Asianux

Asianux Server 4 サーバー構築・運用ガイド

Copyright/Trademarks

目次

第 1 章　ディスク管理 3
1.1　ディスク管理の概要 4
1.1.1　デバイスファイル 4
1.2　パーティション 7
1.2.1　パーティション分割のメリット 9
1.2.2　パーティション分割候補のディレクトリーと分割例 11
1.2.3　パーティションの作成 15
1.2.4　fdisk によるパーティション操作 15
1.2.5　parted によるパーティション操作 19
1.3　ファイルシステム 22
1.3.1　ext4 ファイルシステム 22
1.4　RAW デバイス 27
1.4.1　RAW デバイスの利用 27
1.4.2　RAW デバイスの起動設定 28
1.5　ソフトウェア RAID 29
1.5.1　ソフトウェア RAID の作成 30
1.5.2　RAID の運用 34
1.5.3　RAID 障害 34
1.6　LVM (Logical Volume Manager) 36
1.6.1　物理ボリュームの作成 37
1.6.2　ボリュームグループの作成 38
1.6.3　論理ボリュームの作成 39
1.6.4　論理ボリュームの利用 40
1.6.5　スナップショットの取得 41
1.6.6　ディスクの追加 42
1.6.7　ディスクの交換 44
1.6.8　ディスクの削除 46
1.7　quota の設定 47
1.7.1　quota とは 47
1.7.2　quota の設定方法 47

第 2 章　ネットワーク設定.....51
2.1　ネットワーク設定の概要.....52
2.2　network サービスと NetworkManager サービス.....52
2.3　ネットワークの起動と停止.....54
2.4　ネットワークの設定.....57
2.4.1　設定方法.....57
2.4.2　設定ファイル.....63
2.5　ネットワークの状況の確認.....66
2.5.1　ifconfig.....66
2.5.2　netstat.....66
2.5.3　ping.....67
2.6　ボンディングインターフェイスの設定.....69
2.6.1　設定ファイル.....69
2.6.2　設定確認.....72
2.7　ジャンボフレームの設定.....74

第1章　ディスク管理

この章で説明する内容

目的	ハードディスクの領域の管理方法について理解する
機能	ディスクパーティションの操作 EXT4 ファイルシステムの利用 RAW デバイスの利用 ソフトウェア RAID の利用 LVM の利用 quota の設定
必要な RPM	coreutils util-linux mount e2fsprogs mdadm lvm quota
設定ファイル	/etc/fstab　/etc/mdadm.conf　/etc/lvmtab
章の流れ	1 ディスク管理の概要 2 パーティションの操作 3 ファイルシステム(ext4) 4 RAW デバイス 5 ソフトウェア RAID 6 LVM について 7 quota の設定
関連 URL	`http://linuxjf.sourceforge.jp/JFdocs/INDEX-diskmanage.html`

1.1　ディスク管理の概要

ハードディスクやSAN (Storage Area Network) などのストレージの領域管理は、Linux サーバー管理者の最も重要な仕事の 1 つです。ここでは最も基本的なディスクの管理手順に関して説明します。Linux サーバーに新しくディスクを増設して実際に使用するためには、一般的には次のような手順で行います。

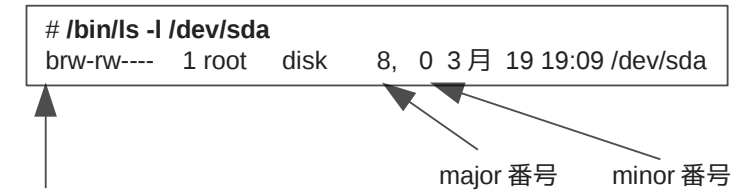
1) ハードディスクの物理的な接続
2) パーティションの作成——fdisk, parted
3) ファイルシステムの作成——mkfs
4) マウント——mount

まず物理的にハードディスクをサーバーに接続します。この作業は、各ハードウェアの取り扱い説明書に従って行ってください。

1.1.1　デバイスファイル

Linux では、ハードディスクや CD-ROM、フロッピーディスクなどのデバイスはデバイスファイルを通じて扱われます。デバイスファイルはデバイスを抽象化してファイルとして表現したものです。通常のファイルはデータを格納するために利用されますが、デバイスファイルは各種デバイスにアクセスするために利用されます。標準的なデバイスファイルは、OS のインストール時に **/dev** ディレクトリー配下に作成されます。**/dev** ディレクトリー配下には、ディスクデバイス以外のデバイス用のデバイスファイルも作成されています。

デバイスファイルは、major 番号と minor 番号を持っており、OS はこの番号を使ってアクセス対象のデバイスを特定します。major/minor 番号はデバイスドライバーによってデバイスごとに決められています。デバイスファイルに関連付けられた major/minor 番号は、**ls** コマンドを使って確認できます。また、デバイスファイルはデバイスの種類によって 2 種類に分かれていて、ディスクのようにブロック単位でアクセスして、ランダムアクセス可能な**ブロックデバイス**と、端末のようにキャラクター単位でアクセスする**キャラクターデバイス**があります。



一般的なディスク装置のデバイスファイルとして、次のものがよく利用されます。

- SCSI デバイス
  - **/dev/sda**、**/dev/sdb**、**/dev/sdc** など

    SCSI コントローラーや、SCSI RAID コントローラーに接続されたSCSI ディスクデバイスを表します。また、Fibre Channel に接続されたストレージ装置のディスクや、USB 接続のディスク装置や、SATA ディスクデバイスなどもこの形式で表されます。1 つのディスクが **/dev/sda** といった形式で表され、そのディスク内のパーティションはパーティション番号に従って **/dev/sda1**、**/dev/sda2** といった形式で表されます。SCSI デバイスのデバイスファイル名はシステム起動時にSCSI デバイスを探索し、ディスクを発見した順番で決まります。SCSI デバイスの探索は、システム起動時にロードされるSCSI デバイス用のドライバーがロードされる順番に基づいて行われます。同一のSCSI チャンネルに接続されたSCSI ディスクの場合、SCSI ID の小さなものから探索が行われます。そして、最初に発見したディスク装置が **/dev/sda**、次に発見したディスク装置が **/dev/sdb** というように割り当てられます。したがって新規に SCSI コントローラーや SCSI デバイスを追加した場合、デバイスファイルの割り当て順が変わる可能性があります。
  - **/dev/scd0**

    SCSI 接続のドライブを利用する場合には /dev/scd0 を使用します。現在一般的に使われている形式のCD ドライブは、sr0 などで表され、scd0 は sr0 へハードリンクされています。

- IDE デバイス
  - **/dev/sda**、**/dev/sdb**、**/dev/sdc**、**/dev/sdd**

  Asianux Server 4 より新しい libata ドライバーが導入されたため、i686、x86_64 アーキテクチャでは IDE デバイス名は /dev/sda、/dev/sdb のように SCSI デバイスとして表示されるようになりました (ただし PPC アーキテクチャには適用されません)。
  IDE 接続の CD-ROM は通常 OS のインストール時に **/dev/sr0** のシンボリックリンクファイルが実際のデバイスファイルを示すように作成されているため、CD-ROM を指定する場合には、デバイスファイルとして **/dev/cdrom** を指定することが一般的です。

- CD/DVD-ROM デバイス
  - **/dev/cdrom**

  CD/DVD-ROM ドライブを表します。

- フロッピーデバイス
  - **/dev/floppy**

  フロッピーデバイスを利用する場合、一般には **/dev/floppy** を指定します。ただし、特殊な用途では **/dev/fd0h1660** などのデバイスファイルを用いる場合もあります。

  Asianux Server 4 では、最初に **modprobe** コマンドを用いて、フロッピードライブのモジュールを読み込む必要があります。

```
# modprobe floppy
```

  上記コマンドを実行することで、**/dev/floppy** にアクセスすることが可能になります。

  フロッピーディスクはマウントしてファイルの操作を行うこともできますが、代わりに、マウントなしでフロッピーディスク専用のコマンド (例えば mdir など) を使用して操作することもできます(ただし、FAT 形式でフォーマットされたフロッピーディスクに限ります)。

通常、デバイスファイルを新たに作成することはあまりありませんが、ストレージデバイスなどで大量にディスク装置を追加した場合や、特殊なハードウェアのためにデバイスファイルが新たに必要になった場合には 、

mknod コマンドでデバイスファイルを作成します。なお、下記のコマンドの「b」はブロックデバイスを意味します。キャラクターデバイスの場合は「c」を指定します。

```
# /bin/mknod /dev/newdev b [major 番号] [minor 番号]
```

1.2 パーティション

1 つのハードディスク上で論理的に分割された各領域のことを**パーティション**と呼びます。個々のパーティションは、それぞれ 1 つのハードディスクのように利用できます。パーティションはディスクの管理を容易にしたり、1 台のコンピュータを複数の OS を切り替えながら使用したりするために作成されます。

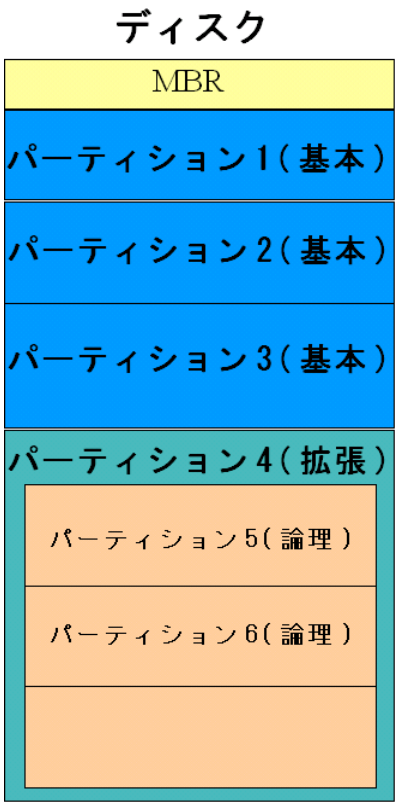
PC/AT 互換機では、1 つのハードディスクを最大 4 つのパーティションに分割できます。これらのパーティション情報は **MBR** (Master Boot Record: ディスクの一番先頭のメモリー) 中のパーティションテーブルに格納されます。

このパーティションテーブルに登録されているパーティションを「基本パーティション」と呼びます。4 つ以上のパーティションが必要な場合はこの 4 つの基本パーティションのうち、1 つを「拡張パーティション」にすることができます。拡張パーティションの中には複数の「論理パーティション」を作成でき、パーティションの合計最大数は IDE ディスク、SCSI ディスクともに 15 個となります。ただし、ディスクアレイを使用する場合には作成できるパーティションの数が制限されることがあります。詳細については、ディスクアレイコントローラーの各メーカーに問い合わせてください。

DOS や Windows 系のシステムでは、「MS-DOS 領域」や「論理 MS-DOS ドライブ」などの言葉を使用しますが、これらとパーティションは次のように対応すると考えて良いでしょう。

- **基本 MS-DOS 領域** —— 基本パーティション
- **拡張 MS-DOS 領域** —— 拡張パーティション
- **論理 MS-DOS ドライブ** —— 論理パーティション

パーティションの作成例を右に示します。



1.2.1 パーティション分割のメリット

1 つのディスクを単一のパーティションではなく、わざわざ複数のパーティションに分割することにはどのような利点があるのでしょうか。ここではLinux で一般的に行われているパーティション分割に対するメリットについて説明します。

- **ファイルシステム障害の局所化**
  システム運用中に不意にシステムのトラブルに遭遇することは珍しいことではありません。原因はさまざまですが、システムのトラブルによってファイルシステムの一部が破壊されることがあります。また、ディスクの不調や故障により特定のブロックが読めなくなる場合もあります。このような場合に備えて、ディスクを複数のパーティションに分割することで、障害発生時の被害を特定のパーティションだけに抑えられる場合があります。

- **ディスク容量不足によるトラブルの防止**
  適切なパーティション分割により、ファイルシステムの空き領域が不足することで発生するシステム異常の被害を最小限に抑えることができます。あるユーザーがパーティションを分割せずにシステムを使用しているとしましょう。このユーザーが自分のホームディレクトリーに巨大なファイル (たとえばCD イメージなど) を複数個置いたところ、ファイルシステムの空き領域がなくなり、そのファイルシステム上において新しいファイルを一切作成できない状態になってしまいました。
  この状態では、システムプログラムが **/etc** や **/var** 配下のファイルを修正したり、**/tmp** などに一時ファイルを作成したりすることもできないため、システムの運用に支障をきたすことがあります。**/home** を独立したパーティションに割り当てている場合、ファイルを作成できなくなるのは **/home** の中だけにとどまるため、 システム全体に影響を与えずに済みます。

- **性能劣化の防止**

  システムが使用するディレクトリーの中には、そのディレクトリー内にあるファイルの生存期間 (ファイルが作成されてから削除されるまでの期間) に特徴を持つものがあります。たとえば、**/var** は数多くのファイルが作成・削除・修正される場所なので、生存期間が短いファイルが集まっているディレクトリーだと言えます。**/usr** の場合は逆に、一度アプリケーションをインストールすれば、そのアプリケーションをアップデートするまでの比較的長い間関連ファイルが存在する (大半のプログラムはアップデートされずインストールしたときのままの状態で存在する) ので、ファイルの生存期間が長いといえるでしょう。多くの Linux ディストリビューションで標準として使われているファイルシステム ext4 は、ファイルに対して連続したブロックを割り当てることでファイルアクセス性能を向上させます。しかし、ファイルの作成や削除が頻繁に起こるような状況で長期間運用を続ける場合、フラグメンテーションと呼ばれる領域の「虫食い状態」が発生して、連続したブロックを割り当てられなくなり性能の劣化が発生します。よって、性能を重要視するシステムではファイルの生存期間を考慮してパーティション分割を行うことが推奨されます。たとえば、ファイルの生存期間が異なる **/var** は、独立したパーティションに分割するのが良いでしょう。

- **複数 OS の共存**

  パーティションを分割することによって、1 台のハードディスク上に複数の OS (たとえば、Linux と Windows) を共存させることができます。1 つの OS には最低 1 つのパーティションを割り当てる必要があります。ただし Asianux Server 4 の主な用途はサーバーシステムのため、複数の OS を共存させて運用することは推奨していません。

パーティション分割に関する詳細な説明は、JF のウェブサイトなどを参考にしてください。
`http://linuxjf.sourceforge.jp/JFdocs/INDEX-diskmanage.html`

1.2.2 パーティション分割候補のディレクトリーと分割例

Linux をサーバーとして運用する場合、どのようにパーティションを分割するのが良いのでしょうか。またそのサイズはどれだけ確保すれば良いのでしょうか。一般的には、次に示すようなディレクトリーがパーティション候補として挙げられます。

- **swap**
- **/boot**
- **/ (ルート)**
- **/usr**
- **/home**
- **/var**
- **/tmp**
- **/opt**

Linux で一般的に利用されるディレクトリーは、FHS (Filesystem Hierarchy Standard, `http://www.pathname.com/fhs/`) によって定義されています。ここでは、パーティション候補として挙げた各ディレクトリーの役割について説明します。

- **swap**

  Linux のスワップパーティションです。Linux のメモリー管理システムは、ページと呼ばれる単位 (Intel 386 系 CPU の場合、1 ページは通常 4KB) でメモリーを管理しています。システムに搭載しているメモリーよりも多くのメモリーが必要になる場合、参照頻度の低いページをハードディスク上に用意されているスワップパーティションに移動します。よってシステムに搭載しているメモリーのサイズとスワップパーティションとして用意したサイズの合計が、仮想記憶領域 (OS が利用できるメモリー領域) のサイズになります。Linux ではファイルの生存期間の観点から、通常スワップパーティションを他のファイルシステムとは別のパーティションに確保します。実際にスワップパーティションがどれぐらい必要になるかは、システム設計の範疇に含まれます。運用するシステムの高負荷時に必要な仮想記憶領域のサイズを想定し、メモリーサイズとスワップサイズの合計がその範囲より大きければ問題ないでしょう。

- **/boot**

カーネルイメージをはじめとする、Linux の起動に必要なファイルがこのディレクトリー配下に存在します。したがってブートローダーからこの配下のファイルが確実に読めなければ、システムを起動できません。BIOS の仕様や不具合によってブートに必要なデータを読めないこともありますので、**/boot** は別パーティションに確保して、なるべくディスクの先頭に位置させておくことを推奨します。また ソフトウェア RAID を使用して **/boot** パーティションもソフトウェア RAID の対象範囲に含めた場合、正常にブートできないことがあります。このようなことを考慮して、**/boot** を独立したパーティションとしてソフトウェア RAID の制御対象外にするのが良いでしょう。
容量は多くの場合 250MB 程度確保しておけば十分なはずです。

- **/ (ルート)**

「ルートディレクトリー」と呼ばれるシステム全体のファイルシステムの最上位のディレクトリーです。Linux ではこのパーティションが必ずどこかに確保されている必要があり、システム起動時にマウントされます。システムに必要な情報ファイルや、システムの起動に必要なコマンドがこのパーティション内に存在します。ファイルシステムの最上位に位置するので、他のパーティションとの関係によって、必要な容量が変わります。

- **/home**

通常、ユーザーのホームディレクトリーがこの配下に置かれ、ユーザーのデータファイルなどがここに置かれることになります。ファイルの生存期間は中程度でしょう。このディレクトリーも独立したパーティションに確保することが望まれます。ユーザーが作成したファイルが格納されるので、運用期間が経過するにつれて、ディスク使用量が増加していくことが一般的です。

- **/usr**

OS のプログラムやライブラリーがこのディレクトリー配下にインストールされます。ファイルの生存期間が比較的長いものが集まっており、性能の観点から独立したパーティションを確保することが望まれます。必要なパーティションサイズはインストールするパッケージによって変わりますが、Asianux Server 4 のインストール時にすべてのパッケージをインストールした場合、8GB 強の容量が必要となります。しかし、システム運用時には年々新たなパッケージやプログラムをインストールすることになるため、**/home** と同様に **/usr** も増加していくことが一般的です。 よって、あらかじめ余裕を持って領域を確保しておきましょう。また 、**/usr/local ディレクトリー**には、ユーザーが独自にインストールしたソフトウェアのプログラムやライブラ

リーが置かれるので、システム構成によっては、**/usr/local** を別パーティションとして確保しても良いかもしれません。

- **/var**

システムのログやスプールファイルなどが、ここに作られます。プログラムによっては **/var/cache** にキャッシュファイルを作ったり、**/var/tmp** に一時ファイルを作ったりするものもあります。ファイルの生存期間が比較的短いものが集まっており、システムの運用状態によってはファイルが次々に作られることもあるので、ルートパーティションとは分けて確保しておくことが望ましいでしょう。パーティションのサイズは最低でも 1GB 程度確保することを推奨しますが、メールサーバーや HTTP サーバーのデフォルト設定では、このディレクトリー配下にメールや HTML ファイルを配置するのでシステムの運用規模に応じた領域を確保しておく必要があります。
なお Asianux Server 4 では仮想化機能の KVM を利用することができ、仮想マシンのイメージファイルはデフォルトで **/var/lib/libvirt/images** に配置されます。仮想化を使用する場合は、/var を大きくすることや別な場所を利用するなどの配慮が必要となります (ちなみに /var の容量を増やすことが難しい場合は **/etc/libvirt/storage/default.xml** を編集し、仮想マシンのイメージ配置場所を変更することも可能です)。

- **/tmp**

/tmp は特殊なディレクトリーでだれもが書き込み可能なディレクトリーです。一時ファイルをこのディレクトリー配下に作成するプログラムも数多く存在します。しかしだれもが書き込み可能なディレクトリーのため、便利である一方で危険な一面もあります。悪意のあるユーザーやプログラムのバグによって自由に大きなファイルを **/tmp** ディレクトリーに作成できるため、**/tmp** ディレクトリーが / (ルート) と同じパーティション内に存在する場合、/ (ルート) パーティションの空き領域がなくなりシステムに異常をきたす可能性があるのです。よってシステムをより安全に運用するためには、/ (ルート) パーティションとは別のパーティションにすることを推奨します。

- **/opt**

  /opt はアプリケーションのインストール先として利用されるディレクトリーです。商用アプリケーションの多くは **/opt** ディレクトリーにアプリケーションのプログラムやデータなどをインストールします。したがって確保すべき容量はインストールするアプリケーションに依存します。OS のインストール時には何もインストールされないので、ディストリビューションのリポジトリから提供されているもの以外のプログラムを使わないのであれば、別パーティションを確保する必要はありません。

システム構築時には、表 1-1 に示すパーティション分割の設定例を参考にしてください。メールサーバーなどを構築する場合には、**/var** により大きな領域を割り当てる必要があるでしょう。

表 *1-1*　パーティションの設定例

パーティション	容量	
	メモリー **2GB** / ディスク **2TB** のファイルサーバー用のシステム	メモリー **4GB** / ディスク **1TB** のデータベースサーバー用のシステム
/boot	250MB	250MB
swap	4GB	8GB
/	50GB	50GB
/var	1.7TB (ファイル共有に利用)	1GB*
/tmp	1GB	1GB
/home	残り全部 (ユーザー用ファイル共有に利用)	1GB
/opt	なし	残り全部 (データベースに利用)*

* MySQL、PostgreSQL の場合は残り全部を /var に、Oracle DB の場合は /opt に割り当てます。

1.2.3 パーティションの作成

パーティションの作成を行う前に理解しておかなければいけないことは、「既存のパーティションサイズを変更することは容易ではない」という点です。パーティションサイズを変更するという作業は、実質的には古いパーティションを削除して、新規にパーティションを作るという作業と等価です。ファイルシステムの内容を保持したままパーティションを拡張、縮小するという作業は難しいため、必ずデータのバックアップを他のデバイスなどに取って新規パーティションとして確保し直してから、バックアップしていたデータをリストアするという作業が必要になります。

次に大事なことは、「新規パーティションは、ディスク上の連続した未使用領域に対してしか作成できない」ということです。このためディスク上に複数の未使用領域が存在していても、連続していない領域をまとめて 1 つのパーティションとすることはできません。したがって通常パーティションを追加する場合には、ディスクの最後尾の未使用領域に新たなパーティションを作成することになります。

このように一度使い始めたパーティションを変更することは非常に大変なため、システムインストール前にパーティション割り当てを十分検討することが大切です。

1.2.4 fdisk によるパーティション操作

fdisk コマンドは伝統的に Linux のパーティション操作に用いられてきたコマンドで、ハードディスクのパーティションを新規に作成したり、あるいは既存のパーティションを削除したりできます。**fdisk** の引数にはデバイスファイルを指定します。たとえば **/dev/sda** (1 つ目の SCSI ディスク) に関する操作を行う場合には次のコマンドを実行します。

```
# fdisk /dev/sda
```

起動後に「m」を入力して [Enter] キーを押すとヘルプが表示されるので、必要なコマンドを入力します。

「p」を入力すると、現在操作中のディスクのパーティション情報が表示されます。

```
コマンド (m でヘルプ): p

ディスク /dev/sda: 16.1 GB, 16106127360 バイト
ヘッド 255, セクタ 63, シリンダ 1958
Units = シリンダ数 of 16065 * 512 = 8225280 バイト
セクタサイズ (論理 / 物理): 512 バイト / 512 バイト
I/O size (minimum/optimal): 512 bytes / 512 bytes
ディスク識別子: 0x000112aa

デバイス      ブート  始点    終点        ブロック Id    システム
/dev/sda1     *       1       26          208813+    83      Linux
/dev/sda2             27      91          522112+    82      Linux スワップ
/dev/sda3             92      665         4610655    83      Linux
/dev/sda4             666     1958        10386022+  f       Win95 拡張領域 (LBA)
/dev/sda5             666     1958        10385991   83      Linux
```

新規にパーティションを作成する場合は「n」を入力して [Enter] キーを押し、対話式にパーティションの作成を行います。最初に基本パーティションか拡張パーティションか聞かれるので、4 つ目以降のパーティションであれば拡張パーティションを選択します。通常は基本領域から作成していきます。

```
# fdisk /dev/sdd
コマンド（m でヘルプ）: n
コマンドアクション
   e   拡張
   p   基本領域（1-4）
p
領域番号（1-4）: 1
```

既に存在するパーティションの番号を指定した場合、「パーティション 1 は定義済みです。まずは削除を行ってください」のように表示されます。この場合は別の値を指定してください。

続いてパーティションのサイズを指定しますが、最初にパーティションの開始位置が聞かれます。デフォルトでは未使用領域の先頭が初期値になっているので、特に変更がなければ [Enter] キーを押します。

続いてパーティションの最後尾またはパーティションのサイズを指定します。サイズを指定するときには「+500M」のように指定します (MB 単位の場合)。

```
最初 シリンダ (1-1024, 初期値 1):          ←<Enter>を入力
初期値 1 を使います
終点 シリンダ または +サイズ または +サイズ M または +サイズ K (1-1024, 初期値 1024):+500M
```

以上で、メモリー上にパーティション情報が作成されました。

```
コマンド (m でヘルプ): p

ディスク /dev/sdd: 1073 MB, 1073741824 バイト
ヘッド 64, セクタ 32, シリンダ 1024
Units = シリンダ数 of 2048 * 512 = 1048576 バイト

 デバイス ブート   始点      終点    ブロック   ID  システム
/dev/sdd1       1     478   489456   83  Linux
```

この時点ではディスクにはパーティションの情報が反映されていないので、サイズを間違えたりした場合には「d」を入力してパーティションを削除してから、再度パーティションを作成します。

スワップパーティションを作成する場合には、パーティションのシステム ID を 82 に変更しなければいけません。システム ID の変更は「t」を入力します。また、Windows で使われている FAT などのパーティションを作る場合にもシステム ID を変更します。システム ID 一覧はシステム ID の入力時に「L」を入力することで表示されます。

```
コマンド (m でヘルプ): t
領域番号 (1-4): 2
16 進数コード (L コマンドでコードリスト表示):82
領域のシステムタイプを 2 から 82 (Linux スワップ) に変更しました

コマンド (m でヘルプ): p

ディスク /dev/sdd: 1073 MB, 1073741824 バイト
ヘッド 64, セクタ 32, シリンダ 1024
Units = シリンダ数 of 2048 * 512 = 1048576 bytes

  デバイス ブート      始点        終点    ブロック      ID  システム
/dev/sdd1               1         478      489456      83  Linux
/dev/sdd2             479         670      196608      82  Linux スワップ  <- システム ID
変更済
```

パーティションの作成を完了したら **w** を入力して、変更をディスクに反映させます。いったんパーティション情報をディスクに反映したら元の状態に戻すことはできないので、既存のパーティション情報を変更する場合には間違いがないことを十分確認してから、パーティション情報を反映させましょう。

```
コマンド (m でヘルプ): w
領域テーブルは交換されました！

ioctl() を呼び出して領域テーブルを再読込みします。
ディスクを同期させます。
```

なお、以下のようなエラーが発生する場合もあります。

```
コマンド (m でヘルプ): w
領域テーブルは交換されました！

ioctl() を呼び出して領域テーブルを再読込みします。

警告: パーティションテーブルの再読込みがエラー 16 で失敗しました: デバイスもしくはリソースがビジー状態です。
カーネルはまだ古いテーブルを使っています。新しいテーブルは次回リブート時か、partprobe(8)またはkpartx(8)を実行した後に使えるようになるでしょう。
ディスクを同期しています。
```

この場合はまだパーティションの変更がカーネルに認識されていません。再起動することでパーティションの変更が認識されます。再起動を行いたくない場合、メッセージ通りに **partprobe** または **kpartx** コマンドを実行することでも変更を認識させることができます。

以上でパーティションの作成は完了です。

1.2.5 parted によるパーティション操作

GNU Parted は fdisk よりも操作が簡単な上に、パーティションのリサイズやコピー機能も備えた優れたパーティショニングツールです。

Parted でパーティション設定を行うには次のコマンドを実行します。ここでは /dev/sdc（3 つ目の SCSI ディスク) を開きます。

```
# parted /dev/sdc
```

すると、**(parted)** という風にプロンプトが現れます。**help** コマンドを入力すると使用できるコマンド一覧が表示されます。

現在のパーティション情報を表示するには、**print** コマンドを入力します。

```
(parted) print

モデル: Virtual HDD [2] (ide)
ディスク /dev/hdc: 1049MB
セクタサイズ (論理/物理): 512B/512B
パーティションテーブル: msdos

番号  開始     終了     サイズ   タイプ   ファイルシステム フラグ
 1    32.3kB  1045MB           1045MB           primary
```

新規にパーティションを作る場合は **mkpart** コマンドを使用します。**mkpart** コマンドは次のように引数を指定して作成するか、引数を指定せずに実行し、対話的に作成を行うかを選択することができます。

```
(parted) mkpart primary ext4 32.3KB 1045MB
```

対話的に作成を進める場合は、次のような流れになります。

```
(parted) mkpart
パーティションの種類?  primary/プライマリ/extended/拡張? primary
ファイルシステムの種類?  [ext2]? ext4
開始? 32.3KB
終了? 1045MB
```

なお、場合によっては以下のようなメッセージが表示され、若干開始位置と終了位置が変更される場合があります。それで問題なければ y を入力してください。

```
警告: 32.3KB から 1045MB までのパーティションを指定されました。
可能な中で最も近いものは 32.4KB から 1045MB になります。
それでもかまいませんか？
はい(Y)/Yes/いいえ(N)/No? y
```

また、fdisk 同様に以下のようなメッセージが出る場合があります。その場合は、再起動することでパーティションの変更結果を適用することができます。

```
警告: WARNING: the kernel failed to re-read the partition table on /dev/sda
(デバイスもしくはリソースがビジー状態です).  As a result, it may not reflect all of your changes
until after reboot.
```

その他のコマンドの使用法については、**man parted** を実行し、参照してください。

なお、Asianux Server 4 に含まれる GNU Parted 2.1 では、パーティションのリサイズを行うコマンド **resize** は ext4 に対応していないことに注意してください。Parted コマンドの代わりに **resize2fs** コマンドが利用可能です。

パーティションの作成を完了したら、**quit** コマンドを入力して変更をディスクに反映させます。いったんパーティション情報をディスクに反映したら元の状態に戻すことはできないので、既存のパーティション情報を変更する場合には間違いがないことを十分確認してから、パーティション情報を反映させましょう。

1.3　ファイルシステム

fdisk などでパーティションを作成しただけでは、そのパーティションを利用することはできません。OS がそのパーティションを利用するためには、そのパーティション上に**ファイルシステム**を作成しなければいけません。ファイルシステムとは OS がファイルを管理するための枠組みであり、Asianux Server 4 では ext2、ext3、ext4 などのファイルシステムを利用できます。

新しいパーティション上にファイルシステムを作成すると、スーパーブロックと呼ばれる管理情報がパーティション内に作成されて、そのパーティションを利用することが可能となります。

ext4 ファイルシステムは**ジャーナリングファイルシステム**と呼ばれるジャーナリング機能を持っています。ジャーナリング機能はファイルシステムの信頼性を向上させるための機能の 1 つです。ジャーナリングファイルシステムにおける「ジャーナル (Journal) 」とは、ファイルシステムの変更に対する操作をあらかじめ準備された領域にログとして記録することを意味します。ジャーナリングファイルシステムは、障害からの復旧時にジャーナルの情報を利用してファイルシステムの復旧を行い、ファイルシステムの一貫性を保つことができます。

一方、Linux の初期の頃から利用されてきた ext2 ファイルシステムはジャーナリング機能を持っておらず、システム障害時などファイルシステムを正常にアンマウントできなかった場合、再起動後のマウント時に **fsck** コマンドによるファイルシステムの検査が行われます。この検査はファイルシステム内のすべてのファイルの一貫性を検査するので、ファイルシステムが大きくなると検査に必要な時間も延び、サービスの停止時間を延ばす要因となります。したがって、現在では ext2 以外のジャーナリングファイルシステムを用いてシステムを運用することが一般的になっています。

1.3.1　ext4 ファイルシステム

ext4 ファイルシステムは、Linux の初期段階から利用されてきた ext ファイルシステムの最新版です。ext4 ファイルシステムは ext3 ファイルシステムと上位互換であり、既存の ext3 ファイルシステムを ext4 ファイルシステムに変更したり、ext3 ファイルシステムを ext4 ファイルシステムとしてマウントしたりすることが簡単にできます。

ext4 ファイルシステムの操作は **e2fsprogs** パッケージに含まれているツールを用います。また ext3 ファイルシステムの操作も ext4 と同じ操作で行うことができます。

（1）ext4 ファイルシステムの作成

ext4 ファイルシステムを新規に構築するには、**mkfs** のオプションとして、ファイルシステムの種類を表す **-t ext4** オプションと、ext4 ファイルシステムを作成するパーティションのデバイスファイルを指定します。

```
# mkfs -t ext4 /dev/sdd1
```

また既存の ext3 ファイルシステムを ext4 ファイルシステムに変換できます。ext3 ファイルシステムを ext4 ファイルシステムに変換するためには、**tune2fs** コマンドを使用します。ファイルシステムの変換はマウント中でも行うことができます。次の手順は **/dev/sda3** 上に作成された ext3 ファイルシステムを ext4 ファイルシステムに変換する例です。このとき **/dev/sda3** 上のデータはすべて保持されます。tune2fs コマンドのオプション -O の引数はファイルシステムに追加する機能群です。

```
# tune2fs -O extents,uninit_bg,dir_index /dev/sda3
```

その後ファイルシステムが正常か否かチェックするため、**e2fsck** コマンドを実行します。**e2fsck** はマウント中は実行できませんので、当該のファイルシステムをアンマウントしてから行う必要があります。

```
# e2fsck -pf /dev/sda3
```

この後、**/etc/fstab** を編集します。以下のような行があるはずですので、ext3 の部分を ext4 に書き換えてください。

```
/dev/sda3 /home ext4 defaults 0 2   # “ext3” から “ext4” に書き換える
```

（2）ext4 ファイルシステムのマウント

作成した ext4 ファイルシステムは **mount** でファイルツリー上にマウントします。次の例では、**/dev/sda3** を **/mnt/asianux1** にマウントします。

```
# mount -t ext4 /dev/sda3 /mnt/asianux1
```

（3）ext4 ファイルシステムのラベル設定

ext4 ファイルシステムにはラベルを設定できます。ラベルを用いることの利点は、デバイスの指定時にデバイスファイル名ではなくラベルによってファイルシステムを特定できることです。この機能により、SCSI デバイスを用いて運用しているシステムで、SCSI デバイスの追加・削除などによってデバイスファイルの割り当てが変更されても、システムの運用に影響を与えなくなります。

ext4 ファイルシステムにラベルを指定するためには、**e2label** を利用します。次の例は、**/dev/sda3** にラベル「asianux1」を指定しています。

```
# e2label /dev/sda3 "asianux1"
```

現在、ファイルシステムに設定されているラベルを確認したいときには、ラベル名を付けずに **e2label** を実行します。

```
# e2label /dev/sda3
asianux1
```

（4）/etc/fstab の変更

作成したファイルシステムをシステムの再起動時に自動的にマウントするためには、**/etc/fstab** に記述を追加します。次の例は **/dev/sda3** デバイスを **/mnt/asianux1** ディレクトリーにマウントするための設定例です。

```
/dev/sda3     /mnt/asianux1  ext4  defaults 0 0
```

UUID を利用して設定する場合には次のように設定します。

```
UUID=d7f5052f-5b69-4f30-bcec-a7f300c7f005 /mnt/asianux1       ext4    default 0 0
```

デバイスの UUID の調べ方は次項の 1.3.1（5）UUID の操作をご覧ください。

また、ラベルを利用して指定する場合には、次のように設定します。

```
LABEL=asianux1  /mnt/asianux1  ext4  defaults 0  0
```

（5）UUIDの操作

Linuxでは、各デバイスにUUIDと呼ばれる一意のIDが割り振られています。これらを自ら指定する必要がある場合は、以下のようにして操作することが可能です。

以下のコマンドにおいて、”a7de1b36-e9de-4de9-966f-7938076bb27f” は適切なUUIDに 、**/dev/sda3** は任意のデバイス名に置き換えてください。

UUIDを指定するには以下のコマンドを実行します。

```
# mkfs.ext4 -U "a7de1b36-e9de-4de9-966f-7938076bb27f" /dev/sda3
```

デバイスのUUIDを確認するためには **tune2fs** コマンドの **-l** オプションを利用します。**tune2fs -l** の実行結果では指定したデバイスの様々な情報が表示されますので、**grep** でUUIDに関する行のみを抜き出しましょう。

```
# tune2fs -l /dev/sda3 | grep UUID
Filesystem UUID:       78cf85fc-ae55-4490-9fa6-f61930dc422f
```

UUIDの変更は以下のようにして行うこともできます。

```
# tune2fs -U "a7de1b36-e9de-4de9-966f-7938076bb27f" /dev/sda3
```

ランダムなUUIDを生成したい場合には、**uuidgen** コマンドを実行することで生成できます。

```
$ uuidgen
0ab9b284-b551-432a-9865-393871beb9b9
```

つまり以下のようにすることで適当なUUIDを自動生成し、それを指定したデバイスのUUIDとすることができます。

```
# tune2fs -U `uuidgen` /dev/sda3
```

なお **blkid** コマンドを利用することで設定された UUID を確認することができます。

```
# blkid /dev/sda3
```

1.4 RAW デバイス

通常のファイルシステムはディスクに対する I/O 処理の際に、カーネル内部の**ページキャッシュ**と呼ばれるキャッシュにいったんデータをコピーしてから、ページキャッシュ内のデータを入出力します。ページキャッシュにデータをコピーしておくことで、読み込み要求に対しては同じデータを何度もディスクから読む必要がなくなり、書き込み要求に対しては実際のディスクに対する書き込みを遅延させたりできるため、I/O 性能の向上に効果を発揮します。

一方で、商用データベースなどではディスクに対する I/O データは、データベースのメモリー管理システム内のバッファにおいて管理されていて、データベースプログラムとカーネルの 2 箇所でバッファを管理することによってオーバーヘッドが発生してしまいます。

そこで、特定のデバイスに対して行われる I/O 要求はページキャッシュを経由しない方法が実装され、この機能が RAW デバイスとして提供されています。

1.4.1 RAW デバイスの利用

RAW デバイスはパーティション単位で管理します。したがって、RAW デバイスを利用したい場合には、まず RAW デバイス用のパーティションを作成します。

たとえば、**/dev/sdd1** ~ **/dev/sdd4** までのパーティションを RAW デバイス用に作成したとします。これらのパーティションを RAW デバイスとして利用するためには、**raw** を利用して、それぞれのパーティションを RAW デバイスにバインドします。

```
# /bin/raw /dev/raw/raw1 /dev/sdd1
/dev/raw/raw1:  bound to major 8, minor 49
# /bin/raw /dev/raw/raw2 /dev/sdd2
/dev/raw/raw2:  bound to major 8, minor 50
# /bin/raw /dev/raw/raw3 /dev/sdd3
/dev/raw/raw3:  bound to major 8, minor 51
# /bin/raw /dev/raw/raw4 /dev/sdd4
/dev/raw/raw4:  bound to major 8, minor 52
```

以上の操作によって、**/dev/raw/raw1 ~ /dev/raw/raw4** までが、RAW デバイスとして利用可能になりました。

RAW デバイスのバインド状況を確認したいときには、**raw** コマンドの **-qa** オプションを使います。

```
# /bin/raw -qa
/dev/raw/raw1:  bound to major 8, minor 49
/dev/raw/raw2:  bound to major 8, minor 50
/dev/raw/raw3:  bound to major 8, minor 51
/dev/raw/raw4:  bound to major 8, minor 52
```

なお、RAW デバイスとして利用している(今回の例の場合、**/dev/sdd1 ~ /dev/sdd4**)　に、ファイルシステムを作成して利用してはいけません。ファイルの不整合が発生する可能性があります。

1.4.2　RAW デバイスの起動設定

システム起動時に自動的に RAW デバイスをバインドするには、**/etc/udev/rules.d/60-raw.rules** ファイルを設定します。次の設定例は、**/dev/sdd1 ~ /dev/sdd4** を、**/dev/raw/raw1 ~ /dev/raw/raw4** に自動的に設定するためのものです。

```
ACTION=="add", KERNEL=="sdd1", RUN+="/bin/raw /dev/raw/raw1 %N"
ACTION=="add", KERNEL=="sdd2", RUN+="/bin/raw /dev/raw/raw2 %N"
ACTION=="add", KERNEL=="sdd3", RUN+="/bin/raw /dev/raw/raw3 %N"
ACTION=="add", KERNEL=="sdd4", RUN+="/bin/raw /dev/raw/raw4 %N"
```

設定が完了したら再起動を行い、**raw -qa** コマンドでデバイスが自動でバインドされているかを確かめます。

1.5　ソフトウェア RAID

RAID は Redundant Array of Inexpensive Disks の略で、安価なディスクを組み合わせて信頼性の高い大容量ディスクアレイを形成しようというものです。実際の機能としてはパーティションを組み合わせて RAID 構成を作りますが、1 台のディスクを複数パーティションに分けて組み合わせても意味がありません。高性能、高信頼性を得るためには、複数台のディスクで RAID システムを構成してください。

ここで説明するソフトウェア RAID は RAID の機能をカーネルで実現します。したがって RAID コントローラーなどのハードウェアがない場合にも利用することが可能です。ただし、RAID コントローラーで RAID の機能を実現しているハードウェア RAID と比較すると、ソフトウェア RAID では I/O 処理において RAID 機能の処理のために余分に CPU を使用しますので、I/O パフォーマンスが低下することがあります。

RAID の種類にはリニアモードと RAID レベルというものがあります。以下に Linux のソフトウェア RAID でサポートしているものについて説明します。

- **リニアモード**

  複数のディスクを単純に結合して大容量のディスクを作成します。1 台のディスクが壊れるとすべてのデータを失う恐れがあるので信頼性は低く、また性能向上もほとんど望めません。

- **RAID-0**

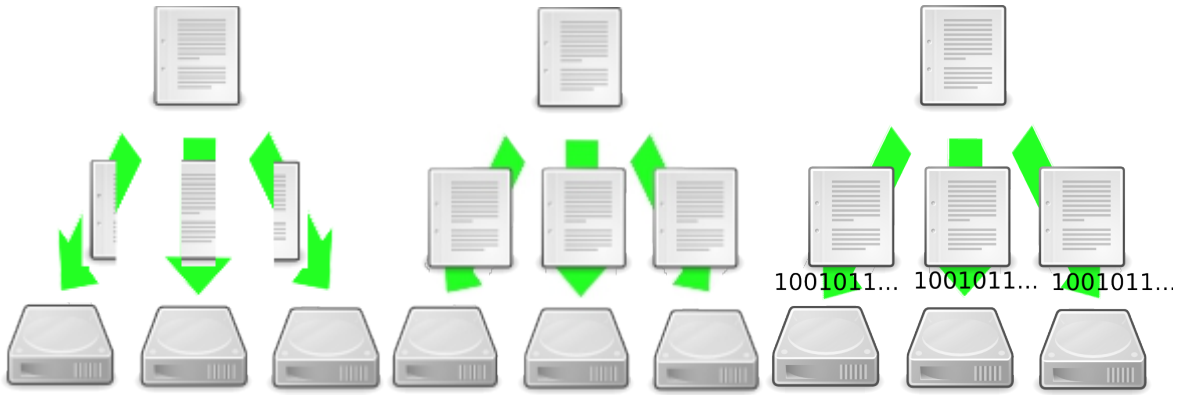
  ストライピングと呼ばれ、データを分割して複数ディスクに書き込みます。これにより I/O 性能が向上しますが、1 台でもディスクが壊れるとすべてのデータを失うことになります。

- **RAID-1**

  ミラーリングと呼ばれ、1 台のディスクの完全なコピーを他のディスクに保持します。信頼性は高くなりますが、I/O の性能は 1 台のディスクよりも低下します。

- **RAID-5**

データの書き込み時にデータのパリティ情報も書き込み、データとパリティ情報を複数のディスクに分散して書き込みます。3 台以上のディスクで構成され、1 台のディスクが壊れてもデータを復旧できます。



RAID-0　　RAID-1　　RAID-5

1.5.1　ソフトウェア RAID の作成

ソフトウェア RAID の作成にはまず少なくとも 2 つのパーティションを用意します。RAID 5 であれば最低 3 つのパーティションが必要となります。実際の運用では、それらのパーティションがそれぞれ別のディスク上になければ意味がありませんが、ソフトウェア RAID のテストであれば同じディスク上にあっても特に問題はありません。それぞれのパーティションサイズが同じである必要はありませんが、異なるサイズのパーティションを利用した場合利用されない領域が発生するので、ディスク領域を効率よく利用するためには等しいことが望ましいでしょう。

ソフトウェア RAID として利用するパーティションは、パーティションの ID を **0xFD** に設定しておくと起動時に自動的にソフトウェア RAID 用のパーティションとして認識されます。**fdisk** でパーティションを作成する時点で、パーティションの ID を **0xFD** に設定しておきましょう。

```
# /sbin/fdisk /dev/sdd

コマンド (m でヘルプ): p

Disk /dev/sdd: 1073 MB, 1073741824 bytes
64 heads, 32 sectors/track, 1024 cylinders
Units = シリンダ数 of 2048 * 512 = 1048576 bytes

 デバイス ブート     始点        終点   ブロック     ID   システム
/dev/sdd1             1       102     104432   83   Linux

コマンド (m でヘルプ): t
Selected partition 1
16 進数コード (L コマンドでコードリスト表示): fd
領域のシステムタイプを 1 から fd (Linux raid 自動検出) に変更しました

コマンド (m でヘルプ): p

Disk /dev/sdd: 1073 MB, 1073741824 bytes
64 heads, 32 sectors/track, 1024 cylinders
Units = シリンダ数 of 2048 * 512 = 1048576 bytes

 デバイス ブート     始点        終点   ブロック     ID   システム
/dev/sdd1             1       102    104432    fd   Linux raid 自動検出

コマンド (m でヘルプ): w
領域テーブルは交換されました！

ioctl() を呼び出して領域テーブルを再読込みします。
ディスクを同期させます。
```

次に **/etc/mdadm.conf** ファイルに RAID システムの構成を記述します。

次の例は、**/dev/sdb1**、**/dev/sdc1**、**/dev/sdd1** を使用して RAID を構成するための設定です。

```
DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1
ARRAY /dev/md0 devices=/dev/sdb1,/dev/sdc1,/dev/sdd1
```

ソフトウェア RAID として構成したいデバイスのデバイスファイル名は、通常は **/dev/md0**、**/dev/md1** という順番で割り当てていきます。設定内容の詳細に関しては **mdadm.conf** のオンラインマニュアルを参照してください。

/etc/mdadm.conf の記述が完了したら、**mdadm** コマンドを使用し RAID デバイスを作成します。

次の例は **/dev/sdb1**、**/dev/sdc1**、**/dev/sdd1** を使用して RAID-5 を構成するためのコマンドです。

```
# /sbin/mdadm -C /dev/md0 -l 5 -n 3 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

以下、コマンドの各パラメーターについて説明します。

-C のパラメーターは RAID デバイスファイル名です。**-l** のパラメーターは RAID レベルで、今回は RAID5 なので、**5** です。**-n** のパラメーターは RAID 配列として接続するハードディスク (パーティション) の数です。今回は **/dev/md0** には **/dev/sdb1**、**/dev/sdc1**、**/dev/sdd1** の 3 つのパーティションが含まれていますので、**3** を指定します。最後にこの RAID に含まれる物理ディスク名を記述します。

詳細に関しては **mdadm** のマニュアルを参照してください。

ソフトウェア RAID の状態を調べるためには、次のように **/proc/mdstat** の参照及び、**mdadm -D /dev/md*** コマンドを利用します。

```
# /bin/cat /proc/mdstat
Personalities : [raid5]
md0 : active raid5 sdd1[2] sdc1[1] sdb1[0]
    196352 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
```

出力の 3 行目にある **[UUU]** は、それぞれの物理ディスクの状態を表しており、U になっている場合は正常に動作しているということです。正常に動作していない場合、**[UU_]** のように不具合のあるディスクがアンダーバーで示されます。

```
# /sbin/mdadm -D /dev/md0
/dev/md0:
        Version : 00.90.01
  Creation Time : Tue Jul 19 18:17:28 2005
     Raid Level : raid5
     Array Size : 196352 (191.75 MiB 201.06 MB)
    Device Size : 98176 (95.88 MiB 100.53 MB)
   Raid Devices : 3
  Total Devices : 3
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Tue Jul 19 18:18:54 2005
          State : clean
 Active Devices : 3
Working Devices : 3
 Failed Devices : 0
  Spare Devices : 0

         Layout : left-symmetric
     Chunk Size : 64K

    Number   Major   Minor   RaidDevice State
       0       3       65        0      active sync   /dev/sdb1
       1       3       66        1      active sync   /dev/sdc1
       2       3       67        2      active sync   /dev/sdd1
           UUID : 5a9fe0cd:e89a279b:29b60829:ec3065d2
         Events : 0.64
```

次のコマンドで RAID デバイスを停止し、すべてのリソースを解放することができます。

```
# /sbin/mdadm -S /dev/md0
```

また次のコマンドで定義済みの RAID デバイスを編成し、起動することができます。

```
# /sbin/mdadm -A /dev/md0
```

1.5.2　RAID の運用

mdadm で初期化が完了した RAID デバイスは、通常のパーティションのように扱うことができます。次のコマンドは **/dev/md0** に割り当てられた RAID デバイスに、ext4 ファイルシステムを構築しています。

```
# /sbin/mkfs -t ext4 /dev/md0
```

Asianux Server 4 では、カーネルの起動時に自動的に RAID デバイスを検出して、ソフトウェア RAID を起動します。ただし、パーティションの ID を **0xFD** にしておくことを忘れないようにしてください。

システム起動時にパーティションを自動的にマウントするためには、**/etc/fstab** にソフトウェア RAID デバイスの設定を追加します。

```
/dev/md0    /mnt/raid    ext4   defaults 1 2
```

1.5.3　RAID 障害

RAID を運用している中で物理ディスクが破損する場合があります。物理ディスクのうちのいくつかが故障したまま運用されている状態をデグレード状態と呼びます。デグレード状態であっても I/O 処理は正常に行われますが、できるだけ早急にディスクの交換を行うべきでしょう。ディスクが欠けている分データの冗長性が低減する上、同時期に購入して取り付けたディスクは故障する時期も同じくらいの時期になる可能性があるため、現在動作しているディスクもすぐに故障する可能性があります。

デグレード状態から復旧させるには、まず故障したディスクを取り外し新しいディスクを取り付けます。

スレーブではなくマスターのディスクが故障した場合、マスターに新しいディスクを取り付けるのではなくスレーブのディスクをマスターに取り付け、新しいディスクをスレーブに取り付ける必要があります。マスターにブートローダーがインストールされていないと起動ができないためです。

次に、**fdisk** などを用いて、新しいディスクに故障しなかったディスクと同様のパーティションを作成します。本ドキュメントの 1.2.4 fdisk によるパーティション操作 または 1.2.5 parted によるパーティション操作を参照してください。

パーティションが作成できたら、新しく取り付けたディスクを RAID デバイスとして追加します。

```
# mdadm /dev/md0 --add /dev/sdb1
```

(/dev/sdb1 は新しく取り付けたディスクの名前に置き換えてください。)

これで復旧作業は完了ですが、バックグラウンドでリビルドが行われていますので完全な復旧には時間がかかります。**cat /proc/mdstat** で、全てのディスクが認識されているか確認してください。

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb1[1] sda1[0]
      154296192 blocks [2/2] [UU]

unused devices: <none>
```

正常に認識されている場合は例えば [UU] のようになっています。[U_] のようにアンダーバーが表示されている場合は認識されていません。

リビルド中に **cat /proc/mdstat** を行うと、下記のように進捗が表示されます。

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb1[2] sda1[0]
      154296192 blocks [2/1] [U_]
      [>....................]  recovery =  1.7% (2729792/154296192) finish=44.6min
speed=56546K/sec

unused devices: <none>
```

1.6　LVM (Logical Volume Manager)

LVM (Logical Volume Manager) はユーザーが扱うパーティションとして、**論理ボリューム**と呼ばれる単位でパーティションを提供して物理的なディスクの存在を隠蔽します。その結果、物理的なディスクの増設や変更などがユーザーやアプリケーションに対して隠蔽されて、ディスクデバイス管理の柔軟性を向上させます。

LVM は、**物理ボリューム** (physical volume)、**ボリュームグループ** (volume group)、**論理ボリューム** (logical volume) から構成され、次の図のような構成で管理されます。

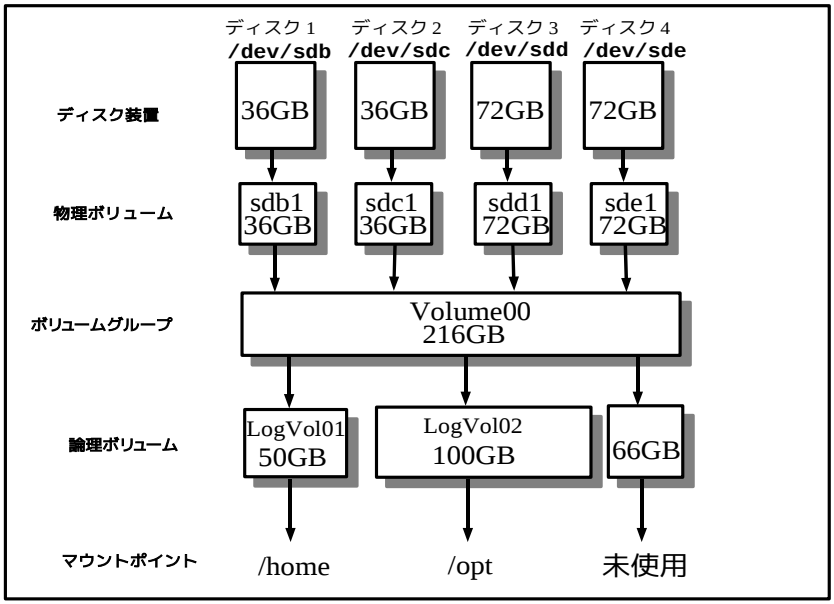


図 *1-1*　*LVM* の構成例

LVM の操作は、**lvm2** パッケージに含まれるツールによって行われます。

1.6.1 物理ボリュームの作成

物理ボリュームはパーティション単位で管理されます。したがって1 つのディスクの全体を 1 パーティションとし 1 つの物理ボリュームとしても構わないですし、一部分だけを LVM 用のパーティションとして確保して 1 つの物理ボリュームとしても構いません。もちろん 1 つのディスクを複数のパーティションに分割して複数の物理ボリュームを作成することもできます。

LVM 用のパーティションとするためには、最初に **fdisk** を使用して作成したパーティションの ID を **0x8E** に設定します。

```
# /sbin/fdisk /dev/sdc
コマンド (m でヘルプ): p

Disk /dev/sdc: 1073 MB, 1073741824 bytes
64 heads, 32 sectors/track, 1024 cylinders
Units = シリンダ数 of 2048 * 512 = 1048576 bytes

 デバイス ブート   始点     終点  ブロック   ID  システム
/dev/sdc1       1      96    98288   83  Linux

コマンド (m でヘルプ): t
Selected partition 1
16 進数コード (L コマンドでコードリスト表示): 8e
領域のシステムタイプを 1 から 8e (Linux LVM) に変更しました

コマンド (m でヘルプ): p

Disk /dev/sdc: 1073 MB, 1073741824 bytes
64 heads, 32 sectors/track, 1024 cylinders
Units = シリンダ数 of 2048 * 512 = 1048576 bytes

 デバイス ブート   始点     終点  ブロック   ID  システム
/dev/sdc1       1     96     98288   8e  Linux LVM

コマンド (m でヘルプ): w
領域テーブルは交換されました！
```

```
ioctl() を呼び出して領域テーブルを再読込みします。
ディスクを同期させます。
```

続いて **pvcreate** でパーティションを物理ボリュームとして初期化します。

```
# /usr/sbin/pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
```

LVM の領域として利用するすべての物理ボリュームに対して初期化を行います。初期化が完了した物理ボリュームは、**pvscan** で確認できます。

```
# /sbin/pvscan
  PV /dev/sdb1       lvm2 [95.76 MB]
  PV /dev/sdc1       lvm2 [95.79 MB]
  Total: 2 [191.55 MB] / in use: 0 [0   ] / in no VG: 2 [191.55 MB]
```

1.6.2　ボリュームグループの作成

すべての物理ボリュームの準備が完了したら、次にそれらの物理ボリュームをもとにボリュームグループを作成します。ボリュームグループは仮想的なディスクに相当すると考えればよいでしょう。

ボリュームグループの作成は **vgcreate** で行います。**vgcreate** にはボリュームグループ名と、そのボリュームグループを構成する物理ボリュームを指定します。以下では、ボリュームグループ名として **Volume00** を、物理ボリュームとして **/dev/sdb1** と **/dev/sdc1** を指定しています。

```
# /usr/sbin/vgcreate Volume00 /dev/sdb1 /dev/sdc1
  Volume group "Volume00" successfully created
```

ボリュームグループを作成するときに、Physical Extent (PE) サイズを指定できます。PE とは LVM でデータを管理する単位で、1 つのボリュームグループは 64K 個の PE を管理できます。デフォルトの PE のサイズは 4MB のため、最大で 256GB のボリュームグループを作成できます。もしそれ以上のサイズのボリュームグルー

プを作成したい場合は、**vgcreate** に **-s** オプションで PE のサイズを指定します。PE のサイズは最小で 1KB で最大値はありません。

作成したボリュームグループの情報は、**vgscan** で確認できます。

```
# /sbin/vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "Volume00" using metadata type lvm2
```

1.6.3 論理ボリュームの作成

作成したボリュームグループの領域を利用して論理ボリュームを作成します。論理ボリュームは通常のパーティションに相当するもので、ボリュームグループ全体を 1 つの論理ボリュームとすることもできますし、複数の論理ボリュームに分割して利用することもできます。

論理ボリュームの作成は **lvcreate** で行います。**lvcreate** には論理ボリュームのサイズ (MB 単位)、論理ボリューム名、論理ボリュームを作成するボリュームグループ名を指定します。

次の例はボリュームグループ **Volume00** 上に、50MB の論理ボリューム **LogVol01** を作成しています。**-L** のパラメーターが論理ボリュームのサイズ、**-n** のパラメーターが論理ボリュームの名前です。最後に付けるのは論理ボリュームを作成するボリュームグループ名です。

```
# /usr/sbin/lvcreate -L 50M -n LogVol01 Volume00
  Rounding up size to full physical extent 52.00 MB
  Logical volume "LogVol01" created
```

作成した論理ボリュームの情報は、**lvdisplay** で確認できます。

```
# /usr/sbin/lvdisplay /dev/Volume00/LogVol01
  --- Logical volume ---
  LV Name                /dev/Volume00/LogVol01
  VG Name                Volume00
  LV UUID                wchfwk-6V2n-wrKb-v8D7-LVdS-JXPB-01uzRb
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                52.00 MB
  Current LE             13
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0
```

1.6.4　論理ボリュームの利用

作成した論理ボリュームを利用するためには、**lvcreate** 時に表示された論理ボリューム用のデバイスファイルを利用します。通常は **/dev/[ボリュームグループ名]/[論理ボリューム名]** にデバイスファイルが作成されます。

通常のファイルシステムとして利用する場合にはファイルシステムを作成してからマウントします。次の例は、論理ボリューム **/dev/Volume00/LogVol01** を ext4 ファイルシステムとして **/hoge** にマウントして利用する例です。

```
# /sbin/mkfs -t ext4 /dev/Volume00/LogVol01
# /bin/mount -t ext4 /dev/Volume00/LogVol01 /hoge
```

RAW デバイスとして利用する場合には、RAW デバイスへのバインドを行ってから、**/dev/raw/raw[n]** のデバイスファイル経由で利用してください。

```
# /bin/raw /dev/raw/raw1 /dev/Volume00/LogVol01
/dev/raw/raw1:  bound to major 253, minor 0
```

システム起動時に自動的にマウントするためには、**/etc/fstab** に LVM の論理ボリュームをマウントするための設定を追加してください。

次の設定は、ext4 ファイルシステムとして作成された論理ボリューム **/dev/Volume00/LogVol01** を **/hoge** ディレクトリーにマウントするための設定です。

```
/dev/Volume00/LogVol01  /hoge           ext4    defaults     1 2
```

1.6.5 スナップショットの取得

LVM には**スナップショット**と呼ばれる機能が備えられています。これは論理ボリュームのデータをスナップショットとして取得した時点で、読み取り専用の別の論理ボリュームとしてコピーしておく機能です。スナップショットは論理ボリューム上のすべてのデータのコピーを行うのではなく、元データへのリンク情報のみを作成するため非常に高速に動作します。スナップショットの取得後に元データが更新された場合、更新される前のデータをスナップショット領域に保存します。そのため通常は元データの領域よりも少ない領域 (一般的には 10%〜20%程度) があればスナップショットとして利用できます。

さらにスナップショットに対しては読み込みしかできないため、データが更新されるといった危険がありません。このためファイルシステムのバックアップを取得する際に、まずスナップショットを取得して、バックアップ操作はスナップショットに対して行うことで、バックアップ実行中のデータ更新も発生しなくなるためバックアップデータの一貫性が保たれます。このようにスナップショット機能はファイルシステムのバックアップを取得する目的に非常に合致します。

スナップショットを取得するためには、ボリュームグループにスナップショットを取得できるための空き領域が必要です。スナップショットに必要な最大サイズはスナップショットの取得対象となる論理ボリュームのサイズですが、通常はそれよりも少ない領域でも問題ありません。スナップショットに必要な領域のサイズは、対象となる論理ボリュームの更新頻度にも依存しています。更新されるデータが多いほどスナップショット用の領域が必要になることを覚えておきましょう。

スナップショットの取得は、**lvcreate** に **--snapshot** オプションを指定して行います。スナップショットにアクセスするためのデバイスファイル名を **-n** オプションで指定します。また、スナップショット領域として利用するサイズを **-L** オプションで指定します。また、**lvcreate** 実行前に、**modprobe** というコマンドを実行し 、dm_snapshot モジュールを読み込む必要があります。

```
# /sbin/modprobe dm_snapshot
# /usr/sbin/lvcreate --snapshot -L 50M -n snap1 /dev/Volume00/LogVol01
  Rounding up size to full physical extent 52.00 MB
  Logical volume "snap1" created
```

1.6.6　ディスクの追加

LVM の利点が特に発揮されるのは、ディスクの追加や削除といった状況が発生したときです。直接パーティションを利用していた場合あるパーティションの容量を拡大したいと思っても、新しいディスクに容量を拡大したパーティションを準備し、すべてのデータをコピーしてから新しいディスクに交換するといった手順が必要となってしまいます。これに対して LVM を利用していた場合、既存のディスクを残したまま新たなディスクを追加してパーティションを拡大できます。

ここでは前述の環境に新たなディスク (**/dev/sdd**) を追加する手順を説明します。

最初に新しいディスクにパーティション (**/dev/sdd1**) を作成して、物理ボリュームを作成します。

```
# /usr/sbin/pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

次に、**vgextend** でこの物理ボリュームを既存のボリュームグループに追加します。

```
# /usr/sbin/vgextend Volume00 /dev/sdd1
  Volume group "Volume00" successfully extended
```

新しい物理ボリュームが追加されたことを **pvscan** で確認します。

```
# /sbin/pvscan
  PV /dev/sdb1   VG Volume00   lvm2 [88.00 MB / 36.00 MB free]
  PV /dev/sdc1   VG Volume00   lvm2 [88.00 MB / 88.00 MB free]
  PV /dev/sdd1   VG Volume00   lvm2 [92.00 MB / 92.00 MB free]
  Total: 3 [268.00 MB] / in use: 3 [268.00 MB] / in no VG: 0 [0   ]
```

次に、**lvextend** で容量を拡大したい論理ボリュームのサイズを拡大します。下記の例では論理ボリューム LogVol01 に 50MB の容量を追加しています。**-L** オプションに指定する単位には、**M** (メガバイト) のほか、**G** (ギガバイト)、**T** (テラバイト) も使えます。また、**+** を指定することを忘れないようにしましょう。

```
# /usr/sbin/lvextend -L +50M /dev/Volume00/LogVol01
  Rounding up size to full physical extent 52.00 MB
  Extending logical volume LogVol01 to 104.00 MB
  Logical volume LogVol01 successfully resized
```

またオプションとしてパーティションを指定することで、拡大する領域を確保するディスクを指定できます。

```
# /usr/sbin/lvextend -L +20M /dev/Volume00/LogVol01 /dev/sdd1
  Extending logical volume LogVol01 to 124.00 MB
  Logical volume LogVol01 successfully resized
```

最後に実際のファイルシステムのサイズを変更する必要があります。ファイルシステムのサイズ変更はファイルシステムの情報を変更するので、作業の際には安全のため論理ボリュームはアンマウントしてから行いましょう。

- **ext4 ファイルシステムの場合**

  ext4 ファイルシステムのサイズ変更は **resize2fs** で行うことができます。論理ボリュームのサイズに合わせたパーティションとするためには特にサイズを指定する必要はありません。作業に先だって、ファイルシステムの整合性をチェックするために **e2fsck** を実行しておきます。

```
# /sbin/e2fsck -f /dev/Volume00/LogVol01
# /sbin/resize2fs -p /dev/Volume00/LogVol01
resize2fs 1.35 (28-Feb-2004)
Resizing the filesystem on /dev/Volume00/LogVol01 to 106496 (1k) blocks.
Begin pass 1 (max = 6)
Extending the inode table     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
The filesystem on /dev/Volume00/LogVol01 is now 106496 blocks long.
```

以上で ext4 ファイルシステムのサイズ拡大は完了です。再度マウントしてから利用してください。

- **RAW デバイスの場合**

  RAW デバイスにはファイルシステムのサイズのような情報はありません。パーティションのサイズがそのまま RAW デバイスで利用可能な最大サイズとなります。したがって、論理ボリュームを拡大すればそのまま拡大したサイズを利用できます。

1.6.7　ディスクの交換

システム運用中には現在利用中のディスクを別のディスクに変更したいことがあります。LVM を使って運用している場合、LVM 用に利用中の物理ボリュームのデータを他の空いている物理ボリュームに移すことができます。したがって、利用中のディスクを交換したい場合にはまず利用中の物理ボリュームのデータを他の物理ボリュームに移動させます。そして未使用状態になったディスクを LVM の管理から切り離します。

物理ボリュームの利用状況は **pvdisplay** で確認できます。PE (Physical Extent) の Allocated PE の項目が利用中のエクステントの数です。この利用中のエクステントを他の物理ボリュームに移すことが最初の作業です。

```
# /usr/sbin/pvdisplay /dev/sdb1
  --- Physical volume ---
  PV Name               /dev/sdb1
  VG Name               Volume00
  PV Size               88.00 MB / not usable 0
  Allocatable           yes (but full)
  PE Size               4.00 Mib
  Total PE              22
  Free PE               0
  Allocated PE          22
  PV UUID               z806zf-ICTk-RZrP-hEVn-I983-11Rm-H8OmCk
```

最初に安全のため利用中の論理ボリュームのファイルシステムをアンマウントします。

次に **pvmove** でその他の空いている物理ボリュームに利用中のエクステントを移動させます。**pvmove** はさまざまなオプションによって移動させる物理ボリュームのエクステントを指定できます。

今回は **/dev/sdb1** に割り当てられているすべてのエクステントを **/dev/sdd1** に移動させます。

```
# /usr/sbin/pvmove /dev/sdb1 /dev/sdd1
  /dev/sdb1: Moved: 86.4%
  /dev/sdb1: Moved: 100.0%
```

この時点でエクステントの移動が完了したので、再度ファイルシステムをマウントして利用できます。

割り当て済みのエクステントがなくなった物理ボリュームはボリュームグループから解放できます。ボリュームグループから解放するために、**vgreduce** で物理ボリュームをボリュームグループから解放します。

```
# /usr/sbin/vgreduce Volume00 /dev/sdb1
  Removed "/dev/sdb1" from volume group "Volume00"
```

以上で、物理ボリュームが解放されたので **pvscan** で状態が **inactive** になっていることを確認してからディスクを取り外します。

```
# /sbin/pvscan
  PV /dev/sdc1   VG Volume00   lvm2 [88.00 MB / 52.00 MB free]
  PV /dev/sdd1   VG Volume00   lvm2 [92.00 MB / 4.00 MB free]
  PV /dev/sdb1                 lvm2 [88.00 MB]
  Total: 3 [268.00 MB] / in use: 2 [180.00 MB] / in no VG: 1 [88.00 MB]
```

1.6.8　ディスクの削除

LVM で利用中のディスクをすべて解放するには次の手順で行います。

最初に論理ボリューム上で利用中のファイルシステムをアンマウントします。

続いて **lvchange** で論理ボリュームの利用を停止し、**lvremove** で論理ボリュームを解放します。論理ボリュームの利用を停止するためには、**lvchange** の **-a** オプションに **n** を指定します。

```
# /usr/sbin/lvchange -a n /dev/Volume00/LogVol01
# /usr/sbin/lvremove /dev/Volume00/LogVol01
  Logical volume "LogVol01" successfully removed
```

ボリュームグループ上で使用しているすべての論理ボリュームを削除すれば、ボリュームグループを削除できるようになります。ボリュームグループの削除は **vgchange** でボリュームグループの利用を停止して、**vgremove** でボリュームグループを解放します。ボリュームグループの利用を停止するためには、**vgchange** の **-a** オプションに **n** を指定します。

```
# /sbin/vgchange -a n Volume00
  0 logical volume(s) in volume group "Volume00" now active
# /usr/sbin/vgremove Volume00
  Volume group "Volume00" successfully removed
```

以上で関連する物理ボリュームも利用停止状態になったので、ディスクを自由に変更することが可能です。通常のパーティションとして利用したい場合には、**fdisk** でパーティションの ID を変更してから利用してください。

1.7 quota の設定

1.7.1 quota とは

quota を制御するツールを用いると、ユーザーごとやグループごとにファイルシステムの使用可能領域を制限できます。制限するのはブロック数と i ノード数です。1 ブロックは 1KB です。i ノードとは、ファイルの情報を格納する領域で通常は 1 ファイルに 1 つ使用されます。ルートパーティション (/) に quota を設定することはできません。quota を設定するファイルシステムは別に用意してください。

なお Asianux Server 4 では、Journaled Quota と呼ばれる方式が利用可能です。以下で説明する方法ではこれを用いて設定を行います。

1.7.2 quota の設定方法

（1）/etc/fstab の修正

まず **/etc/fstab** を編集して **mount** のオプションを加えます。quota の設定を行う対象のファイルシステムの行に以下のようなオプションを書き加えます。

```
 LABEL=/home /home ext4
defaults,usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0 1 2
```

上の例では、**/home** にマウントするファイルシステムにユーザー quota、グループ quota の両方のオプションを指定しています。

usrjquota はユーザー quota を有効にしていることを示すオプションです。パラメーターとして指定されている **aquota.user** は quota データベースファイル名で、quota チェックの対象となるファイルシステムのルートディレクトリーに作成されます。(上の例の場合 **/home/aquota.user** が作成されます。)

grpjquota はグループ quota を有効にするオプションです。**aquota.group** は、ユーザー quota と同様、quota データベースファイル名を表します。

jqfmt は quota のフォーマット指定のためのオプションで、ここでは vfsv0 が指定されています。この項目はユーザー quota またはグループ quota を有効にするに際して必須の項目です。quota のサポートには、**vfsold** (quota バージョン 1)、**vfsv0** (quota バージョン 2)、**xfs** (XFS ファイルシステムでの quota) の 3 種類がありますが、現在指定できるのは vfsv0 のみとなっています。

* Journaled quota を用いない場合は、以下のように **/etc/fstab** を編集します。

```
LABEL=/home /home ext4 defaults,usrquota,grpquota 1 2
```

/etc/fstab を編集したら、該当するファイルシステムをマウントし直します。

```
# /bin/umount /home
# /bin/mount /home
```

ルートパーティション (/) の場合は、以下のコマンドを使用してマウントし直します。

```
# /bin/mount -o remount /
```

（2）quota ファイルの作成

次のコマンドにより、**/etc/fstab** に記述されている quota を設定するファイルシステムを自動的にチェックして、該当するファイルシステムのトップディレクトリーに quota データベースファイル (上の fstab の設定の場合、**aquota.user**、**aquota.group**) を作ります。

```
# /sbin/quotacheck -vaug
quotacheck: Scanning /dev/sda5 [/home] done
quotacheck: Checked 2 directories and 2 files
```

quota ファイルはテキストエディターなどで編集できないので、注意してください。

また、ルートパーティションの場合は **-m** オプションが必要です。

（3）quota ファイルの編集

edquota コマンドで quota ファイルを編集して、各ユーザー、各グループに quota を設定します。次の例では、ユーザー **foo** の設定を行っています。

```
# /usr/sbin/edquota -u foo
```

グループ **asianux** の設定を行いたい場合には、次のコマンドを実行します

```
# /usr/sbin/edquota -g asianux
```

edquota コマンドを実行すると、デフォルトでは **vi** が起動します。(エディターは環境変数 EDITOR で変更できます) 以下は、**edquota** の実行後にエディターに表示される内容です。

```
Disk quotas for user foo (uid 500):
Filesystem blocks soft hard inodes soft hard
/dev/sda5  200    300  500  51    0    0
```

変更するのは、**soft** と **hard** に対応する数値です。

hard は、絶対に超えることのできない最大の制限値です。

soft は、制限時間が設定されている場合に動作する制限値です。ユーザーの使用量が **soft** の値を超えるとユーザーに警告メッセージが出され、猶予期間に入ります。猶予期間中は **hard** 制限値まで使用可能ですが、猶予期間が過ぎると書き込みができなくなります。

blocks の隣にある **soft** と **hard** の値は、それぞれブロック数の制限値、**inodes** の隣にある **soft** と **hard** の値は、ファイル数の制限値です。(厳密に言えば、inode 番号の制限値のことですが、inode 番号は、パーティションごとに 1 から順に振られていくので、事実上ファイル数の制限値ということになります。)

猶予期間の設定は次のコマンドで行います。

```
# /usr/sbin/edquota -t
```

上の場合と同様にエディターが起動するので、設定を変更してください。

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
 Filesystem          Block grace period     Inode grace period
 /dev/sda5                 1days                 1days
```

（4）quota の有効化

上記の設定後にシステムを再起動すれば有効になります。

手動で quota を有効にするには、**quotaon** を使用します。次の例では、**/etc/fstab** に quota の記述がされているすべてのファイルシステムで、ユーザー quota とグループ quota を有効にします。

```
# /sbin/quotaon -vaug
/dev/sda5 [/home]: group quotas turned on
/dev/sda5 [/home]: user quotas turned on
```

無効にするには、次のコマンドを実行します。

```
# /sbin/quotaoff -vaug
/dev/sda5 [/home]: group quotas turned off
/dev/sda5 [/home]: user quotas turned off
```

（5）quota の確認

最後に **quota** コマンドで quota の設定内容を確認します。次の例では、ユーザー foo に対する設定内容を確認しています。

```
# /usr/bin/quota -u foo
Disk quotas for user foo (uid 500):
 Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
 /dev/sda5   312*    300    500    24:00  52     0      0
```

第2章　ネットワーク設定

この章で説明する内容

目的	システムをネットワークに接続する方法について理解する
機能	ネットワーク上の他のシステムとの通信
必要な RPM	initscript —— 基本システムスクリプト net-tools —— ネットワーク設定の基本ツール
設定ファイル	/etc/sysconfig/network /etc/sysconfig/network-scripts/ifcfg-* /etc/hosts /etc/resolv.conf /etc/modprobe.d/xxx.conf /etc/modules.conf
章の流れ	1 ネットワーク設定の概要 2 network サービスと NetworkManager サービス 3 ネットワークの起動と停止 4 ネットワークの設定 5 ネットワークの状況の確認 6 ボンディングインターフェイスの設定 7 ジャンボフレームの設定
関連 URL	**Japanese FAQ Project** `http://linuxjf.sourceforge.jp/JFdocs/INDEX-network.html`

2.1　ネットワーク設定の概要

Linux システムではほとんどの運用ケースにおいて TCP/IP ネットワークに接続します。この章では Linux システムを LAN に接続するときの設定方法、設定内容、また簡単な設定の確認方法などを説明します。

設定手順は接続するネットワーク環境によって異なります。たとえば、DHCP サーバーがあるネットワーク環境ではほとんどのネットワーク情報を自動的に設定できるので、設定作業は非常に簡単です (ただし、Linux システムをサーバーとして運用する場合は IP アドレスやホスト名を固定で設定するケースが一般的です)。

2.2　network サービスと NetworkManager サービス

Asianux Server 4 では 2 つのネットワークサービスがあります。これまでの network サービスに加え 、NetworkManager サービスが Asianux Server 4 で新しく導入され、動的にネットワークを管理、設定することができます。ただし Asianux Server 3 までと同じネットワーク設定ファイルの書式を使用したい場合や、**bonding** デバイス、**VLAN**、**KVM** の **bridge** を使用する場合は従来の network サービスを利用することをおすすめいたします。

NetworkManager を使用せず、network サービスを使用するようにするには次の手順で行います。

/etc/sysconfig/network-scripts/ifcfg-eth* のファイルを次のように編集します。

```
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="00:0C:29:46:7E:A2"
IPV6INIT="no"
NM_CONTROLLED="no"  # <=  "yes" から "no"に変更
ONBOOT="yes"
(…)
```

次に NetworkManager のサービスが動作しているか確認します。**service NetworkManager status** コマンドで確認します。

```
#/sbin/service NetworkManger status
NetworkManager is stopped
```

“NetworkManager is stopped” と出力されている場合はサービスが停止しています。

```
NetworkManager (pid  <プロセス ID>) を実行中...
```

“NetworkManager (pid <プロセス ID>) を実行中...” と出力される場合は次のコマンドで NetworkManager サービスを停止します。

```
#/sbin/service NetworkManager stop
Stopping NetworkManager daemon:    [ OK ]
```

次に NetworkManager サービスがシステム起動時に自動起動しないようにします。まず **chkconfig** コマンドで現在の設定を確認します。

```
# /sbin/chkconfig --list NetworkManager
NetworkManager      0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

この場合ランレベル 2 から 5 で NetworkManager が自動起動するようになっています。次のコマンドで自動起動を停止します。

```
# /sbin/chkconfig –level <ランレベル> NetworkManager off
```

今度は network サービスがシステム起動時に自動起動するようにします。まず **chkconfig** コマンドで現在の設定を確認します。

```
# /sbin/chkconfig --list network
network       0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

この場合すべてのランレベルで network サービスが自動起動しない設定になっています。次のコマンドで自動起動を設定します。

```
# /sbin/chkconfig –level <ランレベル> network on
```

下のコマンドを実行してネットワークを再起動し、設定をシステムに反映させます。

```
# /sbin/service network restart
```

2.3　ネットワークの起動と停止

ネットワークの起動スクリプトは **/etc/rc.d/init.d/network** です。通常はシステムの起動と同時に実行されますが、このスクリプトは以下のように手動で実行することも可能です。

・ネットワークを起動するには、次のコマンドを実行します。

```
# /sbin/service network start
```

・ネットワークを停止するには、次のコマンドを実行します。

```
# /sbin/service network stop
```

・ネットワークを再起動するには、次のコマンドを実行します。

```
# /sbin/service network restart
```

・ネットワークの現在の状況を確認するには、次のコマンドを実行します。

```
# /sbin/service network status
設定されたデバイス:
lo eth0
現在活動中のデバイス:
lo eth0
```

一般にネットワークの設定を修正した場合には、他のサービスとの関連を考慮してシステムを再起動したほうがいいでしょう。

GUI 環境でネットワークの接続/切断を手動で行う場合には **NetworkManager** を用います。GNOME の場合画面右上に NetworkManager アプレットがありますので、ここから操作を行います。

コンピュータをネットワークに接続するには NetworkManager アプレットを左クリックし、接続したいネットワークインターフェイスを選択します。一般的には **System eth0** であることが多いでしょう。

ネットワークから切断するには NetworkManager アプレットを左クリックし、**[切断する]** をクリックします。

ネットワークの現在の状況を確認するには、次のコマンドを実行します。

```
# service network status
設定されたデバイス:
lo eth0
現在活動中のデバイス:
lo eth0
```

システム起動時に自動的にネットワークに接続するためには、NetworkManager から自動接続を行う設定にする (2.4.1参照) か、**/etc/sysconfig/network-scripts/ifcfg-eth0** の **ONBOOT** を **yes** に変更（2.4.2（2）参照) してください (設定後要再起動)。

2.4　ネットワークの設定

2.4.1　設定方法

通常は Asianux Server 4 のネットワークに関する設定がシステムのインストール時に行われます (こちらについては、本製品に同梱されている Asianux Server 4 インストレーションガイド のインストール手順を参照してください)。

ここではシステムをインストールした後にネットワークの設定を変更する方法について説明します。

CUI の場合は **system-config-network-tui** コマンドを用います。

```
# /usr/sbin/system-config-network-tui
```

次の画面が表示されましたら、**[デバイス設定]** を選択します。

次にデバイスを選択します。eth0 を選択する場合は **[eth0]** を、新規のデバイスを追加する場合は **[新規のデバイス]** を選択します。ここでは eth0 を選択しネットワークカードの設定を行います。

ネットワーク設定の画面が表示されます。

DHCP を使用する場合には **[DHCP を使用]** のチェックを入れてください。手動で固定 IP アドレスを割り当てる場合はチェックをはずします。

手動で固定 IP アドレスを割り当てる場合は次の情報を順に入力してください。

(1) 静的 IP ... マシンに割り当てる固定 IP アドレス

(2) ネットマスク

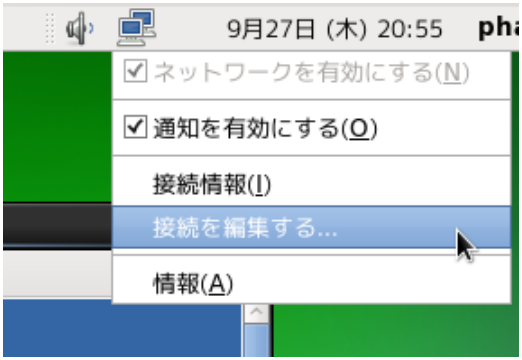
(3) ゲートウェイマシンの IP アドレス

(4) 1 番目の DNS サーバーの IP アドレス

(5) 2 番目の DNS サーバーの IP アドレス

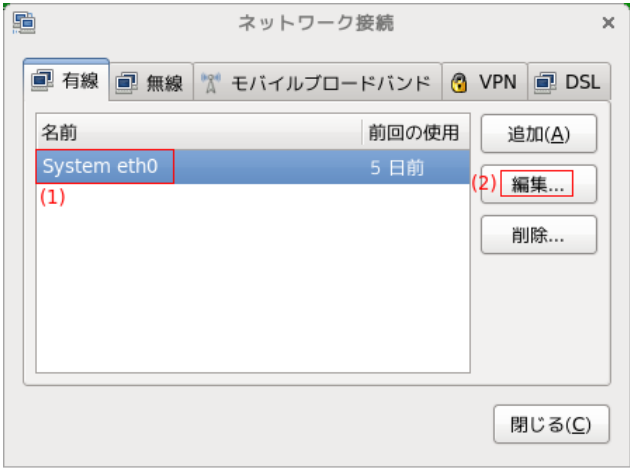
入力が終わりましたら **[OK]** ボタンを押し、**[保存して終了]** で設定を終了します。

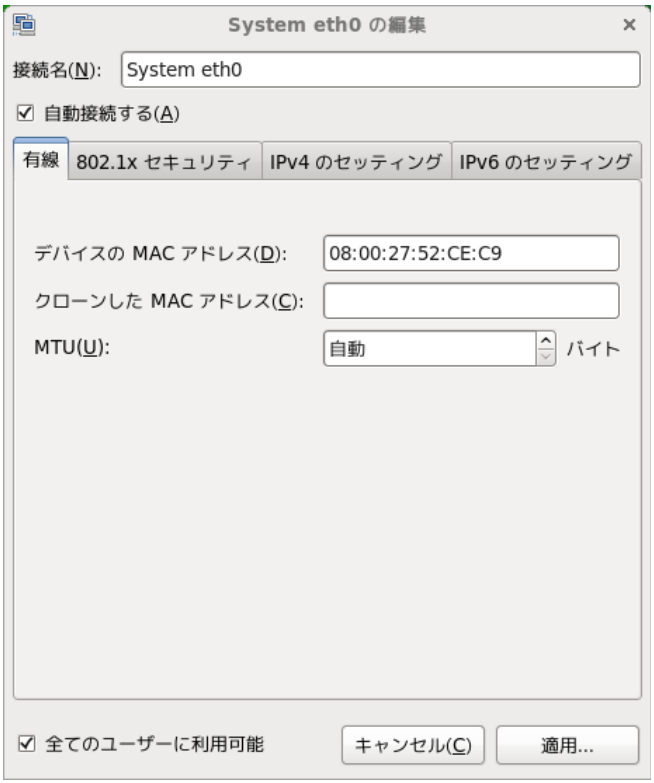
GNOME 環境の場合は **NetworkManager** が利用可能です。画面右上のネットワークアプレットか、**[システム]** → **[設定]** → **[ネットワーク接続]** から利用できます。



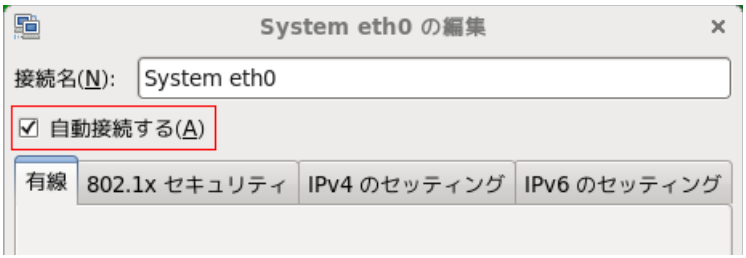
メニューバーの アイコンを右クリックし接続の編集をクリックすると、以下のようなウィンドウが現れますので編集したい接続 (この例では eth0) を選択し **[編集]** ボタンをクリックします。

以下のような設定画面が現れます。

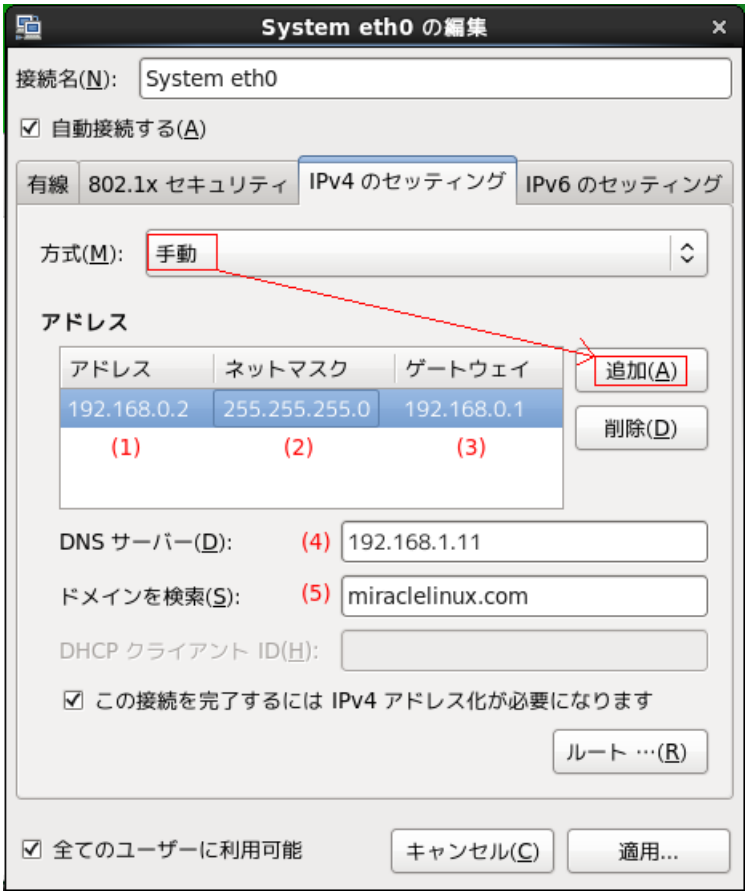




起動時にネットワークに自動接続する場合には **[自動接続する]** のチェックボックスにチェックを入れ、しない場合には外します。



IPv4 のセッティングタブでは接続を DHCP を用いて行うか否かの設定が可能です。



DHCP を用いる場合は **[方式 (M)]** リストボックスを **[自動 (DHCP)]** にしてください。

DHCP を用いず固定 IP アドレスを設定する場合には、**[方式(M)]** リストボックスを **[手動]** に変更し、**アドレス**の右側にある **[追加]** ボタンを押して各項目を設定します。

スクリーンショット中にある設定項目には、それぞれ以下の項目を記述します。

(1) IP アドレス

(2) ネットマスク

(3) ゲートウェイマシンの IP アドレス

(4) DNS の IP アドレス

(5) ホスト名探索のために使用するドメイン名

設定を変更した場合には [OK] ボタンを押し、システムを再起動して設定内容を反映してください。

2.4.2　設定ファイル

ネットワークの設定に関連するファイルには次のようなものがあります。

GUI の場合と同様ネットワークの設定を変更した場合には、他のサービスとの関連を考慮してシステムを再起動した方が良いでしょう。

（1）/etc/sysconfig/network

このファイルには接続するネットワークに関する定義を記述します。設定内容の例を次に示します。

```
NETWORKING=yes
HOSTNAME=localhost.localdomain
GATEWAY=192.168.0.1
```

各変数の意味は次のとおりです。

- **NETWORKING** —— ネットワークを使用するかどうか (**yes** / **no**)
- **HOSTNAME** —— このシステムのホスト名
- **GATEWAY** —— ゲートウェイマシンの IP アドレス

（2）/etc/sysconfig/network-scripts/ifcfg-eth0

このファイルにはそのシステムのネットワークインターフェイスに関する定義を記述します。**eth0** は 1 つ目のネットワークインターフェイスを指します。2 枚のイーサネットカードが装着されている場合には、2 枚目のネットワークインターフェイスは **eth1** になります。このファイルの設定内容の例を次に示します。

```
DEVICE="eth0"
BOOTPROTO=none
TYPE="Ethernet"
UUID="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
HWADDR=xx:xx:xx:xx:xx:xx
IPADDR=192.168.0.3
PREFIX=24
GATEWAY=192.168.0.1
DNS1=192.168.0.1
DOMAIN=miraclelinux.com
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="system eth0"
```

各変数の意味は以下のとおりです。

- **DEVICE** —— ネットワークインターフェイス名
- **BOOTPROTO** —— IP アドレスの割り当て方の設定。**bootp**、**dhcp**、**none** を記述可能
- **NM_CONTROLLED** —— NetworkManager サービスでこのデバイスを設定するかどうか (**yes** / **no**)
- **ONBOOT** —— 起動時にネットワークインターフェイスを有効にするかどうか (**yes** / **no**)
- **TYPE** —— インターフェイスのデバイスタイプ
- **UUID** —— インターフェイスの UUID
- **HWADDR** —— インターフェイスの MAC アドレス
- **IPADDR** —— そのシステムの IP アドレス
- **PREFIX** —— プレフィックス (ネットマスク)
- **GATEWAY** —— ゲートウェイマシンの IP アドレス
- **DNS1** —— 一番目の DNS サーバー
- **DOMAIN** —— ネットワークのドメイン名
- **DEFROUTE** —— デフォルトで利用するインターフェイス
- **IPV4_FAILUER_FATAL** —— IPv4 の設定で失敗した場合 IPv6 の設定を行う
- **IPV6INIT** —— IPv6 を使用するかどうか (**yes** / **no**)
- **NAME** —— NetworkManager で使用される名前

（3）/etc/hosts

このファイルにはネットワーク内のシステムの IP アドレスとホスト名の対応を記述します。このファイルの設定内容の例を次に示します。

1 つ目の値が IP アドレス、2 つ目の値がホスト名、3 つ目以降はそのホスト名のエイリアスです。

```
192.168.0.197 host2 host2.your.domain.name
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1       localhost localhost.localdomain localhost6 localhost6.localdomain6
```

（4）/etc/resolv.conf

このファイルにはホスト名から IP アドレスを調べるために利用するネームサーバーの IP アドレスや、ホストを探索するためのドメイン名などを記述します。このファイルの設定内容の例を次に示します。

```
domain your.domain.name
search your.domain.name
nameserver 192.168.1.11
```

domain 行には接続しているネットワークのローカルドメイン名を、**search** 行にはホスト名を調べるために使うドメイン名を記述します。ネームサーバーが複数ある時には、**nameserver** 行を 3 つまで記述できます。

2.5　ネットワークの状況の確認

ここではネットワークの設定や状況の確認を行うためのコマンドをいくつか紹介します。

2.5.1　ifconfig

このコマンドはネットワークインターフェイスの起動、設定内容の確認などで使用されます。このコマンドを実行すると以下のようにすべてのネットワークインターフェイスの設定内容、状態を確認できます。

```
# ifconfig
eth0  Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
      inet addr:xxx.xxx.xxx.xxx Bcast:xxx.xxx.255.255 Mask:255.255.0.0.
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:19910 errors:0 dropped:0 overruns:1 frame:0
      TX packets:819 errors:0 dropped:0 overruns:0 carrier:1
      collisions:0 txqueuelen:100
      Interrupt:11 Base address:0xdc00
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0.
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:10 errors:0 dropped:0 overruns:0 frame:0
      TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
```

2.5.2　netstat

このコマンドはネットワークシステムに関する多くの情報を表示できます。実行例を次に示します。

-er オプションは IP 経路テーブルを表示します。

```
# /bin/netstat -er
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.2.0.0        *               255.255.255.0   U     0      0        0 eth0
default         xxx.xxx.xxx.xxx 0.0.0.0         UG    0      0        0 eth0
```

-ei オプションはネットワークインターフェイスの設定を **ifconfig** と同様に表示します。

```
# netstat -ei
eth0  Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
      inet addr:xxx.xxx.xxx.xxx Bcast:xxx.xxx.xxx.255 Mask:255.255.255.0.
      UP BROADCLTICAST MTU:1500 Metric:1
      RX packets:19910 errors:0 dropped:0 overruns:1 frame:0
      TX packets:819 errors:0 dropped:0 overruns:0 carrier:1
      collisions:0 txqueuelen:100
      Interrupt:11 Base address:0xdc00
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0.
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:10 errors:0 dropped:0 overruns:0 frame:0
      TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
```

2.5.3 ping

このコマンドはネットワーク上のホストへパケットを送信し、通信が行われていることを確認するものです。正常に接続されているときにはパケット通信の状況を標準出力に表示します。引数にはホスト名を指定することもできます。

実行例を次に示します。

```
# ping 10.2.240.10
PING 10.2.240.10 (10.2.240.10) 56(84) bytes of data.
64 bytes from 10.2.240.10: icmp_req=1 ttl=63 time=0.438 ms
64 bytes from 10.2.240.10: icmp_req=2 ttl=63 time=0.298 ms
64 bytes from 10.2.240.10: icmp_req=3 ttl=63 time=0.324 ms
64 bytes from 10.2.240.10: icmp_req=4 ttl=63 time=0.374 ms
64 bytes from 10.2.240.10: icmp_req=5 ttl=63 time=0.333 ms
64 bytes from 10.2.240.10: icmp_req=6 ttl=63 time=0.327 ms
64 bytes from 10.2.240.10: icmp_req=7 ttl=63 time=0.298 ms
64 bytes from 10.2.240.10: icmp_req=8 ttl=63 time=0.322 ms
64 bytes from 10.2.240.10: icmp_req=9 ttl=63 time=0.333 ms
64 bytes from 10.2.240.10: icmp_req=10 ttl=63 time=0.312 ms
64 bytes from 10.2.240.10: icmp_req=11 ttl=63 time=0.306 ms
^C
--- 10.2.240.10 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 9997ms
rtt min/avg/max/mdev = 0.298/0.333/0.438/0.040 ms
```

2.6　ボンディングインターフェイスの設定

ネットワークの冗長化を行う方法に、ボンディングインターフェイスを利用する方法があります。ここではそのボンディングインターフェイスを使用するアクティブ－バックアップ構成のネットワーク設定について紹介します。この節では、2 つのネットワークインターフェイスをボンディング化する方法を説明しますが、3 枚以上での構成も可能です。

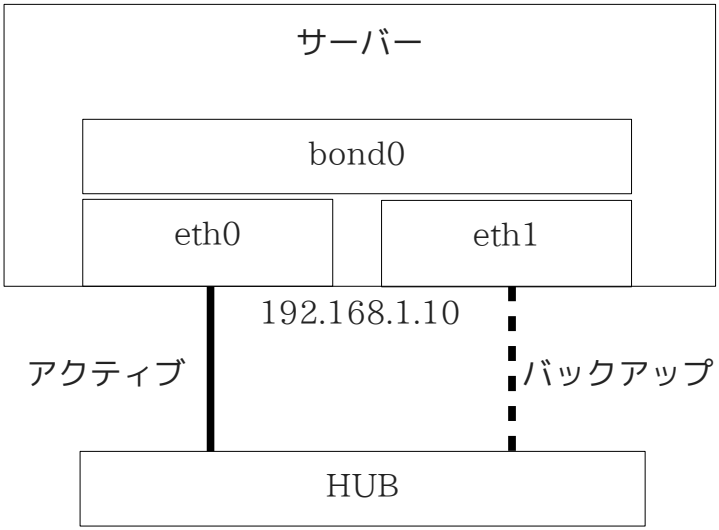


図 *2-1*　ネットワークの構成例

2.6.1　設定ファイル

（1）/etc/modprobe.d/xxx.conf

/etc/modprobe.d/ ディレクトリー以下に拡張子が .conf の適当な名前のファイル (ここでは bonding.conf とします) を作成し、そのファイルに bonding ドライバーを自動的にロードするための記述を追加します。

```
alias netdev-bond0 bonding
options bond0 miimon=100 mode=active-backup
```

各パラメーターの意味は次のとおりです。

- **bond0** ——ボンディングインターフェイス名
- **miimon** —— MII リンク監視を行う間隔（ミリ秒単位） MII リンク監視によって、NIC が正常に動作しているか否かを確認することができます。
- **mode** —— ボンディングモード。値には以下のものがあります。**mode=active-backup** のように記述しても、**mode=1** のように数値で記述しても構いません。

  0. balance-rr - ラウンドロビン
  1. active-backup - アクティブバックアップ
  2. balance-xor - バランス XOR
  3. broadcast - ブロードキャスト
  4. 802.3ad - ダイナミックリンクアグリゲーション
  5. balance-tlb - アダプティブトランスミットロードバランシング
  6. balance-alb - アダプティブロードバランシング

（2）/etc/sysconfig/network-scripts/ifcfg-＊

bond0 を含むネットワークインターフェイスの設定を行います。

・bond0 の設定 (ifcfg-bond0)

/etc/sysconfig/network-scripts ディレクトリー以下に **ifcfg-bond0** を作成し、以下の内容を書き込みます。

```
DEVICE="bond0"
BOOTPROTO=none
IPADDR=192.168.1.10
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
USERCTL=no
```

パラメーターの意味は次のとおりです。

- **USERCTL** ——root ユーザー以外によるデバイス制御を許可するかどうか（**yes** / **no**）

・eth0 の設定 (ifcfg-eth0)

```
DEVICE="eth0"
BOOTPROTO=none
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED=no
```

各パラメーターの意味は次のとおりです。

- **MASTER** ——結合されるボンディングインターフェイス名
- **SLAVE** —— ボンディングインターフェイスで制御されるかどうか (**yes** / **no**)
- **NM_CONTROLLED** —— NetworkManager サービスを使用するかどうか (**yes** / **no**)

  NM_CONTROLLED の行が存在しない場合は追加してください。

・eth1 の設定 (ifcfg-eth1)

```
DEVICE="eth1"
BOOTPROTO=none
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED=no
```

Asianux Server 4 の NetworkManager は bonding デバイスに対応していないため NetworkManager サービスではなく network サービスで bonding の管理を行うことをおすすめいたします。 NetworkManager サービスを使用しない方法は 2.2 network サービスと NetworkManager サービスをご覧ください。

2.6.2　設定確認

ボンディングインターフェイスの動作状況は **/proc/net/bonding/bond*** で確認することができます。

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

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
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: 00:40:26:97:15:ab

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:90:27:3c:82:ff
```

ネットワークインターフェイスの動作状況は **ifconfig** コマンドで確認することができます。

```
# /sbin/ifconfig
bond0   Link encap:Ethernet  HWaddr 00:0C:29:01:65:4B
        inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
        inet6 addr: fe80::240:26ff:fe97:15ab/64 Scope:Link
        UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
        RX packets:23698 errors:0 dropped:0 overruns:0 frame:0
        TX packets:31143 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:2667097 (2.5 MiB)  TX bytes:3996717 (3.8 MiB)
```

```
eth0   Link encap:Ethernet  HWaddr 00:0C:29:01:65:4B
       inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
       inet6 addr: fe80::240:26ff:fe97:15ab/64 Scope:Link
       UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
       RX packets:23699 errors:0 dropped:0 overruns:0 frame:0
       TX packets:31149 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:2667525 (2.5 MiB)  TX bytes:3997925 (3.8 MiB)
       Interrupt:10 Base address:0x1080
eth1   Link encap:Ethernet  HWaddr 00:0C:29:01:65:4B
       inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
       inet6 addr: fe80::240:26ff:fe97:15ab/64 Scope:Link
       UP BROADCAST RUNNING NOARP SLAVE MULTICAST  MTU:1500  Metric:1
       RX packets:8 errors:0 dropped:0 overruns:0 frame:0
       TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:480 (480.0 b)  TX bytes:210 (210.0 b)
       Interrupt:9 Base address:0x1400
```

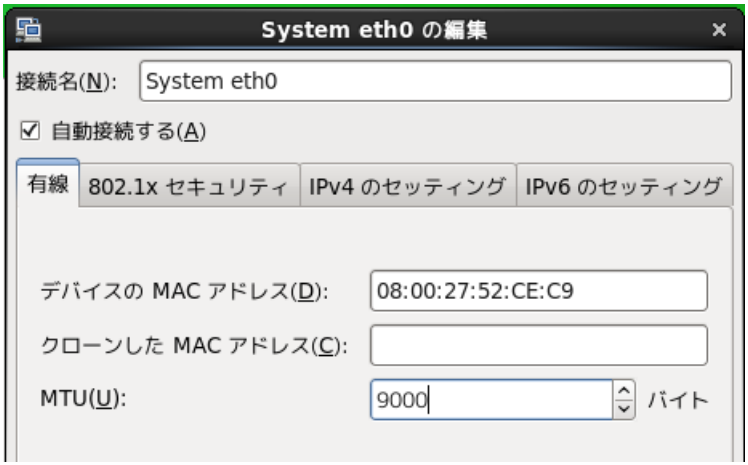
2.7　ジャンボフレームの設定

従来、イーサネットのデータ転送で１度に転送できるフレームサイズは1518バイトと定められていましたが、100Mbpsや1Gbpsのイーサネット規格が普及し1518バイトでは転送効率が悪くなりました。そこで、１度に転送できるフレームサイズを拡張したのがジャンボフレームです。

ジャンボフレームの設定を行う前に、**/etc/sysconfig/network-scripts/ifcfg-eth0** (eth0 ではない場合は他のネットワークインターフェイス名) をエディターで開き、次の１行を追加もしくは編集します。

```
mtu=9000
```

もしくはNetworkManager の接続編集画面 (2.4.1 設定方法 参照) を開き、**有線**タブにある**MTU(U)**の値を 9000 にします。



MTU とは **Maximum Transmission Unit** の略で、上記の例ではこれを 9000 バイトに設定しています。デバイスによって設定上限値が異なるため事前に調べておくと良いでしょう。なおイーサカードが MTU 拡張に対応していない場合、ジャンボフレームの設定は不可能ですので注意してください。

設定した内容を反映させるには、network サービスを再起動します。

```
# /sbin/service network restart
```

一時的にフレームサイズの変更を行う場合は **ifconfig** コマンドを使用します。この方法で設定した場合、再起動すると設定が変更前に戻ります。

```
# /sbin/ifconfig eth0 mtu 9000
```

索引

--snapshot....41
-L....39
/bin/mknod....7
/bin/raw....27
/boot....12
/dev....4
/dev/cdrom....6
/dev/floppy....6
/dev/md*....32
/dev/raw/raw1....28
/dev/raw/raw4....28
/dev/scd0....5
/dev/sda....15
/dev/sdc....19
/dev/sdd4....27
/dev/sr0....6
/dev/Volume00/LogVol01....40
/etc....9
/etc/fstab....24, 34
/etc/hosts....65
/etc/libvirt/storage/default.xml....13
/etc/mdadm.conf....31
/etc/modprobe.d/....69
/etc/rc.d/init.d/network....54
/etc/resolv.conf....65
/etc/sysconfig/network....63
/etc/sysconfig/network-scripts/ifcfg-*....70
/etc/sysconfig/network-scripts/ifcfg-eth0....63
/etc/udev/rules.d/60-raw.rules....28
/home....12
/mnt....23
/opt....14
/proc/mdstat....32
/proc/net/bonding/bond*....72
/proc/net/bonding/bond0....72
/sbin/chkconfig....53
/sbin/e2fsck....43
/sbin/fdisk....31, 37
/sbin/mdadm -A....33
/sbin/mdadm -S....33
/sbin/pvscan....38, 42, 45
/sbin/resize2fs....43
/sbin/service....53
/sbin/vgchange....46
/sbin/vgcreate....38
/sbin/vgscan....39
/tmp....13
/usr....12
/usr/local....12
/usr/sbin/lvchange....46
/usr/sbin/lvcreate....39
/usr/sbin/lvdisplay....40
/usr/sbin/lvextend....43
/usr/sbin/lvremove....46
/usr/sbin/pvcreate....42
/usr/sbin/pvdisplay....44
/usr/sbin/pvmove....45
/usr/sbin/system-config-network-tui....57
/usr/sbin/vgextend....42
/usr/sbin/vgreduce....45
/usr/sbin/vgremove....46
/var....13
/var/cache....13
/var/lib/libvirt/images....13
/var/tmp....13
802.3ad....70

A

active-backup....70
Allocated PE....44

B

balance-alb....70
balance-rr....70
balance-tlb....70
balance-xor....70
blkid....26
bond0....70
bonding デバイス....52
bootp....64
bridge....52
broadcast....70

C

CD/DVD-ROM デバイス....6

D

DEFROUTE....64
dev/sdd1....27
DEVICE....52, 64, 70, 71
dhcp....64
DHCP....52
dm_snapshot....41
DNS サーバー....59
DNS1....64
domain....65
DOMAIN....64

E

e2fsck....43
e2fsprogs....22
e2label....24
edquota....49
eth0....63
eth1....71
ext2....22
ext3....22
ext4....22
ext4....22

F

FAT....17
fdisk....4, 15, 30, 37
FHS....11
Fibre Channel....5
fsck....22

G

GATEWAY....63, 64
grpjquota....47

H

hosts....65
HWADDR....64

I

IDE デバイス....6
ifcfg-bond0....70
ifcfg-eth0....71
ifcfg-eth1....71
ifconfig....66
IP アドレス....64
IPADDR....64
IPV4_FAILURE_FATAL....64
IPV6INIT....52, 64
i ノード....47

J

Journaled Quota....47
jqfmt....48

K

kpartx....19
KVM....52

L

LABEL....25
LAN....52
logical volume....36
Logical Volume Manager....36
lvchange....46
lvcreate....39, 41
lvdisplay....39
lvextend....43
LVM....36, 42
lvm2....36
lvremove....46

M

major/minor 番号....4
major 番号....4
MASTER....71
Maximum Transmission Unit....74
MBR....7
mdadm....32, 34
mdstat....32

miimon....70
minor 番号....4
mkfs....4, 23
mkfs 23
mknod....7
mkpart....20
mode....70
modprobe....6
mount....4, 23, 24, 47, 48
MS-DOS 領域....7
MTU....74

N

NAME....63, 64
nameserver....65
netstat....66
network サービス....52
NETWORKING....63
NetworkManager....60
NetworkManager サービス....52
NM_CONTROLLED....52, 71
none....64

O

ONBOOT....52, 56, 70, 71

P

parted....19
partprobe....19
PE....38, 44
Physical Extent....38
physical volume....36
ping....67
PREFIX....64
pvcreate....38
pvdisplay....44
pvmove....44
pvscan....38, 42, 45

Q

quota....47, 50
quotacheck....48
quotaoff....50
quotaon....50

R

RAID....29
RAID コントローラー....29
RAID-0....29
RAID-1....29
RAID-5....30
raw....27
RAW デバイス....44
RAW デバイス....27
resize2fs....21, 43

S

SAN....4
SCSI デバイス....5
search....65
SLAVE....71
sr0....5
swap....11
System eth0....55
system-config-network-tui....57

T

TCP/IP 52
tune2fs 23
TYPE....64

U

umount....48
USB....5
USERCTL....70
usrjquota....47
UUID....24, 64
uuidgen....25

V

vgchange....46
vgcreate....38
vgextend....42

vgreduce....45
vgremove....46
vgscan....39
VLAN....52
volume group....36
Volume00 38

あ

アクティブ−バックアップ構成....69
アンマウント....44
イーサネット....63
インストール....12, 14
エイリアス....65
エクステント....44
空き領域....9

か

キャラクターデバイス....4
ゲートウェイ....63, 64
拡張パーティション....7
基本パーティション....7
固定 IP アドレス....59

さ

サイズ変更....43
ジャーナリングファイルシステム....22
ジャンボフレーム....74
ストライピング....29
ストレージ....4
スナップショット....41
スワップ....11
ソフトウェア RAID....29
障害....9
冗長化....69
生存期間....10, 13

た

ディスクアレイ....29
ディスクの交換....44
ディスクの削除....46
ディスクの追加....42
ディスク管理....3
データベース....27
データベースサーバー....14
デバイスファイル....4, 23
トラブル....9

な

ネームサーバー....65
ネットワーク....51
ネットワークアプレット....60
ネットワークインターフェイス....63
ネットワークカード....58
ネットワークの起動....54
ネットワークの設定....57

は

パーティション....3, 7, 15
パーティションテーブル....7
パーティション分割....9, 11, 14
ハードディスク....4
パケット....67
バックアップ....41
パフォーマンス....29
パリティ情報....30
ファイルシステム....22
ファイルシステムのラベル....24
フラグメンテーション....10
フレームサイズ....74
ブロックデバイス....4
フロッピーデバイス....6
ページキャッシュ....27
ホスト....63
ホスト名....65
ボリュームグループ....36
ボンディングインターフェイス....69
複数 OS の共存....10
物理ボリューム....36

ま

マウント....23, 24
ミラーリング....29

ら

ラベル.....24
リニアモード.....29
ルート.....12
ローカルドメイン.....65
ログ.....13, 22
論理 MS-DOS ドライブ.....7
論理パーティション.....7
論理ボリューム.....36

Asianux Server 4 サーバー構築・運用ガイド
2013 年 10 月 11 日　初版発行
発行　ミラクル・リナックス株式会社