別を常に明確にしておきます。
    - (『正しい』とも『正しくない』とも)証明されていない仮説を事実と混同しない事が大切です。特に、他の情報源から得られた情報は、問題の起きているシステムにとっても事実と言えるのか、あるいは仮説に過ぎないのか、常に意識してください。
    - 証明された仮説から得られる事実についても、思い込みが混入しないよう、常に論理的に判断してください。

OS構成情報の収集ツール

- sosreportは、RHELに標準で付属するOS構成情報の収集ツールです。OSから認識されているハードウェア情報なども取得します。

```
# sosreport

sosreport (version 2.2)

This utility will collect some detailed  information about the
hardware and setup of your Red Hat Enterprise Linux system.
The information is collected and an archive is  packaged under
/tmp, which you can send to a support representative.
Red Hat Enterprise Linux will use this information for diagnostic purposes ONLY
and it will be considered confidential information.

This process may take a while to complete.
No changes will be made to your system.

ENTER を押して継続するか、又は CTRL-C で終了します。
```

  - 指示にしたがって[Enter]を入力していくと、tar.xz形式のアーカイブファイルが作成されます。これは次のコマンドで展開できます。
    - tar -xvJf sosreport-*.tar.xz
  - アーカイブファイルには、主要な設定ファイル/ログファイルのコピーと情報収集コマンドの出力がまとめられています。

```
# ls -xp
boot/        chkconfig      date        df              dmidecode  etc/
free         hostname       ifconfig    installed-rpms  java       lib/
lsb-release  lsmod          lsof        lspci           mount      netstat
proc/        ps             pstree      root/           route      sar08
sestatus     sos_commands/  sos_logs/   sos_reports/    sys/       uname
uptime       var/
```

sysstatによるシステム稼働情報の収集

- sysstatパッケージにより、OSのシステム稼働情報を定期的に自動収集することができます。
  - sysstatのRPMパッケージを導入すると、設定ファイル「/etc/cron.d/sysstat」にcronジョブのエントリが追加されます。
  - 10分毎に1分間の統計情報をバイナリファイル「/var/log/sa/saXX」に収集した後、毎晩23:53に1日分の統計情報をまとめたテキストファイル「/var/log/sa/sarXX」を作成します。
    - XX はデータの日付けの ”日” の数字で、1 ヶ月前までのファイルが保管されます。
  - バイナリファイルに含まれるデータはsarコマンドで内容を確認します。-fオプションによるファイル名指定を省略した場合は、当日のデータ収集中のバイナリファイルを対象とします。
    - sar -u -f /var/log/sa/saXX ⇒ CPU使用状況
    - sar -r -f /var/log/sa/saXX ⇒ メモリ使用状況
    - sar -b -f /var/log/sa/saXX ⇒ ディスクI/Oの状況
    - sar -W -f /var/log/sa/saXX ⇒ スワップ・イン、スワップ・アウトの発生状況
    - sar -n DEV -f /var/log/sa/saXX ⇒ ネットワークパケット送受信の状況
    - sar -A -f /var/log/sa/saXX ⇒ 全てのデータ

メモとしてお使いください

メモとしてお使いください

システムログの収集

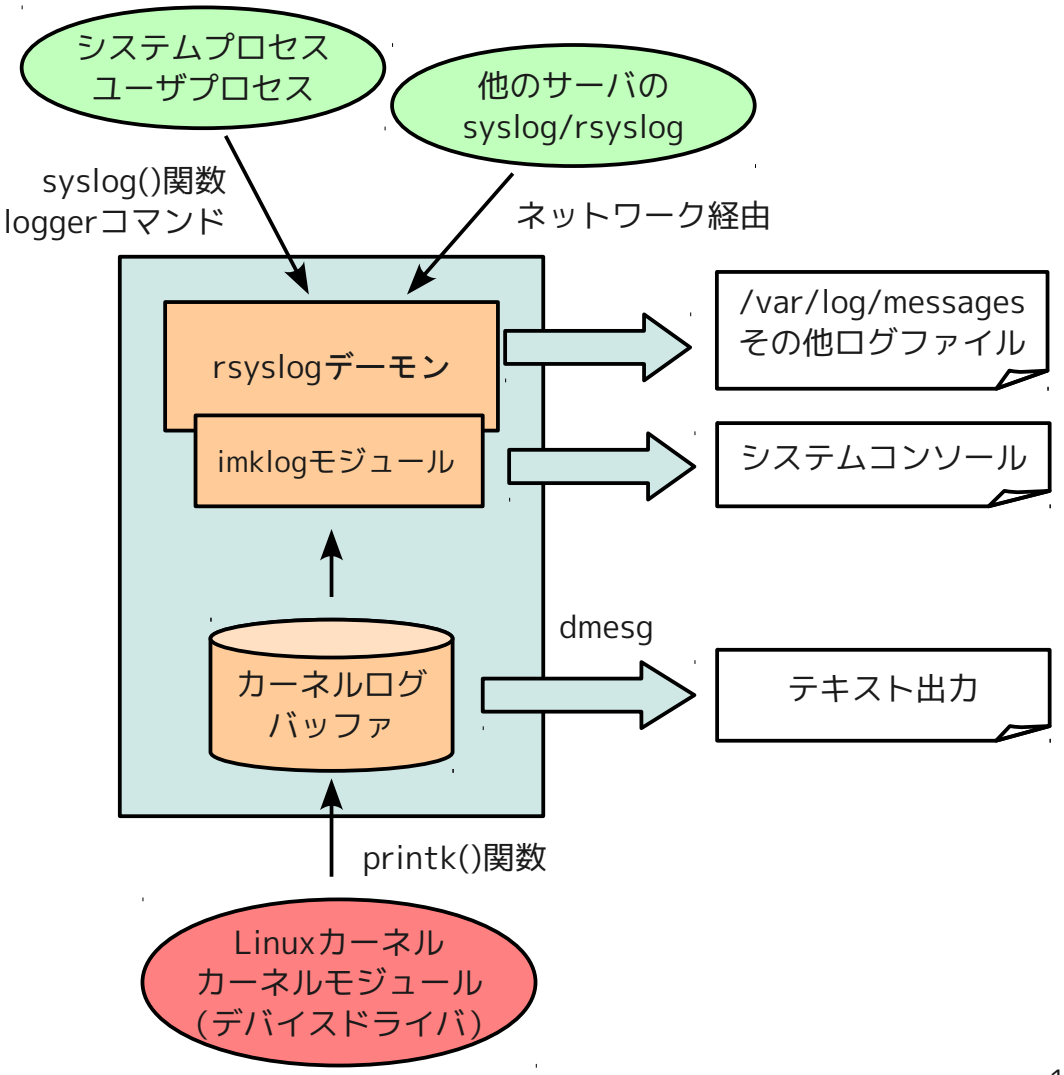
Linuxの主なログファイル

- Linuxの問題判別でよく参照されるシステムログ(RHEL6の例)です。

ログファイル	説明
/var/log/messages	システム関連ログのデフォルトの出力ファイルです。
/var/log/secure	ユーザのログイン認証など、セキュリティ関連の情報が記録されます。
/var/log/audit/audit.log	auditdによるシステム監査ログが記録されます。
/var/log/boot.log	システム起動時の起動スクリプトが出力するログです。主に、サービスの起動・停止に関する情報が記録されます。
/var/log/dmesg	システム起動直後のカーネルログバッファの内容が記録されます。 (カーネルログバッファについては後述)
/var/log/cron	cronジョブの実行履歴が記録されます。
/var/log/Xorg.n.log	X Windowサーバのログファイルです。（nはディスプレイ番号に対応）
/var/log/wtmp	ユーザのログイン履歴を記録するバイナリファイルです。lastコマンドで参照します。
/var/log/lastlog	ユーザの最終ログイン記録を保管するバイナリファイルです。lastlogコマンドで参照します。
/var/run/utmp	現在ログイン中のユーザ情報を記録するバイナリファイルです。uptime、wコマンドなどが利用します。

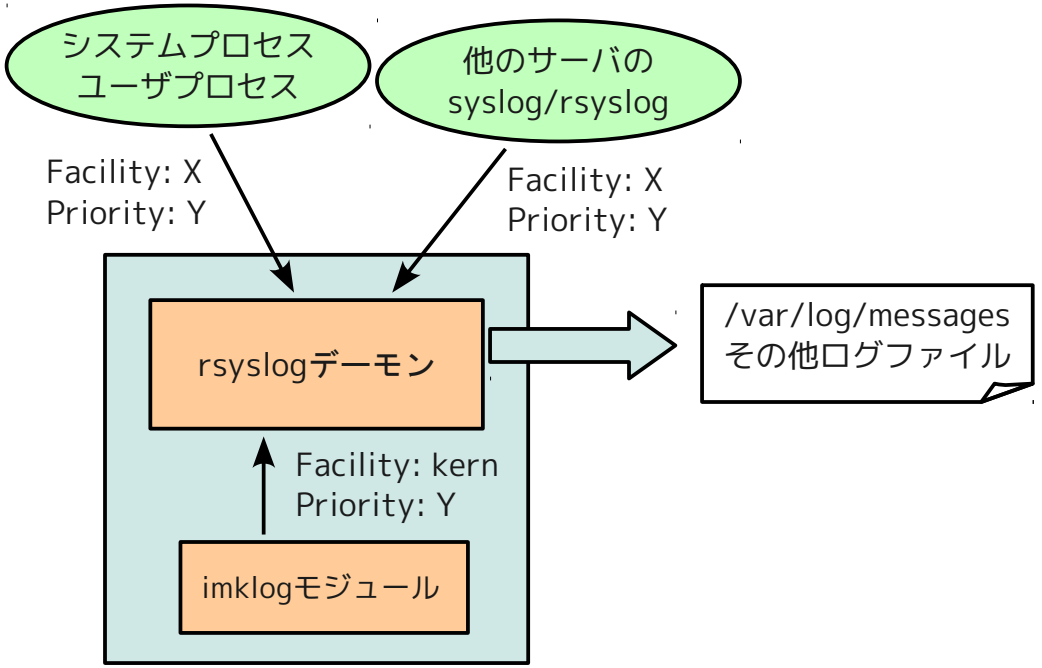
Linuxのロギングシステム概要

- Linuxのロギングシステム3つのコンポーネントで構成されます。
  - カーネルログバッファ
    - Linuxカーネル、およびカーネルモジュール（デバイスドライバなど）が出力するログを一次保存するメモリ上のバッファ領域です。
  - imklogモジュール
    - rsyslogデーモンのプラグインモジュールで、カーネルログバッファの内容をrsyslogデーモンに転送します。
    - 緊急度の高いメッセージはシステムコンソールにも出力します。
  - rsyslogデーモン
    - 一般のプロセスのログメッセージ、およびimklogモジュールから転送されたカーネルメッセージをログファイルに出力します。
    - 他のサーバのsyslog/rsyslogデーモンのメッセージをネットワーク経由で受信することもできます。



SyslogメッセージのFacilityとPriority

- rsyslogデーモンが受けるメッセージには、FacilityとPrioiryが付与されています。
  - Facility: メッセージの種類を分類します。
    - auth, authpriv, cron, daemon, lpr, mail, news, security (auth の別名), syslog, user, uucp, mark：システム規定のエントリです。
    - kern：imklogモジュールから転送されるカーネルメッセージに使用します。
    - local0, local1, local2, local3, local4, local5, local6, local7：規定のエントリーに該当しないメッセージについて、アプリケーションが任意に利用します。
  - Prioirty: メッセージの緊急度（重要度）を表します。
    - debug, info, notice, warning, warn (warningの別名), err, error (errの別名), crit, alert,emerg, panic (emergの別名)：この順に緊急度(重要度)が高くなります。(debugが低く、emergが高い。)
- これらに基づいて出力先のログファイルが決まります。
  - 設定ファイル「/etc/rsyslog.conf」で設定します。



rsyslog.confの設定

- rsyslogの設定は「/etc/syslog.conf」で行います。
  - メッセージのFacility と出力先のログファイルを結び付けます。また、Priority を指定して、指定以上の緊急度(重要度)のメッセージのみを出力します。
  - 例：
    - *.emerg: 任意のFacilityで、emerg以上の緊急度。
    - mail.*: Facilityが mailの全てのメッセージ。
    - *.info; mail.none: 任意のFacilityで、info 以上の緊急度。ただし、Facilityがmailのものは除く。
- loggerコマンドを利用すると、FacilityとPriorityを指定してrsyslogデーモンにメッセージを送信できます。
  - 次は実行例です。

```
# logger -p local0.info -t TEST 'Hello, World!'
# tail -1 /var/log/messages
Apr 2 17:30:44 server01 TEST: Hello, World!
```

rsyslog.confの例

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

(*)「/etc/rsyslog.conf」に「$ActionFileEnableSync on」を指定するとログファイルの書き出しごとにディスクキャッシュをフラッシュします（デフォルトはoff）。「-/var/log/maillog」の頭の「-」は、上記オプションがonの場合でも、このファイルについてはディスクキャッシュのフラッシュをしない事を指定します。

カーネルログバッファの役割

- Linux カーネル、およびカーネルモジュール(デバイスドライバなど)のログメッセージは、一旦、カーネルログバッファ(メモリ上のバッファ領域)に保存されます。
  - これにより、rsyslogデーモンが稼動していない状態(システム起動時のrsyslogが起動する以前の段階など)でもカーネルメッセージを失うことがありません。
    - カーネルログバッファの容量は 128KB で、一杯になると古いものから順に削除されます。
  - dmesg コマンドで、カーネルログバッファの内容を出力できます。システム起動直後のカーネルログバッファの内容は、「/var/log/dmesg」ファイルにも記録されています。

- Linuxカーネル内部では、printk()関数でログメッセージを出力しています。
  - 各ログメッセージには、0 ~ 7のログレベル(優先度)が指定されます(0が高く、7が低い)。
  - カーネルメッセージのログレベルは、imklogモジュールが転送する段階で、8種類のPriorityに変換されます。Faicilityにはkernが指定されます。
    - 0 ~ 7 の優先度が、それぞれ、emerg, alert, crit, err, warning, notice, info, debug に対応します。

- imklogモジュールは、優先度の高いメッセージをシステムコンソールにも出力します。
  - 「/etc/rsyslog.conf」のオプション「$klogConsoleLogLevel」に1～8の値を指定すると、指定した値よりも(値は含まない)高い優先度のメッセージがコンソールに出力されます。

メモとしてお使いください

メモとしてお使いください

Linuxのプロセス管理

プロセスシグナルについて

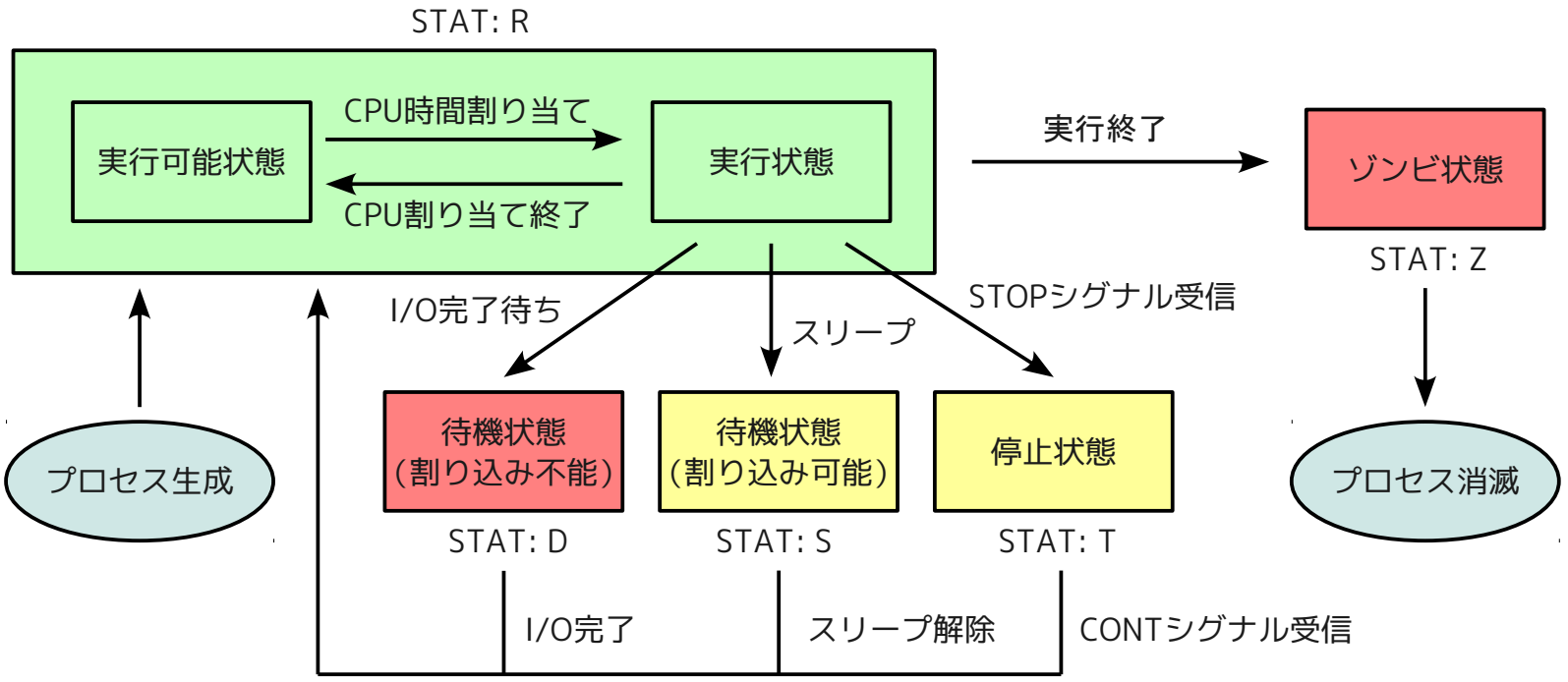
- Linux では、実行中のプロセスに対して「シグナル」を送ることでプロセスの動作を制御することができます。
  - シグナルが発生するのは、主に次の場合です。
    - ユーザが killコマンドでシグナルを送る場合。
    - 実行中のプロセスが他のプロセスに対してシグナルを送る場合。
  - 右表は、Linuxで使用する主なシグナルです。
  - killコマンドでシグナルを送る場合は、シグナル名（もしくはシグナル番号）と送信先プロセスIDを指定します。次の2つは同じ意味になります。
    - kill -HUP 31874
    - kill -1 31874

シグナル名	シグナル番号	処理内容
HUP	1	プロセスのリスタート
KILL	9	プロセスの終了（強制終了）
QUIT	3	プロセスの終了（コアダンプ）
TERM	15	プロセスの終了（通常終了）
STOP	19	プロセスの一時停止
CONT	18	プロセスの一時停止からの復帰
CHLD	17	子プロセスの終了通知

```
# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  19400  1468 ?        Ss    2011   0:44 /sbin/init
root         2  0.0  0.0      0     0 ?        S     2011   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S     2011   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S     2011   0:01 [ksoftirqd/0]
...
root     31862  0.1  0.0  79580  4728 ?        S    20:47   0:00 /usr/sbin/packa
root     31872 30.8  1.8 470496 144464 ?       D    20:48   0:02 /usr/bin/python
enakai   31874  1.0  0.0 110292  1148 pts/2    R+   20:48   0:00 ps aux
```

プロセスの状態遷移 (1)

- プロセスの動作異常が発生した場合は、psコマンドなどでプロセスの状態を確認する必要があります。
  - 特に、「ps aux」の「STAT」列に表示される 1 文字目(R, D, S, Z)は、該当プロセスが下図の各状態にあることを表します。



プロセスの状態遷移 (2)

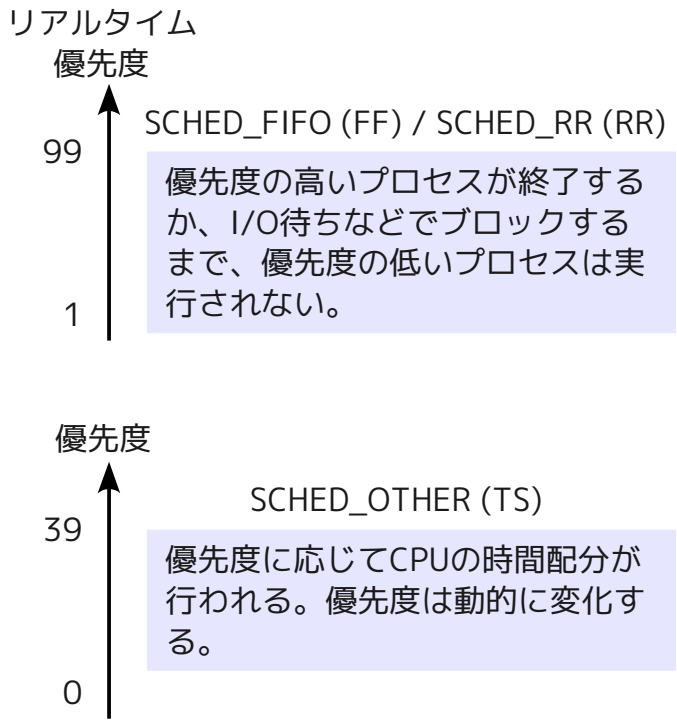
- 前ページの各状態 (STAT) の意味は次の通りです。
  - R (Running / Runnable): プロセスは、CPU を使用中です。
    - 厳密には、複数のプロセスが順番にCPUを使用するために、CPUの割り当て待ちの瞬間 (Runnable) も存在しますが、psコマンドでは、まとめて「R」状態として表示されます。
  - D (Uninterruptible sleep): プロセスは、ディスクI/Oの処理が完了するのを待っています。
    - この状態のプロセスが多数存在する場合、システムのディスクI/O負荷が高い可能性があります。
    - この状態のプロセスは、シグナルによる強制終了ができません。デバイスドライバの障害で、I/O処理が完了しない(エラーにもならない)ような場合に問題となることがあります。
  - S (Interruptible sleep): プロセスは、プログラムの指示により自発的にスリープしています。
    - 「一定時間後に復帰」「セマフォの解除を受けて復帰」など復帰条件を指定してスリープ状態に移行します。
  - T (Stopped): プロセスは、STOP シグナルを受けて、一時停止状態になっています。
    - CONT シグナルを送ると、プロセスは実行状態に戻ります。
    - 仮想ターミナルのフォアグラウンドで実行中のプロセスに対しては、Ctrl+zでも一時停止が可能です。実行状態に戻す場合は、fgコマンド(もしくはbgコマンド)を使用します。
  - Z (Defunct/Zombie): プロセスは実行を終了して、親プロセスの完了確認処理を待っています。
    - 実行が完了したプロセスは、親プロセスにCHLDシグナルを送信して、Defunct/Zombieと呼ばれる状態で停止します。CHLDシグナルを受けた親プロセスが完了確認処理を実施すると、このプロセスは消滅します。
    - 親プロセスがCHLDシグナルを受信できない場合、Defunct/Zombie状態のプロセスは、そのまま残ります。Defunct/Zombie状態のプロセスが発見された場合、親プロセスに何らかの異常がある可能性があります。

プロセスのスケジュールポリシー

- Linuxカーネルは内部的に複数のスケジュールポリシーを使用します。それぞれのプロセスは、どれか1つのポリシーに割り当てられます。
  - リアルタイムポリシーに属する「SCHED_FIFO (FF)」および「SCHED_RR (RR)」とノーマルポリシーに属する「SCHED_OTHER (TS)」が主に使用されます。

- リアルタイムポリシーは、すべての処理に優先してプロセスの実行を行います。
  - 主に、短時間で処理が完了するカーネルの内部プロセス（カーネルスレッド）が使用します。
  - 一般のユーザプロセスを割り当てることも可能ですが、CPUを使い続けるプロセスがあると、カーネル内部のI/O処理なども一切実行されず、システムがハングしたようになる恐れがあります。

- ノーマルポリシーは、リアルタイムポリシーで実行するべきプロセスが無い時にはじめて、プロセスの実行が行われます。
  - 一般のユーザプロセスはこのポリシーを使用します。



プロセスの優先度とNiceレベル (1)

- 各プロセスは内部的に優先度の値が割り当てられます。
  - リアルタイムポリシーに属するプロセスは、1～99の「リアルタイム優先度」を持ちます。
    - リアルタイム優先度の高い（値が大きい）プロセスから順に実行されます。リアルタイム優先度が最も高いプロセスが、常にCPUを使用します。リアルタイム優先度は、自動的に変化することはありません。
    - 各プロセスのスケジュールポリシーとリアルタイム優先度の値は、chrtコマンドで設定します。
  - SCHED_FIFO (FF)とSCHED_RR (RR)の違いは次のとおりです。
    - SCHED_FIFO (FF)は、同じリアルタイム優先度のプロセスを起動順に実行します。
    - SCHED_RR (RR)は、同じリアルタイム優先度のプロセスを一定時間ごとにラウンドロビンで実行します。
  - ノーマルポリシーに属するプロセスは、0～39の「優先度」を持ちます。
    - 優先度の値（値が大きと優先度が高い）に応じてCPU時間の配分が行われます。優先度はプロセスの実行状態に応じて動的に変化します。
    - RHEL6では、CFS（Complete Fair Scheduler）によって、インタラクティブプロセスのレイテンシ（反応時間）と非インタラクティブプロセスのスループット（総処理量）のバランスをとるように調整されます。
- ノーマルポリシーの優先度の変化は、プロセスの「Niceレベル」で間接的に制御することができます。
  - Niceレベルは、-20～19の値で指定します。Niceレベルが小さいほど優先度が大きくなる傾向が強くなります。

プロセスの優先度とNiceレベル (2)

- リアルタイムポリシーでのプロセス起動には、chrtコマンドを使用します。
  - chrt [-f][-r] <リアルタイム優先度> <コマンド>　：リアルタイムポリシーでプロセスを起動
    - -f: SCHED_FIFO (FF), -r: SCHED_RR (RR)
  - chrt -p <リアルタイム優先度> <プロセスID>　：実行中プロセスのリアルタイム優先度を変更

- ノーマルポリシーでのNiceレベル指定には、nice/reniceコマンドを使用します。
  - nice -n <Niceレベル> <コマンド>　　：Niceレベルを指定してコマンドを実行
  - renice <Niceレベル> <PID>　　：実行中プロセスのNiceレベルを変更

- 実行中プロセスのスケジュールポリシー、優先度などは下図のコマンドで表示します。
  - 次の項目を確認します。
    - CLS: スケジュールポリシー
    - RTPRIO: リアルタイム優先度
    - NI: Niceレベル
    - PRI: 優先度（リアルタイムポリシーの場合は「リアルタイム優先度 + 40」を表示）

```
# ps -eo user,pid,ppid,class,rtprio,ni,pri,pcpu,stat,comm
USER       PID  PPID CLS RTPRIO  NI PRI %CPU STAT COMMAND
root         1     0 TS       -   0  19  0.0 Ss   init
root         2     0 TS       -   0  19  0.0 S    kthreadd
root         3     2 FF      99   - 139  0.0 S    migration/0
root         4     2 TS       -   0  19  0.0 S    ksoftirqd/0
...
enakai    3177  2546 TS       -   0  19  0.0 S    gnome-panel
enakai    3206  2546 TS       -   0  19  0.0 S    nautilus
gdm       3207     1 TS       - -11  30  0.0 S<sl pulseaudio
enakai    3208     1 TS       - -11  30  0.0 S<sl pulseaudio
rtkit     3210     1 TS       -   1  18  0.0 SNl  rtkit-daemon
...
```

Upstartによる起動処理 (1)

- Linuxカーネルが最初に起動するプロセス「/sbin/init」は、ユーザレベルのサーバ設定やサービス起動などの処理を行います。

- RHEL5の「/sbin/init」は、設定ファイル「/etc/inittab」にしたがって次の処理を行います。
  - 初期設定スクリプト「/etc/rc.d/rc.sysinit」を実行して、ファイルシステムのfsckチェックとマウント処理を実施します。
  - スクリプト「/etc/rc.d/rc」を実行して、起動時のランレベルに応じたサービスを起動します。
  - コンソールログインの受付プロセス(mingetty)を起動します。
  - ランレベル5の場合は、GUIのログイン画面を起動します。

- RHEL6の「/sbin/init」はUpstartと呼ばれるイベントベースのジョブ管理システムに変更されています。
  - 「/etc/inittab」はデフォルトのランレベル設定のみが記載されます。

RHEL5の/etc/inittabの例

```
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
（中略）

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Upstartによる起動処理 (2)

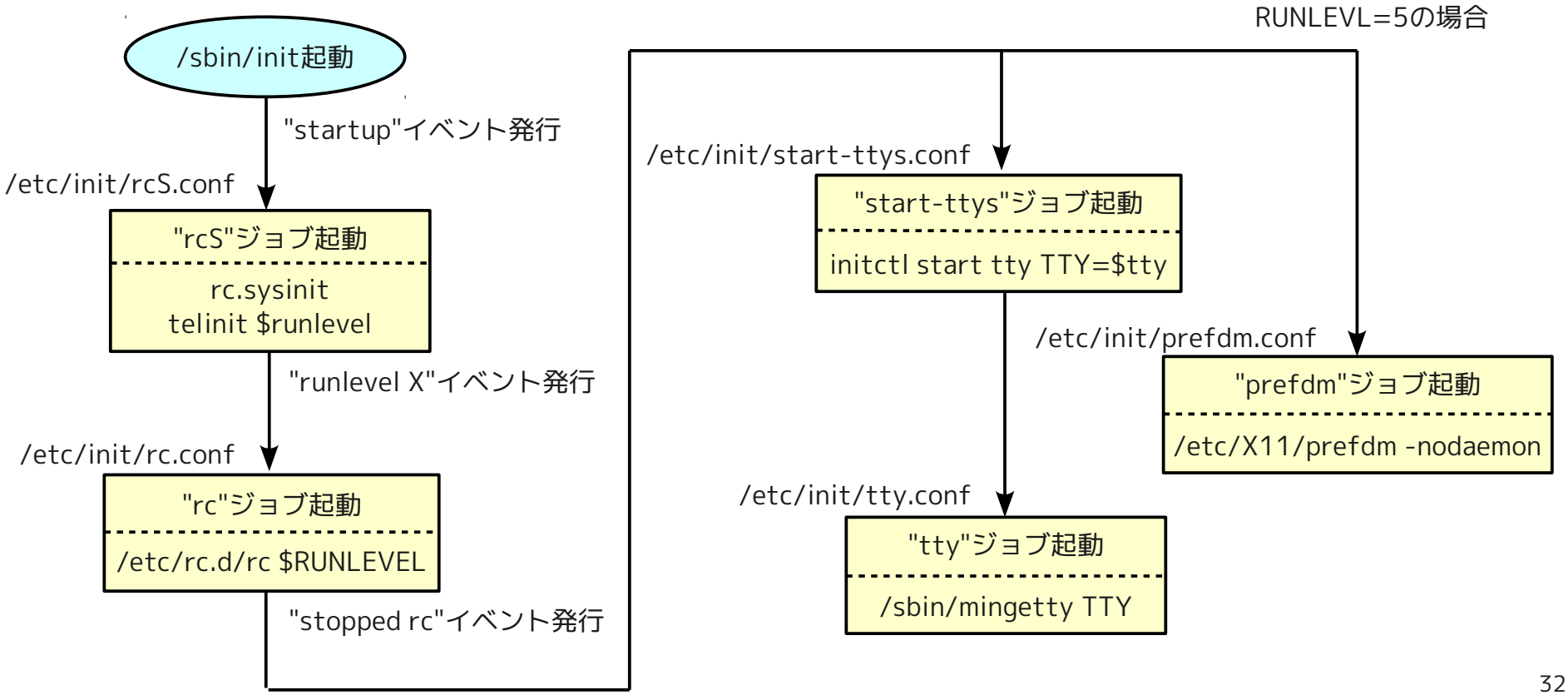
- Upstartでは、「/etc/init/<Job>.conf」に個々のジョブの定義を記載します。
  - 基本的には、1つのジョブは1つのプロセスに対応します。
    - インスタンス指定により、同じジョブに属するプロセスを複数起動することも可能です。
  - 次のような内容を定義します。
    - ジョブを起動するイベント、停止するイベント
    - ジョブの起動コマンドなどの指定
    - ジョブに対応するプロセスが終了した際に再起動するかどうか(task/respawn)
  - initctlコマンドでイベントの発生、ジョブ管理などが可能です。
    - # initctl emit <イベント>
    - # initctl start <ジョブ>
    - # initctl stop <ジョブ>
    - # initctl restart <ジョブ>
    - # initctl reload <ジョブ>
    - # initctl status <ジョブ>
    - # initctl list

Upstartの主なイベント

イベント	説明
startup	/sbin/initの起動時に発生
starting <Job>	<Job>の起動処理の開始時に発生
started <Job>	<Job>の起動処理の完了時に発生
stopping <Job>	<Job>の停止処理の開始時に発生
stopped <Job>	<Job>の停止処理の完了時に発生
runlevel X	telinitコマンドでランレベル変更時に発生

Upstartによる起動処理 (3)

- RHEL6では、RHEL5までの/etc/inittabと同等のプロセス起動をUpstartのジョブとして定義しています。
  - 下図は代表的なジョブの起動順序を示します。とくに「“rc”ジョブ」において、chkconfigコマンドで設定された/etc/init.d/以下のサービスを起動します。



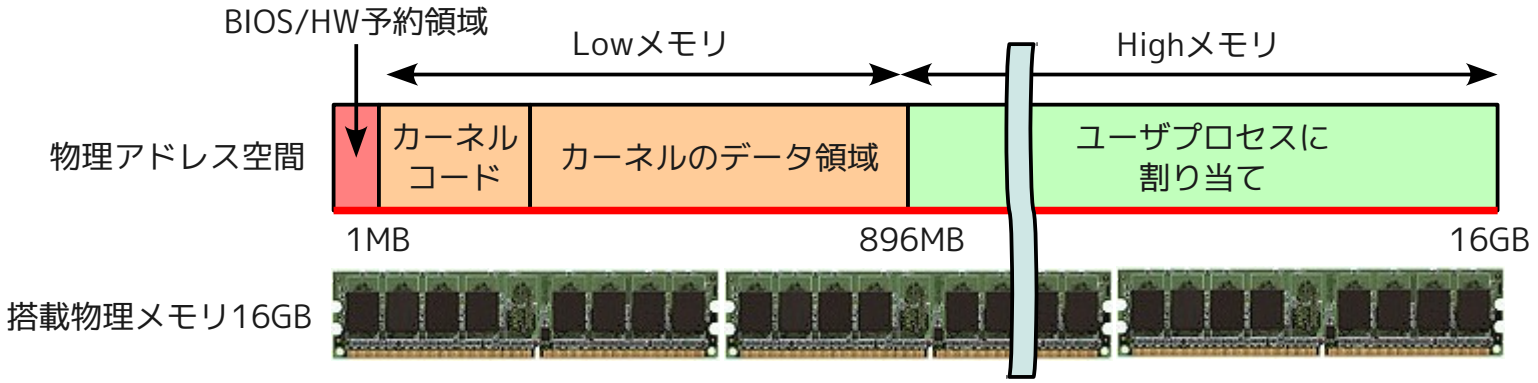
メモとしてお使いください

メモとしてお使いください

Linuxのメモリ管理

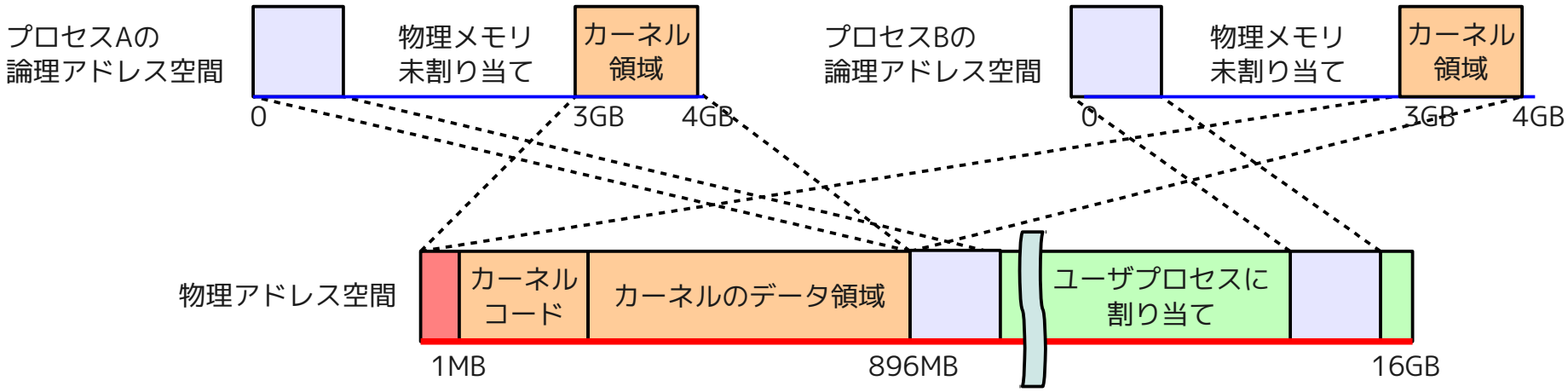
物理アドレス空間

- 「物理アドレス空間」と「論理アドレス空間」を区別して理解する事が大切です。物理アドレス空間は、サーバに搭載された物理メモリの領域を指定するアドレスです。
  - 例えば、16GB のメモリを搭載したサーバでは、物理アドレス空間は「0~16GB」になります。
  -
- x86アーキテクチャ（32bitカーネル）ではLinuxカーネルが使用できる物理メモリに上限があります。
  - Linuxカーネルは1MB~896MBの物理アドレス空間(Lowメモリ)のみを使用します。896MB以降は、ユーザプロセスが使用するメモリ、およびディスクキャッシュとしてのみ利用されます。
    - Lowメモリでカーネルが使用していない部分は、ユーザプロセスも使用可能です。
  - x86_64アーキテクチャ（64Bitカーネル）ではこのような制限はありません。
    - 1MB以降のすべての物理メモリがカーネルにも使用できるLowメモリとして認識されます。



x86アーキテクチャの論理アドレス空間

- x86アーキテクチャでは、Linux上の各ユーザプロセスは、プロセスの起動時に論理アドレス空間と呼ばれる「0~4GB」の仮想的なアドレス空間を割り当てられます。
  - 各ユーザプロセスは、論理アドレスを指定してメモリへのアクセスを行います。あくまで仮想的なアドレスであり、各アドレスに対応する物理メモリが必ず存在するわけではありません。
- ユーザプロセスが、論理アドレスを指定してメモリへのアクセスを行うと、Linuxカーネルは、『必要に応じて』物理メモリを割り当てていきます。
  - 各プロセスの使用する論理アドレスと、対応する割り当て済みメモリの物理アドレスの変換は、Linuxカーネルが自動的に行います。



x86アーキテクチャのメモリ容量制限

- x86アーキテクチャでは、論理アドレス3GB～4GBの領域には、カーネルが存在するLowメモリがマッピングされます。
  - ユーザプロセスを実行中のCPUは、カーネルの処理が必要になるとカーネル領域に処理を飛ばします。
- x86アーキテクチャでは、使用できる論理アドレスが「0x0000 0000~0xFFFF FFFF」の4GB（32ビット）の範囲に限られます。
  - この限られた範囲をユーザメモリとカーネル領域のマッピングに使用する必要があり、ユーザメモリを「0～3GB」、カーネル領域を「3GB～4GB」にマッピングすると決められています。
- この結果、1つのプロセスが使用できるメモリは最大3GB、カーネルが使用できるメモリは最大1GB（正確には900MB弱）というメモリ容量の制限が発生します。
  - x86_64アーキテクチャでは、論理アドレスの範囲は64ビット（実質上無限）あるためこのような制限は発生しません。
    - 具体的には、論理アドレス「0x0000 0000 0000 0000～0x0000 7FFF FFFF FFFF」をユーザメモリが使用して、論理アドレス「0xFFFF FFFF 8000 0000～0xFFFF FFFF FFF0 0000」にカーネル領域がマッピングされます。

論理アドレス空間の確認

- 各プロセスの論理アドレスの使用状況は、pmapコマンドで表示することができます。
  - 右図は、プロセスID「8263」で実行中の「httpd」の論理アドレス空間を表示しています。
  - 一番右の列は、論理アドレス空間の各領域の使用方法を示します。
    - <ファイル名>：ディスク上のファイルを読み込む。
    - [anon]：プロセスのデータ領域として使用。
    - [stack]：プロセスのスタック領域として使用。
  - 左の2列が各領域の論理アドレスの範囲を示します。
- 一番最後の行は、vsyscall（仮想システムコール）機能に使用します。
  - カーネル領域のメモリの一部をユーザプロセスに公開して、ユーザプロセスから呼び出し可能にしています。

```
# pmap -x 8263
8263:   /usr/sbin/httpd
Address           Kbytes     RSS   Dirty Mode   Mapping
00007fb6f52d8000       8       0       0 r-x--  apr_ldap-1.so
00007fb6f52da000    2044       0       0 -----  apr_ldap-1.so
00007fb6f54d9000       4       4       4 rw---  apr_ldap-1.so
...
00007fb6f8673000      16      12       0 r-x--  mod_dnssd.so
00007fb6f8677000    2048       0       0 -----  mod_dnssd.so
00007fb6f8877000       4       4       4 rw---  mod_dnssd.so
00007fb6f8878000       8       0       0 r-x--  mod_version.so
00007fb6f887a000    2044       0       0 -----  mod_version.so
00007fb6f8a79000       4       4       4 r----  mod_version.so
00007fb6f8a7a000       4       4       4 rw---  mod_version.so
...
00007fb702d7d000      28      28      28 rw---    [ anon ]
00007fb702d95000       4       4       4 rw---    [ anon ]
00007fb702d96000       4       4       4 r----  ld-2.12.so
00007fb702d97000       4       4       4 rw---  ld-2.12.so
00007fb702d98000       4       4       4 rw---    [ anon ]
00007fb702d99000     332      56       0 r-x--  httpd
00007fb702fec000      16      16      16 rw---  httpd
00007fb702ff0000      12       8       8 rw---    [ anon ]
00007fb7045e8000    2132    2004    2004 rw---    [ anon ]
00007fb7047fd000     476     356     356 rw---    [ anon ]
00007ffff9cf1000      84      32      32 rw---    [ stack ]
00007ffff9dff000       4       4       0 r-x--    [ anon ]
ffffffffff600000       4       0       0 r-x--    [ anon ]
----------------  ------  ------  ------
total kB          226136    4716    3812
```

デマンドページングについて

- pmapコマンドの出力は、各領域の「使用方法」を示していますが、必ずしも各領域に物理メモリが割り当てられているわけではありません。これは、物理メモリを効率的に使用するための機能で、「デマンドページング」と呼ばれます。
  - ファイルの読み込み領域は、プログラムが実際にその論理アドレスにアクセスして、ファイルの内容を読み込もうとした時に、初めて物理メモリが割り当てられ、ディスク上のファイルの内容が物理メモリに転送されます。
  - [anon] の領域は、プログラムが malloc() 関数などでメモリの割り当てを要求した際に、論理アドレス空間上に領域が登録されます。その後、プログラムが実際にその領域にデータを書き込もうとした際に、初めて物理メモリが割り当てられます。

- プログラムがmalloc()関数でメモリ割り当てを要求する際に、実際に使用可能な物理メモリをこえて要求できます。これは「メモリのオーバコミット」と呼ばれます。
  - オーバコミットが許可されない場合は、malloc()関数はエラーを返します。

カーネルパラメータ	説明
vm.overcommit_memory （デフォルトは0）	0：カーネルは経験則に基づいて、オーバコミットの許可／不許可を決定します。 1：カーネルは常にオーバコミットを許可します。 2：カーネルはオーバコミットの上限を 「総スワップ領域 + 総物理メモリ x vm.overcommit_ratio(%)」に制限します。
vm.covercommit_ratio （デフォルトは50）	「vm.overcommit_memory = 2」の説明を参照。

ディスクキャッシュの使用

- Linuxは、物理メモリの空き容量の大部分をディスクキャッシュとして使用します。
  - ディスクキャッシュ上のデータは、ファイルアクセス完了後も解放されずに残るため、大容量のファイルアクセスを実施した後は、見かけ上、メモリの空き容量が大きく減少します。
  - プロセスが必要とするメモリが不足した場合、ディスクキャッシュは適宜解放されていきますので、実質的なメモリの空き容量は、ディスクキャッシュの容量を含めて考える必要があります。
    - freeコマンドのbuffers と cached の合計がディスクキャッシュとして使用されているメモリ容量になります。「-/+ buffers/cache:」の行の値は、ディスクキャッシュを空き容量と見なした値を示します。
  - 次のコマンドは、解放可能なディスクキャッシュを強制的に解放します。
    - # sync; echo 3 > /proc/sys/vm/drop_caches

- ディスクのスワップ領域は、メモリの空き容量が不足した際に、物理メモリの一部をスワップ領域に退避（スワップアウト）して必要な空き容量を確保する仕組みです。
  - スワップアウトした内容は、プロセスからのアクセスが発生すると、再度、物理メモリへの読み込み（スワップイン）が行われます。
  - 長期間アクセスされないデータをスワップ領域に移動することで、物理メモリを効率的に使用できますので、スワップアウトが発生する事自体は問題ではありません。
  - スワップアウトと同時にスワップインが頻繁に発生する場合は、必要なデータがスワップ領域に転送されており、スワップ処理によるシステムパフォーマンスの低下が予想されます。

tmpfsと共有メモリについて

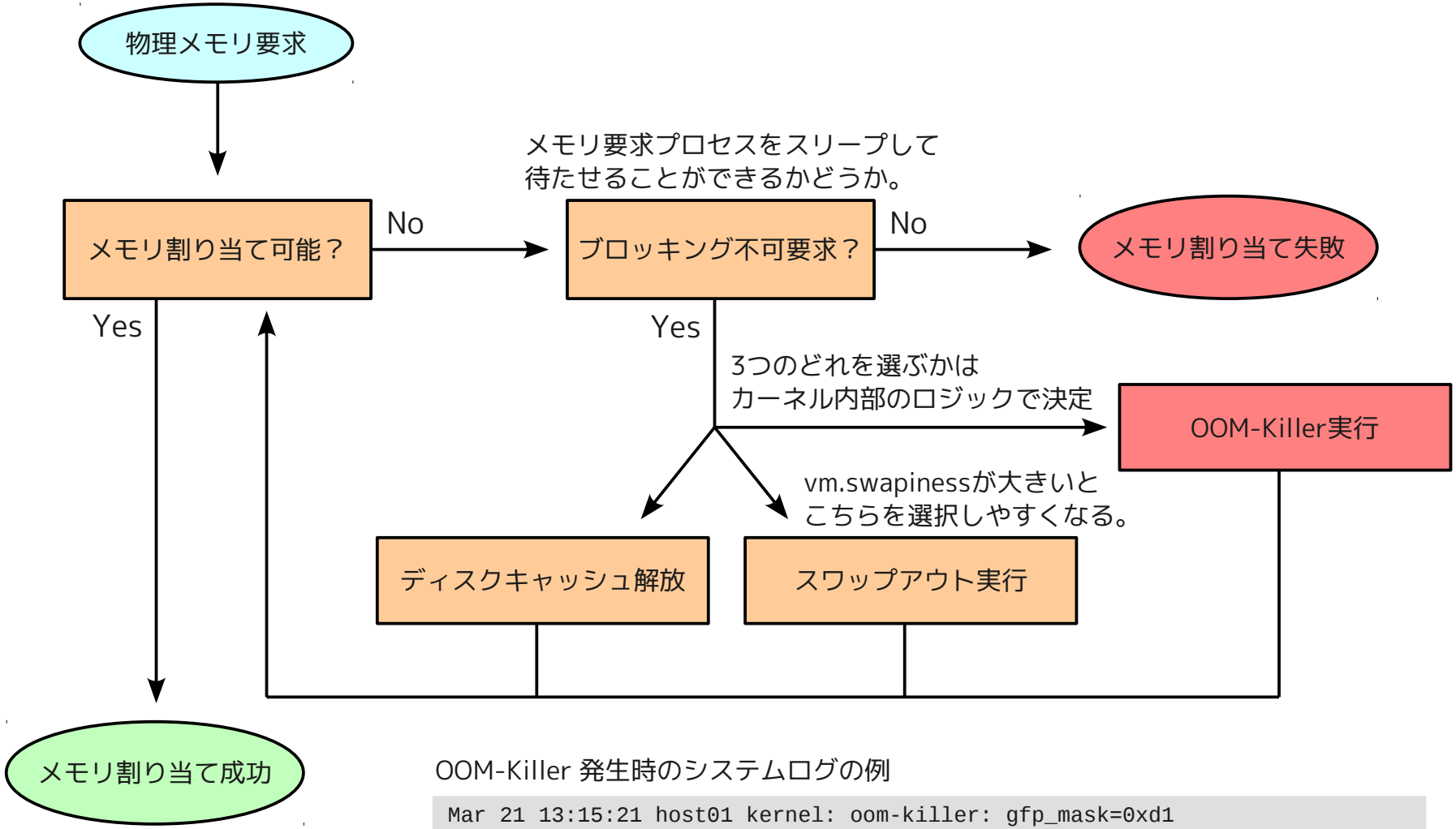
- tmpfsは、サーバのメモリを利用したラムディスク機能を提供します。
  - 次は、1024MBのラムディスクを「/data」にマウントする例です。
    - # mount -t tmpfs -o size=1024M tmpfs /data
  - tmpfsに保存したファイルは、ディスクキャッシュ領域に書きこまれます。tmpfsが使用中のディスクキャッシュ領域は解放されることはありませんが、スワップアウトの対象にはなります。

- Linuxでは、プロセス間共有メモリの領域として、内部的にtmpfsを使用します。
  - 共有メモリを使用するアプリケーションが稼動している場合、共有メモリの使用量は、ディスクキャッシュの使用メモリに含まれます。
    - POSIX共有メモリは、「/dev/shm」にマウントされたtmpfsを使用します。dfコマンドで使用量が確認できます。
    - SysV共有メモリは、カーネルが内部的にtmpfsを作成するので、dfコマンドでは表示されません。SysV共有メモリの使用状況は、ipcsコマンドで確認します。

ディスクキャッシュとスワップのチューニング

- ディスクキャッシュに関する主なカーネルパラメータは次のとおりです。
  - /proc/sys/vm/dirty_background_ratio, /proc/sys/vm/dirty_ratio
    - ディスクキャッシュの内容が変更された後に、それをディスクに書きだす割合を制御します。
    - ディスクに未反映のディスクキャッシュ(dirtyキャッシュ)の割合を「dirty_background_ratio(%)」 以下に保つように試みます。特にdirtyキャッシュの割合が「dirty_ratio(%)」を越えると、ディスクキャッシュの書き出し頻度を増加します。
  - /proc/sys/vm/dirty_writeback_centisecs, /proc/sys/vm/dirty_expire_centisecs
    - 長時間ディスクに反映されていないdirtyキャッシュをディスクに書き出す頻度を決定します。
    - 「dirty_writeback_centisecs(単位は1/100秒)」毎にディスクキャッシュをチェックして、「dirty_expire_centisecs(単位は1/100秒)」以上の間、書き出されていなかった内容を書き出します。
- スワップの使用に関する主なカーネルパラメータは次のとおりです。
  - /proc/sys/vm/min_free_kbytes
    - 物理メモリの空き容量がこの値(単位は KB)以下になると、ディスクキャッシュの解放、あるいはスワップアウトの処理を開始します。
  - /proc/sys/vm/swappiness
    - 空きメモリが不足した際に、ディスクキャッシュの解放よりもスワップアウトを選択する傾向を0~100の値で指定します。値が大きいほどスワップアウトが頻繁に発生します。デフォルト値は60です。

物理メモリの割り当てロジック



OOM-Killer 発生時のシステムログの例

```
Mar 21 13:15:21 host01 kernel: oom-killer: gfp_mask=0xd1
~メモリ使用状況の詳細を記録~
Mar 21 13:15:21 host01 kernel: Out of Memory: Killed process 31065 (java).
```

参考資料: 問題判別の基本コマンド

manページについて

- Linuxの多くのコマンドには、詳細なオンラインマニュアルが用意されています。
  - これらは、manコマンドで参照するため「manページ」と呼ばれます。ここで紹介する各コマンドについても、指定可能オプションなどの詳細は、manページを参照してください。

- manコマンドの書式は次のとおりです: man <セクション番号> <コマンド>
  - <セクション番号> には次のようなものがあります。省略時は若い番号が選択されます。

セクション番号	説明
0p	ヘッダファイル
1 / 1p	一般のユーザコマンド
2	システムコール(カーネルが提供する関数)
3 / 3p	ライブラリ関数
4	デバイス(/dev ディレクトリのスペシャルファイル)
5	ファイルフォーマットの説明(/etc/passwdなど)
7	その他(マクロパッケージや標準規約)
8	システム管理ツール(root ユーザー専用)
9	Linux 独自のカーネルルーチン

  - あるコマンドのマニュアルを含むセクションは、whatisコマンドで確認します。

```
# whatis vmstat
vmstat               (8)  - Report virtual memory statistics
```

基本情報の確認 (1)

- システムの動作に異常があった場合、まずはファイルシステムの空き容量不足、メモリの不足、CPUを占有しているプロセスの存在など基本的な問題の有無を確認します。(*)
  - uname -a ⇒ カーネルのバージョン、カーネルタイプ (32bit/64bit) などを確認します。
    - 図は 64bitカーネルの例です。32bitカーネルでは、「x86_64」の部分が「i686/i386」になります。
  - df ⇒ マウント中のファイルシステムと使用量を確認します。
  - free ⇒ メモリとスワップ領域の使用量を確認します。
    - Linuxは空きメモリをディスクキャッシュとして使用します。ディスクキャッシュ部分は必要に応じて解放されるので、実質的な空きメモリは、ディスクキャッシュ部分を空きメモリと見なした値で考えます。これは、「-/+ buffers/cache:」 の行のfreeの値になります。

```
# uname -a
Linux server01 2.6.32-220.2.1.el6.x86_64 #1 SMP Tue Dec 13 16:21:34 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
```

```
# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda2            403173232 326577724  56115508  86% /
tmpfs                  3968536       352   3968184   1% /dev/shm
/dev/sda1               495844     82406    387838  18% /boot
/dev/mapper/backup_vg01-backup_lv01
                     206424760 126958948  68980052  65% /backup
```

```
# free
             total       used       free     shared    buffers     cached
Mem:       7937076    4572572    3364504          0     383008    2137408
-/+ buffers/cache:    2052156    5884920
Swap:      2097144          0    2097144
```

(*) 問題判別の情報収集をする際は、次のコマンドで、出力内容を英語に設定することをお勧めします。

```
# export LANG=C
```

基本情報の確認 (2)

- ps -ef ⇒ 稼動中のプロセスを確認します。
  - デフォルトでは、画面の右端を越える部分は表示が省略されますが、「ww」オプションを追加すると、行を折り返して、すべての内容が表示されます。
- top ⇒ CPUを使用中のプロセスを確認します。(qで終了)

```
# ps -efww
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0  2011 ?        00:00:43 /sbin/init
root         2     0  0  2011 ?        00:00:00 [kthreadd]
root         3     2  0  2011 ?        00:00:00 [migration/0]
root         4     2  0  2011 ?        00:00:01 [ksoftirqd/0]
...
root      1293     1  0  2011 ?        00:00:00 /sbin/portreserve
root      1300     1  0  2011 ?        00:00:14 /sbin/rsyslogd -i /var/run/syslogd.pid -c 4
...
```

```
# top

top - 18:07:21 up 14 days,  6:12,  3 users,  load average: 0.58, 0.46, 0.38
Tasks: 281 total,   1 running, 280 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.0%us,  1.0%sy,  0.0%ni, 96.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   7937076k total,  4561456k used,  3375620k free,   383020k buffers
Swap:  2097144k total,        0k used,  2097144k free,  2137600k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11077 root      20   0  708m  71m  13m S  9.0  0.9  12:42.58 python
 2270 root      20   0  998m  32m 4776 S  4.0  0.4  74:50.71 libvirtd
26395 qemu      20   0 2331m 805m 3688 S  3.0 10.4 162:23.67 qemu-kvm
 7399 root      20   0 15220 1376  952 R  0.3  0.0   0:00.06 top
11017 enakai    20   0 99108 3228  996 S  0.3  0.0   0:15.34 sshd
    1 root      20   0 19400 1468 1156 S  0.0  0.0   0:43.92 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.12 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.37 migration/0
```

基本情報の確認 (3)

- ifconfig -a ⇒ ネットワークインターフェースの構成を確認します。
- netstat -a ⇒ 他のサーバ、クライアントとのTCP/UDP接続の状態を確認します。
  - 「n」オプションを指定すると、ホスト名、ポート名が数字で表記されます。
- ethtool ethX ⇒ ネットワークインターフェース「ethX」の通信モード(通信速度、全二重/半二重など)を確認します。
- cat /proc/net/bonding/bondX ⇒ Bondingデバイス「bondX」の稼動状況を確認します。

```
# ethtool eth0
Settings for eth0:
      Supported ports: [ TP ]
      Supported link modes:   10baseT/Half 10baseT/Full
                              100baseT/Half 100baseT/Full
                              1000baseT/Full
      Supports auto-negotiation: Yes
      Advertised link modes:  10baseT/Half 10baseT/Full
                              100baseT/Half 100baseT/Full
                              1000baseT/Full
      Advertised pause frame use: No
      Advertised auto-negotiation: Yes
      Speed: 100Mb/s
      Duplex: Full
      Port: Twisted Pair
      PHYAD: 2
      Transceiver: internal
      Auto-negotiation: on
      MDI-X: off
      Supports Wake-on: pumbg
      Wake-on: g
      Current message level: 0x00000001 (1)
      Link detected: yes
```

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.0.3 (March 23, 2006)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth0
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:5b:82:c5

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:5b:82:cf
```

基本情報の確認 (4)

- uptime ⇒ サーバの連続稼動時間などを表示します。
- last ⇒ サーバにログインしたユーザーの履歴を表示します。リブートの記録も確認できます。
- w ⇒ サーバにログイン中のユーザーを表示します。

```
# uptime
 18:15:11 up 14 days,  6:20,  3 users,  load average: 0.08, 0.14, 0.24
```

```
# last | head -10
enakai   pts/3        vpn-247-8.xxx.re Mon Jan  9 15:48   still logged in
enakai   pts/2        vpn-247-8.xxx.re Mon Jan  9 15:42   still logged in
enakai   pts/3        vpn-247-7.yyy.re Mon Jan  9 11:59 - 12:03  (00:04)
enakai   pts/2        vpn-247-8.yyy.re Mon Jan  9 10:52 - 13:07  (02:14)
enakai   pts/5        vpn-247-2.zzz.re Sun Jan  8 21:15 - 22:03  (00:47)
enakai   pts/5        dhcp-193-143.zzz Fri Jan  6 13:25 - 16:39  (03:13)
...
```

```
# w
 18:17:06 up 14 days,  6:22,  3 users,  load average: 0.01, 0.09, 0.21
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
enakai   pts/0    :99.0            Wed12    5days  0.08s  0.68s /usr/bin/gnome-
enakai   pts/2    vpn-247-8.xxx.re 15:42    0.00s  0.16s  0.05s sshd: enakai [p
enakai   pts/3    vpn-247-8.yyy.re 15:48    2:28m  0.06s  0.06s sshd: enakai [p
```

接続デバイスの確認

- LinuxがOSレベルで認識しているデバイスの情報は、次のコマンドで確認できます。
  - lspci ⇒ PCI 接続デバイスの情報を表示します。
    - 「-v」オプションでより詳細な情報が表示されます。
  - lsusb ⇒ usb 接続デバイスの情報を表示します。
    - 「-v」オプションでより詳細な情報が表示されます。
  - cat /proc/scsi/scsi ⇒ SCSI 接続デバイスの情報を表示します。
  - cat /proc/interrupts ⇒ 各デバイスの CPU 割り込み回数を表示します。
  - lsmod ⇒ デバイスドライバなど、ロードされているカーネルモジュールを表示します

```
# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: ST3500413AS      Rev: JC47
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: Hitachi HDP72505 Rev: GM4O
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi3 Channel: 00 Id: 00 Lun: 00
  Vendor: TSSTcorp Model: DVD-ROM TS-H353C Rev: D100
  Type:   CD-ROM                           ANSI  SCSI revision: 05
```

導入パッケージの確認

- サーバに導入されているパッケージの情報は、次のコマンドで確認できます。
  - rpm -qa ⇒ サーバに導入されているRPMパッケージを表示します。
    - 64bitカーネル環境では、次のコマンドでアーキテクチャ (32bit用/64bit用) 情報を含めて表示できます。

```
# rpm -qa --qf '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n'
psutils-1.17-34.el6.x86_64
m17n-contrib-assamese-1.1.10-4.el6_1.1.noarch
libbonobo-2.24.2-4.el6.x86_64
xorg-x11-drv-evdev-2.6.0-2.el6.x86_64
xsane-common-0.997-8.el6.x86_64
paps-libs-0.6.8-13.el6.2.x86_64
lynx-2.8.6-27.el6.x86_64
・・・
```

  - rpm -ql <rpmパッケージ名> ⇒ 導入済みのRPMパッケージに含まれるファイルを表示します。
  - rpm -qlp <rpmパッケージファイル>

    ⇒ RPMパッケージファイルに含まれるファイルを表示します。
  - rpm -qf <ファイルパス> ⇒ ファイルを含む RPM パッケージを表示します。
  - rpm -qi <rpmパッケージ名> / rpm -qip <rpmパッケージファイル>

    ⇒ RPMパッケージの説明文を表示します。
  - rpm -q --changelog <rpmパッケージ名> / rpm -q -changelog -p <rpmパッケージファイル>

    ⇒ RPMパッケージに含まれる変更履歴を表示します。

プロセスに関する情報

- サーバ稼動中のプロセスに関する情報は、次のコマンドで確認できます。
  - ps -ef ⇒ 稼動中のプロセスを確認します。
  - top ⇒ CPUを使用中のプロセスを確認します。(qで終了)
  - lsof ⇒ 各プロセスがオープン中のファイルを表示します。
  - lsof -i ⇒ 各プロセスがオープン中のTCP/UDPポートを表示します。
  - fuser -v <ファイルパス> ⇒ ファイルをオープン中のプロセスを表示します。
  - fuser -vm <マウントポイント>

    ⇒ マウント中のファイルシステム内のファイルをオープン中のプロセスを表示します。

```
# fuser -v /usr/sbin/sshd
                     USER        PID ACCESS COMMAND
/usr/sbin/sshd:      root       1640 ...e. sshd
                     root      26643 ...e. sshd
                     enakai    26647 ...e. sshd
```

```
]# lsof -i
COMMAND     PID    USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
portreser  1293    root    5u  IPv4   12452      0t0  UDP *:ipp
rpcbind    1353     rpc    6u  IPv4   12713      0t0  UDP *:sunrpc
rpcbind    1353     rpc    7u  IPv4   12717      0t0  UDP *:entrust-aaas
rpcbind    1353     rpc    8u  IPv4   12718      0t0  TCP *:sunrpc (LISTEN)
rpcbind    1353     rpc    9u  IPv6   12720      0t0  UDP *:sunrpc
rpcbind    1353     rpc   10u  IPv6   12722      0t0  UDP *:entrust-aaas
rpcbind    1353     rpc   11u  IPv6   12723      0t0  TCP *:sunrpc (LISTEN)
...
```

リソース使用状況の確認

- サーバリソースの使用状況は、次のコマンドで確認できます。
  - vmstat <計測間隔(秒)> <回数>

    ⇒ 実行待ちプロセス数、空きメモリ、I/O、平均CPU使用率などを総合的に表示します。
    - 最初の出力値のみ、システム起動後からの平均値を表します。
  - mpstat -P ALL <計測間隔(秒)> <回数> ⇒ CPU 別の使用率を表示します。
  - iostat -xd <計測間隔(秒)> <回数> ⇒ 各ディスクデバイスの I/O 量を表示します。
    - 最初の出力値のみ、システム起動後からの平均値を表します。
  - swapon -s ⇒ スワップ領域の使用量を表示します。
  - netstat -i ⇒ ネットワークデバイスのパケット転送量を表示します。

```
# vmstat 1 5
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0      0 3444736 383024 2140504    0    0     8    16    5    0  1  1 98  0  0
 1  0      0 3438892 388964 2140532    0    0  5940    20 2773 5193  1  2 84 13  0
 1  0      0 3428236 398848 2140504    0    0  9884     0 3841 11697  4 10 74 12  0
 1  0      0 3421788 403616 2140504    0    0  4768     4 3952 8931  4  9 86  1  0
 0  0      0 3415464 408180 2140508    0    0  4564     0 3735 13685  3  6 85  6  0
```

```
# netstat -i
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
em1        1500   0 41015768      0      0      0 24547511      0      0      0 BMRU
lo        16436   0  2493067      0      0      0  2493067      0      0      0 LRU
virbr0     1500   0     7839      0      0      0    10850      0      0      0 BMRU
```

カーネルパラメータの確認

- Linuxでは、/proc以下のファイルを通して、プロセスやカーネルの動作に関するパラメータを変更することができます。
  - /proc 以下のファイルは、ディスクに存在するファイルとは無関係で、これらの内容をcatコマンドで表示すると現在の設定値が表示されて、echoコマンドで値を書き込むと設定値が即座に変更されます。
  - /proc/sys直下のディレクトリに、各種のカーネルパラメータがまとめられています。
    - /proc/sys/fs：ファイルシステムに関するパラメータが存在します。
    - /proc/sys/kernel：カーネルの動作に関するパラメータが存在します。
    - /proc/sys/net：ネットワークに関連したパラメータが存在します。
    - /proc/sys/vm：メモリー管理に関連したパラメータが存在します。

- /proc/sys 以下のパラメーターはシステム起動時に、設定ファイル「/etc/sysctl.conf」 の内容で初期化されます。
  - /etc/sysctl.conf の内容を変更後、変更を即座に反映する場合は、次のコマンドを使用します。
    - # sysctl -p
  - 全項目の現在の設定値は次のコマンドで表示されます。
    - # sysctl -a

メモとしてお使いください

メモとしてお使いください

メモとしてお使いください