日医標準レセプトソフト利用における

Ubuntu の基礎知識

2012 年度第 1.1.2 版

日本医師会総合政策研究機構

変更履歴

版数	日付	変更種類・内容	備考
1	2010/08/16	初版	
1.0.1	2010/11/16	・p12 「3.2.1 ネットワーク接続設定」 /etc/network/interfaces ファイルの記述例にある broadcast address を 192.168.1.255 に変更	
1.0.9	2011/07/22	・8.04(hardy helon)の記述を 10.04(lucid lynx)に変更 ・2.3 構成ソフトウェア数を lucid の数値に変更 ・2.3.1 表 1 に 11.04, 11.10, 12.04 の項目を追記 ・2.3.5 表 3 に 10.10, 11.04 の項目を追記 ・2.3 パッケージ名 openssh-server を ssh に変更 ・3.2.1 network-manager について追記。 ・3.3.3 ORCA プロジェクトの apt ラインを lucid の記述に変更 ・3.3.4 表 5 の aptitude upgrade を aptitude safe-upgrade に、aptitude dist-upgrade を aptitude full-upgrade に修正 ・3.4.1 パッケージ名 openssh-server を ssh に変更 ・3.6.3 IIIMF の記述を削除し、iBus について追記。kinput2 の記述を削除。 ・3.6.4 atokx の記述を削除、anthy 用辞書パッケージ orca-medic について追記 ・「3.6.6 日本語環境の設定」, 「3.6.7 インプットメソッドの設定ファイル」, 「3.6.8 Canna に追加辞書をインストールする」, 「3.6.9 Canna 使用時の問題の切り分け」を削除 ・「3.6.7 日本語入力方法」を追加。 ・3.6.10 ORCA フォントから Takao フォントに記述を変更 ・3.7.1 xorg.conf の再生成についてを削除 ・3.7.2 「X が起動しない場合の対処方法」を「利用カードが対応していない」に変更 ・PostgreSQL 8.3 の記述を 8.4 に変更 ・3.8.3 PostgreSQL へのアクセス方法の記述を変更。md5 認証を削除して SSL につい	

		て追記。 ・3.8.4 データベースユーザを oruser から orca に変更 ・3.8.5 定期保守の VACCUME を削除して定期バックアップを推奨に変更。 ・3.9.2 ブートローダを grub2 の記述に変更 ・3.9.4 grub2 の画面に変更 ・3.9.5 grub2 の利用方法に記述修正 ・「3.3.4 apt-key コマンド」追記	
1.0.9a	2011/07/25	・3.3.2 「apt コマンド」のタイトルを「apt/aptitude コマンド」に変更 ・3.3.4 apt-key コマンドに apt-key(8) を参照する一文を追記 ・3.4.3 OpenSSH,sshd の用語統一 ・3.4.4 ufw について追記 ・3.7.1 dpkg-reconfigure xserver-xorg のコマンド削除 ・3.8.2 grub の設定項目を太字に変更 ・3.9.1 linux-backports-modules-hardy を linux-backports-modules-*に変更 ・3.9.1 linux-restricted-modules を削除	
1.0.9b	2011/07/25	・全体の誤字脱字修正 ・参考 URL は箇条書きでタイトルを挿入 ・3.7.5 データベースのバックアップとリストア、コマンド修正 ・4.6 パスワード生成コマンドを mkpasswd から pwgen に変更 ・4.8 sudo ログ出力を整形	
1.1.0	2011/07/25	・3.8.3 図番号 2,3 が反転してるのを修正	
1.1.1	2012/07/04	・2.3 対応アーキテクチャを修正 ・2.3.1 Ubuntu のバージョンを追加 ・2.3.2 LTS の説明を変更 ・2.3.3 Launchpad の説明を変更 ・2.3.4 パッケージ種別の説明を変更 ・2.3.5 Ubuntu バージョンによるパッケージの差異を修正 ・2.3.6 サポート期間を修正 ・3.2.1 参照先 URL 変更 ・3.3 言い回しを修正 ・3.3.3 パッケージ取得先 URL を変更 ・3.6.2 言い回しを変更 ・3.6.5 Mozc について追記 ・3.6.6 3.6.7 と順序入れ替え／Takao フォントについて追記 ・3.6.7 3.6.6 と順序入れ替え	

		・3.7.1 X.org のログの記載を削除 ・3.7.2 言い回しを修正 ・3.8.3 PostgreSQL の説明を移動／参照先 URL 変更 ・3.8.4 参照先 URL 変更	
1.1.2	2012/07/26	・2.3 対応アーキテクチャから powerpc を削除 ・2.3.1 コードネームの typo 修正 ・2.3.3 Debian の記載を削除 ・3.3.3 書式を修正	

目次

1. はじめに

このテキストはシステム主任者として日医標準レセプトソフトを運用するために必要となる知識や能力について、日医標準レセプトソフトに特化しない一般的な内容を記しています。日医標準レセプトソフト固有のサービスやアプリケーションについては、別テキストを参照ください。

テキストに掲載されている会社名および製品名は一般に各社の登録商標または商標です。

1.1. このテキストで扱う内容

- Ubuntu の概要
- システム運用/トラブル対処 (日医標準レセプトソフト非固有部分)
- システムセキュリティ (日医標準レセプトソフト非固有部分) の基礎知識

2. Ubuntu について

この章では、日医標準レセプトソフトのプラットフォームとなる OS、Ubuntu の概要を記しています。

2.1. Ubuntu とは

Ubuntu -- http://www.ubuntu.com/

Ubuntu は Debian GNU/Linux をベースにした Linux ディストリビューションです。

Ubuntu は Canonical Ltd.が支援している Ubuntu コミュニティが開発しています。

Linux ディストリビューションとは、Linux カーネルと多くの GNU ツール、オープンソースソフトウェアをまとめたオペレーティングシステムです。

GNU Project -- http://www.gnu.org/

フリーなオペレーティングシステム開発の為のプロジェクトです。GNU プロジェクトにより作られた多くのライブラリ/ツール/プログラムは現在、様々な OS 上で利用されています。代表的なソフトウェアに gcc、 emacs 等があります。

Debian Project -- http://www.debian.org/

フリーなオペレーティングシステム開発の為のボランティア団体です。

Linux -- http://www.kernel.org/

UNIX ライクな OS のカーネルです。一般的には Linux (カーネル)を利用した OS 全体が「Linux」と呼ばれることが多いですが、厳密には Linux カーネルのみを指します。
GNU/Linux とは、多くの GNU 製コンポーネントを含んだ、Linux カーネルを使ったシステム、という意味です。

2.2. 「フリーソフトウェア」とは

フリーソフトウェアとは、フリーなライセンスの元で配付されるソフトウェアです。フリーなライセンスの多くに共通する特長として以下があります。

- （再）配付の自由
- 改変の自由
- 作者名の保持
- ソースコードの公開配付

よく採用されるフリーなライセンスに以下があげられます。

- GPL (http://www.gnu.org/copyleft/gpl.html)
- BSD (http://www.debian.org/misc/bsd.license)
- Artistic (http://www.perl.com/language/misc/Artistic.html)

DFSG(Debian Free Software Guideline)

ソフトウェアライセンスがフリーソフトウェアライセンスであり、Debian の一部として含めることが可能かどうか判定するために Debian プロジェクトが使用しているガイドラインです。

DFSG の日本語訳は以下の URL で参照できます。

http://www.debian.org/social_contract.ja.html#guidelines

Ubuntu ライセンスポリシー

Ubuntu で「フリーソフトウェア」として提供するソフトウェアに含めるかどうか判定するためのガイドラインで、一部例外を除き DFSG にほぼ準拠しています。

Ubuntu ライセンスポリシーは以下の URL で参照できます。

http://www.ubuntu.com/project/about-ubuntu/licensing

2.3. Ubuntu の特徴

フリーなライセンスのソフトウェアから構成

Ubuntu ライセンスポリシー (DFSG にほぼ沿ったポリシー) に準拠したフリーソフトウェアで構成されています。これらは main コンポーネントで提供されます。

その他フリーではないライセンスを利用しているソフトウェアもコンポーネントを指定することで利用することができます。

定期リリースとパッケージバージョン

Debian GNU/Linux の開発版パッケージを利用する事で比較的新しいバージョンのパッケージを利用できます。リリースは半年に一度行います。

複数のアーキテクチャに対応

Ubuntu 12.04 リリース時点では i386、amd64、arm に対応しています。さらに非公式版ではいくつかのアーキテクチャに対応しています。

> i386 と amd64 の違い
> i386 との互換モードのある 64 ビット CPU を対象としたアーキテクチャを歴史的な理由により amd64 と呼びます。amd64 が動作するアーキテクチャでは、i386 版の Ubuntu も動作させることができます。
> i386 版では使用出来るメモリの上限が制限されますが、PAE カーネルを使用することにより 64GB まで使用することが出来ます(1 プロセスのメモリは 4GB まで)。

構成ソフトウェア数が豊富

10.04(lucid lynx)では 34,980 個、12.04(Precise Pangolin)では 44,818 個のパッケージが収録されています。(収録パッケージ数)

パッケージ管理システム

deb 形式のパッケージを dpkg コマンドで管理します。dpkg のフロントエンドとして apt コマンドが利用できます。

Ubuntu は Debian GNU/Linux の派生ディストリビューションであり、deb パッケージ管理システムも利用していますが、Debian GNU/Linux とのパッケージ互換性は保証されていません。

LiveCD

インストール対象 PC にインストール CD を LiveCD として起動し、ハードウェアの動作確認ができます。

2.3.1. Ubuntu のバージョンとコードネーム

Ubuntu は一年に二回リリースされます。バージョン番号はリリースされた西暦下二桁と月からなります。また、二年おきに LTS(Long Term Support:長期サポート版)となるリリースがあります。

それぞれのリリースにはコードネームが付加されます。

表 1 Ubuntu のコードネーム

バージョン番号	コードネーム	リリース日
4.10	Warty Warthog Sounder	2004 年 10 月
5.04	Hoary Hedgehog Array	2005 年 4 月
5.10	Breezy Badger Colony	2005 年 10 月
6.06(LTS)	Dapper Drake Flight	2006 年 6 月
6.10	Edgy Eft Knot	2006 年 10 月
7.04	Feisty Fawn Herd	2007 年 4 月
7.10	Gusty Gibbon Tribe	2007 年 10 月
8.04(LTS)	Hardy Heron	2008 年 4 月
8.10	Intrepid Ibex	2008 年 10 月
9.04	Jaunty Jackalope	2009 年 4 月
9.10	Karmic Koala	2009 年 10 月
10.04(LTS)	Lucid Lynx	2010 年 4 月
10.10	Maverick Meerkat	2010 年 10 月
11.04	Natty Narwhel	2011 年 4 月
11.10	Oneiric Ocelo	2011 年 10 月
12.04(LTS)	Precise Pangolin	2012 年 4 月
12.10	Quantal Quetzal	2012 年 10 月(予定)

2.3.2. LTS: Long Term Support

Ubuntu の LTS は他のリリース版(1 年半)とは違い長期間のサポートが約束されています。

10.04 までは Desktop Edition が 3 年、Server Edition が 5 年のサポート期間でした。

12.04 からはどちらも 5 年のサポート期間になっています。

2.3.3. Ubuntu サポート

Ubuntu は、Canonical Ltd.による有償サポートの他、無償で以下のサポートを受けることができます。

- オフィシャルドキュメントへのアクセス(https://help.ubuntu.com/)
- LaunchPad への問い合わせ(https://answers.launchpad.net/ubuntu: 英語のみ)
- セキュリティアップデートパッケージの提供(http://www.ubuntu.com/usn)
- コミュニティへの問い合わせ(IRC, Web フォーラム, メーリングリスト)

LaunchPad

フリーソフトウェア開発のための Web サイト。Ubuntu 開発元の Canonical Ltd.により開発・運用されています。

https://launchpad.net/

ソースコードホスティング、ナレッジデータベース等の Web アプリケーションが稼働しており、Ubuntu のバグトラッキングシステムとしても利用されます。

USN

Ubuntu Security Notice。Ubuntu のセキュリティ警告文書です。以下の URL で参照できます。

http://www.ubuntu.com/usn

2.3.4. コンポーネント(パッケージ分類)

Ubuntu は Debian GNU/Linux のようにすべてオープンソース、ライセンスフリーなソフトウェアのみで構成することに拘ってはいません。修正できないバイナリファームウェア (NIC やビデオカードなどで利用) 、フリーではないソフトウェアも提供されています。

Ubuntu ではライセンスの種類とコミュニティによるサポート体制によりソフトウェアパッケージをコンポーネントに分類しています。

表 2 Ubuntu のコンポーネント

ライセンス種別	サポート	コンポーネント
フリーソフトウェア	有	Main
フリーソフトウェア	無	Universe
非フリーソフトウェア	有	Restricted
非フリーソフトウェア	無	Multiverse

Main コンポーネントは Ubuntu ライセンスポリシーに準拠するソフトウェアパッケージで構成され、Canonical による公式なサポートが保証されます。

Universe コンポーネントはコミュニティによってメンテナンスされているソフトウェアパッケージで構成され、サポートは保証されていません。

Restricted コンポーネントはフリーではないソフトウェアで構成されますが、Ubuntu を利用するために必要なソフトウェアが含まれます。サポートはありますが、フリーソフトではないため、Main のような完全なサポートは受けられません。

Multiverse コンポーネントはフリーではなく、Restricted で代替の効く重要ではないと思われるソフトウェアで構成されます。サポートもありません。

2.3.5. 8.04(Hardy Heron)と 12.04(Precise Pangolin)の差異

Ubuntu 8.04 (hardy heron) から 12.04 (Precise Pangolin) までの主な変更点を表 3 にまとめます。

表 3 最近の各バージョンでの主な変更点

8.04	・GNOME 2.22 ・Linux 2.6.24 ・X.org 7.3 ・PostgreSQL 8.2.7/8.3.11 ・SELinux/AppArmor サポート
8.10	・GNOME 2.24 ・X.org 7.4 ・Linux 2.6.27 ・DKMS(Dell Kernel Module Support)サポート ・service コマンドサポート
9.04	・GNOME 2.26 ・X.org 7.6 ・Linux 2.6.28 ・ext4 ファイルシステムサポート
9.10	・GNOME 2.28 ・Linux 2.6.31 ・hal 廃止 ・Grub2 デフォルトブートローダ化 ・iBus をデフォルトインプットメソッドに採用, upstart サポート
10.04	・GNOME 2.30 ・Linux 2.6.32 ・PostgreSQL 8.4.4 ・Takao フォント追加 ・rsyslog をデフォルト syslog に採用
10.10	・Linux 2.6.35 ・GNOME 2.32
11.04	・Linux 2.6.38 ・GNOME 3.0 ・PostgreSQL 8.4.8 ・OpenOffice.org から LibreOffice に移行 ・Unity を標準デスクトップに採用
11.10	・Linux 3.0 ・GNOME 3.2 ・multiarch 機能の提供 ・LightDM をログインマネージャに採用
12.04	・Linux 3.2.14 ・GNOME 3.4 ・PostgreSQL 9.1 ・Desktop 版が 5 年サポートに変更

2.3.6. Ubuntu の Server Edition と Desktop Edition の違い

Ubuntu は、すべてのバージョンで Server Edition と Desktop Edition をリリースしています。二つの Edition は同じ apt リポジトリを利用しますが若干の違いがあります。違いは Server Edition では X 環境がデフォルトでは含まれない事、Linux カーネルのオプションが異なる事があります。

X 環境がデフォルトで含まれない、というのは Ubuntu Server Edition のインストーラでは「X Environment」が選択できないということです。インストール後に apt コマンドで直接パッケージ名を指定すれば X 環境を構築することが可能です。

10.04 までは、Desktop Edtion のサポート期間は 3 年、Server Edition は 5 年です。

そのため、ORCA プロジェクトは Ubuntu での利用にサポート期間の長い Server Edition を採用し、必要最小限な Desktop Edition のパッケージを ORCA プロジェクトでサポートすることになっています。

12.04 からは Desktop Edition のサポート期間が 5 年になるため、制限は緩和されます。

3. システムの運用

この章では Ubuntu 運用の際に注意すべき点を説明します。

3.1. Ubuntu のインストール

3.1.1. 最小パッケージ構成のインストール

システムをインストールする際、以下の tasksel 画面でいくつかのセクションにチェックをいれることで、まとめてインストールするパッケージを選択できます。

どのパッケージセクションを入れていいかよくわからない、ハードディスクの容量に余裕があるなどの理由で、とりあえず選択できるパッケージセクションをすべて選んでインストールするのはセキュリティの観点からも推奨できません。

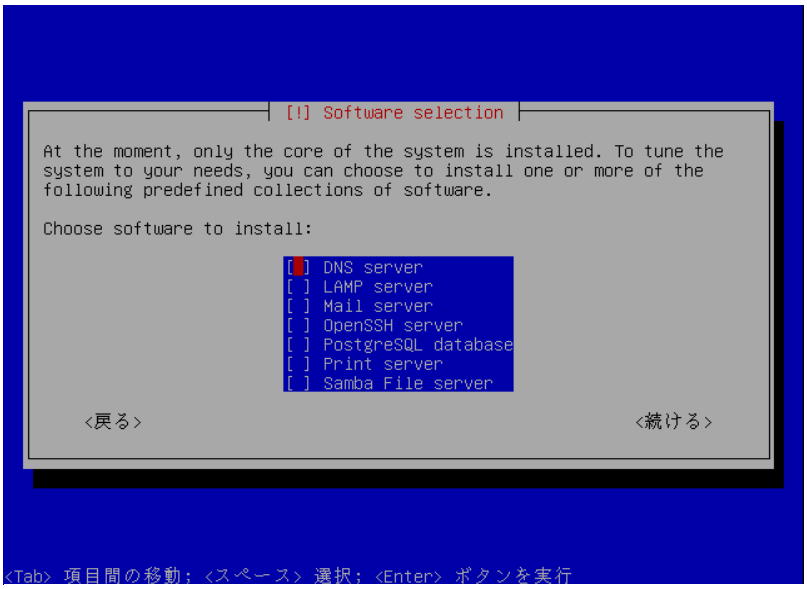

図 1 tasksel 画面

tasksel で全てのセクションにチェックを入れずにインストールを進めると、システムを利用するのに最低限必要なパッケージがインストールされます。ユーザはシステムのインストール終了後から本当に必要なパッケージをインストールするようにしましょう。

Ubuntu は基本パッケージには、リモートアクセスするサービスを提供するパッケージが含まれません。OpenSSH のパッケージも別途インストールする必要があります。

3.1.2. インストール時のトラブル対応

インストール中になんらかの問題が起きた場合、Alt-F<数字> で仮想コンソールに移動し、エラーメッセージやログを見ることができます。

Alt-F2 で移動できる仮想コンソールでは shell を利用して、ls や cat といったコマンドが利用できます。

Alt-F4 ではインストールログを見ることができます。

いずれの操作を行った後も、Alt-F1 でインストール画面に戻ります。

3.2. Ubuntu の構成

3.2.1. ネットワーク接続設定

network-manager を利用して設定する場合は以下の資料を参照してください。
「Ubuntu 10.04LTS Lucid Lynx オンライン請求の設定手順書」に network-manager を利用した設定方法の記述があります。

- http://ftp.orca.med.or.jp/pub/data/receipt/download/lucid/lucid-online-2010-11-11.pdf

Ubuntu で network-manager を利用しない場合のネットワークの設定は以下のファイルに記述します。

/etc/network/interfaces

ネットワークインターフェイスの設定を記述します。

DHCP で動的 IP を取得するか、静的な IP アドレスとデフォルトゲートウェイアドレスを設定するかを指定します。

またオプションとして、各インターフェイスが有効・無効になった時に実行するコマンドを指定することもできます。以下は eth0 を有効にした後に「/usr/local/bin/alert-interface」スクリプトを実行するように指定しています。

```
auto eth0
 iface eth0 inet static
    address 192.168.1.42
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    up /usr/local/bin/alert-interface
```

また、/etc/network/if-up.d や /etc/network/if-down.d 以下に実行権をつけたスクリプトを用意することで、ネットワーク設定が有効・無効になった際にスクリプトを実行する事もできます。

/etc/resolv.conf

名前解決に利用するドメインネームサーバを指定します。以下は設定例です。

```
 search  example.com
 nameserver 192.168.11.1
```

search には、ローカルドメイン以外で検索するドメインを指定します。上記の例では「www」というホスト名を検索した場合、www.example.com の名前解決を行おうとします。nameserver には利用する DNS サーバを指定します。

/etc/nsswitch.conf

名前解決には DNS 以外にもデフォルトで/etc/hosts や mDNS(Multicast DNS)が使用される。mDNS は OS が混在した環境等ネットワーク環境により接続に時間が掛かる

問題を引き起こすことがあります。その場合は、/etc/nsswitch.conf の hosts:エントリーから mdns4 を削除すると改善されます。

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns
```

3.3. パッケージ管理システム

Ubuntu は Debian GNU/Linux と同じ、deb パッケージ管理システムを利用します。

パッケージ管理システムを利用することで以下のような利点があります。

パッケージをソフトウェアコンポーネント単位で追加/削除

「環境」を作成するための単位「コンポーネント」を用意しています。コンポーネントを利用することで環境を簡単に構築できます。

以下は aptitude コマンドを利用して OpenSSH によるリモートメンテナンス環境をインストールするコマンドの例です。

```
$ sudo aptitude install ssh
```

パッケージ間の依存関係/排他関係などの管理

例えば画像表示アプリケーションの eog が、JPEG 画像ファイルを扱うのに libjpeg62 を必要とします。これを eog が libpng62 に依存する関係と言います。

また、libjpeg62 をシステムから削除してしまうと、eog で JPEG 画像を見れなくなってしまいます。これは libjpeg62 が eog に依存されていることに影響します。

パッケージ管理システムでは、このような関係を考慮し、目的のパッケージが依存する他のパッケージをあわせてインストールします。また、パッケージの削除を行う際も依存されているパッケージがあることを警告します。

ネットワークを介したパッケージ取得/インストールが可能

インターネットに接続でき、パッケージリポジトリサーバにアクセスができる環境であれば、システム稼働環境にパッケージファイルをローカルに用意・保存しなくても、パッケージファイルの取得、インストールができます。

Ubuntu ではこれらの機能を dpkg、apt、aptitude コマンドで実現しています。

3.3.1. dpkg コマンド

Debian Package 管理システムの deb パッケージを操作する為のプログラムです。deb パッケージのインストール、アンインストール、ビルドやインストールされているパッケージの管理を行います。dpkg には複雑なコマンドラインパラメータの指定が必要です。ユーザにやさしいフロントエンドとして後述する apt/aptitude があります。

RedHat 系ディストリビューションで利用される rpm(Rpm Package Manager)と dpkg の比較を表 4 に示します。

表 4 dpkg と rpm の代表的オプションの比較

パッケージ操作	dpkg	rpm
説明を表示	dpkg -s パッケージ名	rpm -qi パッケージ名
ファイルの一覧	dpkg -L パッケージ名	rpm -ql パッケージ名
説明表示(ファイル指定)	dpkg -l deb ファイル	rpm -qpi rpm ファイル
ファイルの一覧(ファイル指定)	dpkg -c deb ファイル	rpm -qpl rpm ファイル
インストール済みパッケージ一覧	dpkg -l	rpm -qa
ファイルの持ち主パッケージの検索	dpkg -S ファイル	rpm -qf ファイル名
インストール	dpkg -i|--install deb ファイル	rpm -i rpm ファイル
削除(設定ファイルは残す)	dpkg -r|--remove パッケージ名	rpm -e パッケージ名
削除(設定ファイルも消す)	dpkg -P|--purge パッケージ名	該当オプションなし
パッケージの展開	dpkg -x deb ファイル展開先ディレクトリ	rpm2cpio rpm ファイル | cpio -id

3.3.2. apt/aptitude コマンド

apt は、「Advanced Packaging Tool」の略で、dpkg コマンドのフロントエンドです。パッケージのインストール/削除の際の依存関係の解決を容易にするための仕組みであり、以下のようなツールが提供されています。

apt-get/aptitude

指定されたパッケージのインストールや削除を行ったり、システム全体のアップグレードを行うためのコマンドです。管理者権限で実行する必要があります。

apt-get と aptitude は混在して利用することができます。両者には利用できるオプションが異なるなど違いがあります。

apt-cache/aptitude search/aptitude show

提供されているパッケージの情報を参照するためのコマンドです。指定したパッケージの詳細な情報を表示したり、依存パッケージをリストできます。一般ユーザでも実行可能です。

3.3.3. apt の設定

apt を使用するには、パッケージ提供元をあらかじめ登録しておく必要があります。リポジトリ情報は/etc/apt/sources.list および/etc/apt/sources.list.d/以下に記述します。

書式は以下のとおりです。

```
deb  URI  DISTRIBUTION  [COMPONENT]...
```

URI は Debian パッケージリポジトリの URI を指定します。URI の scheme には http の他に、ftp や ssh、CD-ROM から読み込む cdrom などを指定できます。

DISTRIBUTION は OS のバージョンです。Ubuntu では「hardy」や「lucid」など、Debian GNU/Linux では「squeeze」や「lenny」を指定します。

COMPONENT にはコンポーネントを指定します。Ubuntu では「main」や「restricted」、Debian GNU/Linux では「main」や「contrib」などを指定します。

ORCA プロジェクトが提供するパッケージ群を、apt を使ってインストールできるようにするには「/etc/apt/sources.list.d/jma-receipt-lucid46.list」に次のような記述を追加します。

```
deb http://ftp.orca.med.or.jp/pub/ubuntu lucid4.6 jma
deb http://ftp.orca.med.or.jp/pub/ubuntu lucid-common jma
```

ORCA プロジェクトのパッケージリポジトリでは「jma」というコンポーネント名を提供しています。

この設定ファイルは以下の URL から取得することができます(バージョンには 46 や 45 等最初の 2 桁の「.」を除いたバージョンがはいります)。
http://ftp.orca.med.or.jp/pub/ubuntu/jma-receipt-lucid(バージョン).list

apt を使用してパッケージをインストールする際の基本的な流れは以下のようになります。

1. /etc/apt/sources.list および/etc/apt/sources.list.d/以下に必要な記述を行う
2. apt-get/aptitude update を実行する
3. apt-get/aptitude install package ... を実行する

1 はパッケージの提供元を追加・削除した場合に行ないます。

2 はパッケージ提供元からパッケージ情報をダウンロードします。通常は 1～数日に 1 度実行すれば十分です。ただし、パッケージ提供元の状態とダウンロードされた情報がくいちがってしまうと、正常にパッケージをインストールできなくなる場合があります。

なお、パッケージのインストールに CD-ROM を使用する場合には apt-cdrom を次のように実行することで、/etc/apt/sources.list に適切な記述を追加することができます。また、それと同時に CD-ROM にあるパッケージ情報がシステムに登録されます。

```
$ sudo apt-cdrom add
```

一部のパッケージでは、そのパッケージで提供しているプログラムを使用する上で必要となる設定を、パッケージのインストールと同時に行うことができるようになっています。このようなパッケージの多くで debconf という仕組みが使われています。

debconf の利点は、一度設定した内容を覚えていることです。この機能によって、パッケージを更新するたびに何度も同じ質問にパラメータを探さなければいけない状況を回避することができます。もしも debconf を使用する場合は次のようにパッケージ名を指定してコマンドを実行します。

```
$ sudo dpkg-reconfigure xserver-xorg
```

このコマンドはシステムの状態を変えるものであるため管理者権限で実行する必要があります。

3.3.4. apt-key コマンド

Debian/Ubuntu では、パッケージの改竄を防止するための仕組みが導入されています。あらかじめ GPG の公開鍵を登録してある提供元以外からパッケージを取得しようとすると、apt-get/aptitude update 時に以下のようなエラーが出力されます。

```
W: GPG error: http://ftp.orca.med.or.jp lucid4.5 Release: 公開鍵を利用できないため、以下の署名は検証できませんでした: NO_PUBKEY 137E0B9A69C4E4D0
```

そのため、ORCA プロジェクトからパッケージを取得する前に ORCA プロジェクトの GPG 公開鍵を登録する必要があります。ORCA プロジェクトの Ubuntu 用公開鍵は以下の URL から取得できます。

- http://ftp.orca.med.or.jp/pub/ubuntu/archive.key

この公開鍵を wget でダウンロードしておき、以下のコマンドで登録します。

```
$ sudo apt-key add archive.key
```

apt-key コマンドについて、鍵登録以外の詳細については apt-key(8)を参照してください。

3.3.5. apt の有効活用

apt により、パッケージインストール/アンインストールの際の依存関係を柔軟に処理することができます。代表的なコマンド/オプションを表 5 に挙げます。

表 5 apt/aptitude の代表的な使い方

コマンドとオプション	目的
aptitude update	リポジトリのパッケージ情報をダウンロードする。
aptitude install パッケージ名	指定したパッケージをインストールする。
aptitude remove パッケージ名	指定したパッケージを削除する。
aptitude safe-upgrade	システム全体のアップグレード。
aptitude full-upgrade	パッケージ入れ換えを含むアップグレード。
aptitude clean	取得した deb パッケージを削除する。ダウンロードした deb パッケージは /var/cache/apt/archives 以下に保存される。
apt-cache show パッケージ名	指定したパッケージに関する情報を表示する。
apt-cache showpkg パッケージ名	依存関係など、より詳しい情報を表示する。
apt-cache depends パッケージ名	指定したパッケージのの依存情報を表示する。
aptitude search パターン	指定したパターンを元にパッケージを検索する。

aptitude remove を実行した場合、指定したパッケージに依存するパッケージも削除されることに注意してください。設定ファイルも含めて削除する場合は aptitude purge を指定します。

3.3.6. パッケージ管理の注意点

不要なパッケージは入れない

Ubuntu で用意されている dpkg や aptitude を利用するとアプリケーションの追加や削除は容易になります。

しかし、簡単だからといって不必要なパッケージまでシステムにインストールすることはシステムを守るセキュリティ的視点で見ても望ましいものではありません。

不要なパッケージの発見と削除

deborphan コマンドを使用すると、パッケージ間の依存関係を分析して、どのパッケージからも依存されていないパッケージを見付けることができます。

ただし、他のパッケージから依存されてはいないが、システムで使われている重要なパッケージであるケースもありえます。パッケージの削除には注意が必要です。

deborphan はインストール状態を保持すべきパッケージを登録できるようにもなっているので、使用されていることがわかっているパッケージについてはあらかじめ登録しましょう。

deborphan コマンドは deborphan パッケージに含まれています。

定期的なアップデート

インストールが容易なシステムなのでアップデートも容易に実施できます。

ついつい忘れがちですが、定期的なアップデートを実施しましょう。

アップデートは以下の手順でコマンドを実行します。

```
$ sudo aptitude update
$ sudo aptitude safe-upgrade
```

アップデートされるパッケージをアップデート適用前に確認するには以下のようにシミュレートオプション「-s」をつけて aptitude コマンドを実行します。

```
$ sudo aptitude update
$ sudo aptitude safe-upgrade -s
```

3.4. リモートメンテナンス

稼働するマシンの状況によってはリモートメンテナンスが必要な場合もあります。ここではリモートメンテナンスを SSH を利用して実施する場合の準備、アクセス制御について説明します。

3.4.1. OpenSSH サーバのインストール

Ubuntu 10.04(lucid lynx)ではデフォルトで OpenSSH のサーバパッケージがインストールされません。これは Ubuntu コミュニティによる、システムをデフォルトでできるだけ安全にしておくための配慮です。

OpenSSH を利用するには ssh パッケージをインストールします。以下のコマンドでパッケージをインストールします。

```
$ sudo aptitude install ssh
```

パッケージがインストールされると sshd が稼働します。デフォルトではパスワード認証でのログインが有効になっています。管理者権限を持つユーザのパスワードが簡単な場合は早急に変更しましょう。

3.4.2. アクセス制御(TCP Wrappers)

システムのリモートメンテナンス準備のために sshd を稼働した後、だれでもアクセスできるようにサーバを開放しておくのは非常に危険です。

ネットワークの設定で sshd へのアクセスをコントロールしましょう。

sshd は libwrap をリンクしているので TCP Wrappers のアクセスコントロールを利用できます。

TCP Wrappers はシステムへのリクエストをモニタするプログラムです。モニタする機能のうち、/etc/hosts.allow と/etc/hosts.deny にそれぞれ許可、不許可のホスト及びネットワークを記述するこでアクセスコントロールを行うことができます。

アクセスコントロールファイルの書式は以下の通りです。(シェルコマンドについては本書では割愛します)

デーモン名 : クライアントリスト [: シェルコマンド]

「デーモン名」はデーモンのプロセス名です。sshd の場合「sshd」と記述します。サーバのポート番号やワイルドカードを指定することもできます。

「クライアントリスト」はホスト名、ホストのアドレス、クライアントにマッチするパターンやワイルドカード(すべてのホストを表す「ALL」や「.」を含まないホスト名を表す「LOCAL」など)を記述します。パターンは以下のように指定できます。

「.」で始まる文字列

後ろに続くドメイン部分にマッチするホストからの接続をコントロールします。

例えば「.example.com」と記述した場合、「www.example.com」や「mail.example.com」はアクセスコントロール対象になります。

「.」で終わる文字列

前に並ぶ IP アドレス部分にマッチするホストからの接続をコントロールします。

例えば「192.168.」と記述した場合、「192.168.1.1」や「192.168.245.245」がアクセスコントロールの対象となります。

ネットワークアドレス/ネットマスク（IPv4）

特定のネットワークのみを許可する場合は「192.168.1.0/24」か「192.168.1.0/255.255.255.0」と記述することでアクセスコントロール対象を指定できます。ホストを指定したい場合は「192.168.1.1」のように IP アドレスのみを記述します。「192.168.1.1/255.255.255.255」と記述すると有効になりません。

「255.255.255.255」は無効な記述方法です。

ネットワークアドレス/ネットマスク（IPv6）

IPv6 からのアクセス制御は「[n:n:n:n:n:n:n:n]/m」もしくは「Net/Prefix 長」のように記述します。

他にも「/」で始まる文字列の場合はホスト名、ネットワーク、ホスト IP アドレスなどを記述したファイルを指定したと判断します。また「?」と「*」を使ってホスト名、IP アドレスをマッチさせることもできます。

実際の設定ははじめに/etc/hosts.deny に、すべての接続を禁止する設定を記述します。

```
ALL: ALL
```

次に SSH サーバへアクセスできるホストを 192.168.11.1 だけに設定します。以下は /etc/hosts.allow に 192.168.11.1 からの sshd へのアクセスを許可する記述です。

```
sshd: 192.168.11.1
```

3.4.3. アクセス制御(認証)

OpenSSH でのリモートメンテナンスは、ネットワークによるアクセス制御と併せて認証の強化を行うことを推奨します。ここでは公開鍵認証による設定を説明します。

デフォルトの sshd はパスワード認証でログインができます。これは「ユーザ名」と「パスワード」があればシステムにログインすることができる、ということですが、ユーザ名とパスワードが同じであったり、予想可能な脆弱なパスワードであった場合には辞書攻撃などで簡単にシステムを乗っ取られる事もあります。

公開鍵認証を利用するメリットは、システムへ事前に登録した鍵を持つクライアントからのみアクセスできることです。

公開鍵認証は 2 つの鍵「キーペア」を作成します。キーペアは「秘密鍵」と「公開鍵」からなり、公開鍵をサーバに登録します。サーバに登録されている公開鍵のペアである秘密鍵を持つクライアントだけがサーバにログインできるようになります。

公開鍵から秘密鍵を作成することはできませんから誰かに公開鍵を見られても問題はありません。また秘密鍵を利用するためにパスフレーズを設定できます。これにより、(1) 公開鍵が登録されている事 (2) キーペアの秘密鍵をもつクライアントである事 (3) 秘密鍵を利用するパスフレーズを知っている事の条件でシステムへログインができるようになります。

設定はキーペアを作成することから始めます。ssh-keygen コマンドに-t オプションで鍵のタイプを指定します。鍵のタイプは RSA 認証を利用する場合の「rsa」と、DSA 認証を利用する場合の「dsa」が選択できますが、通常は RSA 認証を選択します。

実行すると鍵の保存場所とパスフレーズを 2 回尋ねられます。

それぞれ入力して鍵を生成しましょう。

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oruser/.ssh/id_rsa):
<- 鍵の保存場所指定
Enter passphrase (empty for no passphrase):   <-  パスフレーズを入力
Enter same passphrase again:                  <-  再度パスフレーズを入力
Your identification has been saved in /home/oruser/.ssh/id_rsa.
Your public key has been saved in /home/oruser/.ssh/id_rsa.pub.
The key fingerprint is:
XX:XX:XX:c9:cd:d3:fb:db:95:87:36:10:4d:99:83:e9 oruser@ubuntu
```

コマンドを実行した後、デフォルトで~/.ssh/id_rsa.pub に公開鍵が、~/.ssh/id_rsa に秘密鍵が保存されます。公開鍵の id_rsa.pub の方をログインしたいサーバにコピーします。

ssh-copy-id コマンドで目的のホスト名とログインするユーザ名を指定します。

```
$ ssh-copy-id  ユーザ名@ホスト名
```

これにより、システムのホームディレクトリに公開鍵が保存されます。

鍵はデフォルトでは ~/.ssh/authorized_keys に保存されます。

ただし、ssh-copy-id コマンドは既に公開鍵が登録されているユーザか、パスワード認証ができるシステムにしか利用できません。公開鍵認証になっているシステムに公開鍵を登録する場合は、管理者に本人確認などの手順を踏んだ後、公開鍵を渡して保存を依頼していください。

鍵を登録した後はサーバ側で sshd が公開鍵認証のみを受け付ける設定に変更します。

sshd の設定は以下のファイルを編集します。

```
/etc/ssh/sshd_config
```

以下のパラメータを「no」に設定し、sshd を再起動します。

```
PasswordAuthentication no
UsePAM no
```

再起動後、sshd へのアクセスには公開鍵認証のみになります。

3.4.4. アクセス制御(パケットフィルタリング)

アプリケーションレベルでのアクセス制御も重要ですが、カーネルの Netfilter モジュールを利用したパケットフィルタリングを活用する事も推奨されます。

Ubuntu ではパケットフィルタリングを構成するには iptables コマンドを利用します。

Netfilter パケットフィルタリングはルールとチェイン、テーブルを構成する事から始まります。

ルールはマッチングルールとターゲットからなります。マッチングルールはパケットヘッダの抽出方法、ターゲットはそのパケットの処理方法を定義します。以下はチェインにルールとターゲットを追加するコマンド表記法です。

iptables -A チェイン　マッチングルール　-j ターゲット

チェインは複数のルールから成り立ちます。指定するチェインにルールを追加、削除し、チェインを構成していきます。

テーブルは複数のチェインから構成されます。特に iptables でテーブルを-t オプションで指定しない場合は「filter」テーブルがデフォルトで使用されます。

以下は簡単なホスト型パケットフィルタリング構成です。チェインにはデフォルトの INPUT、FORWARD と OUTPUT を利用します。テーブルは特に指定していないのでデフォルトの「filter」が利用されています。

```
#!/bin/sh
$IPTABLES -F
$IPTABLES -P INPUT   DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT  ACCEPT

# localhost
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

# Accept ICMP and SSH
$IPTABLES -A INPUT -p ICMP -j ACCEPT # PING
$IPTABLES -A INPUT -p TCP --dport 22 -j ACCEPT # SSH

# Stateful
$IPTABLES -A INPUT -m state --state RELATE,ESTABLISHED -j ACCEPT
```

上記スクリプトを実行後、「iptables -L -n」を実行すると次のステータスが表示されます。チェイン INPUT、OUTPUT と FORWARD でのルール構成がわかります。

```
$ sudo /sbin/iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:22
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
```

「filter」以外のテーブルを表示する場合は「-t」をオプションに「nat」などを追加して表示することができます。

Ubuntu でシステム起動時にパケットフィルタリングを有効にするには /etc/networking/interfaces ファイルに up パラメータにスクリプトを渡すか /etc/network/if-up.d 以下にスクリプトを用意する方法があります。

スクリプト以外には iptable-save と iptables-restore を利用する方法があります。iptables-save は、設定されているフィルタリングルールを標準出力にダンプするツールです。iptables-restore は iptables-save でダンプされたファイルを読み込んで IP パケットフィルタを構成するツールです。

```
$ sudo sh -c "iptables-save > /etc/default/iptables.rule"
```

上記のコマンドのように IP テーブルをテキストファイルにダンプし、下記のように interfaces ファイルの up パラメータに指定することで、システム起動時に Netfilter を利用できます。

```
* /etc/network/interfaces
iface eth0 inet static
    :
  up iptables-restore < /etc/default/iptables.rule
```

Ubuntu ではもう少し簡単に iptables によるパケットフィルタリングを利用できるように ufw コマンドが用意されています。ufw コマンドの詳細については ufw(8)を参照してください。

3.5. デフォルトのユーザ設定ファイル

useradd コマンドで新規ユーザを作成した場合、/etc/skel/以下にあるファイルが新規ユーザのホームディレクトリにコピーされます。全ユーザ共通の設定や追加設定ファイルなどがあれば、あらかじめ/etc/skel/以下に置いておくと良いでしょう。

また、環境変数設定以外にも/etc/profile.d/以下に「.sh」の拡張子の実行権をつけたシェルスクリプトを用意することで、ログインシェルが起動される度に環境を調整するスクリプトを実行することもできます。

3.6. 日本語環境

日本語の入力、表示について説明します。

3.6.1. locale

locale は、環境変数 LANG に言語環境を定義します。

locale の記述方法は、「xx_YY.ZZZZ」です。

各コンポーネントは以下の意味を持ちます。

> **xx: 言語コード**
> **YY: 国コード**
> **ZZZZ: エンコーディング**

Ubuntu 10.04(lucid lynx)の日本語環境ではデフォルトで「ja_JP.UTF-8 (ja_JP.utf8)」が設定されます。

3.6.2. インプットメソッド

インプットメソッドとは、キーボード入力から文字を入力するソフトウェアで、平仮名や漢字などの入力に利用されます。

一般的に利用されるキーボードはアルファベット入力用になっており、日本語を入力する場合は複数のキー入力が必要になります。(例えば「に」であれば「n」と「i」を入力する)

Linux のインプットメソッドとしては、主に2種類の方式が利用されています。

一つは XIM で、もうひとつは immmodule です。

X Window System 上で稼働するアプリケーションの場合、XIM(X Input Method)という入力のためのフレームワークが用意されています。キーボードからの入力をアプリケーションが受け取り、XIM サーバと通信します。XIM サーバは仮名漢字変換サーバと通信して変換した文字列を受け取り、アプリケーションに渡します。

immodule は GUI ツールキットの gtk+2.0 以降と qt に実装されているモジュールで、XIM に比べて動的に入力言語を切り替えが容易などの特徴があります。

3.6.3. Ubuntu 10.04(lucid lynx)で利用できるインプットメソッド

Ubuntu 10.04(lucid lynx)で利用できるインプットメソッドを以下に示します。

- iBus （XIM, immmodule ともに利用可能）
- SCIM （XIM, immmodule ともに利用可能）
- UIM （XIM, immmodule ともに利用可能）
- kinput2-{canna,wnn} （XIM のみ利用可能） … glclient2 では使用不可

Ubuntu 10.04(lucid lynx)ではデフォルトのインプットメソッドは ibus が利用されています。

3.6.4. 仮名漢字変換サーバ

仮名漢字変換サーバはインプットメソッドと連動してキーボードの入力を受け取り仮名や漢字に変換し、アプリケーションに文字を入力するアプリケーションです。

3.6.5. Ubuntu 10.04(lucid lynx)で利用できる仮名漢字変換サーバ

Ubuntu 10.04(lucid lynx)で利用できる仮名漢字変換サーバを以下に示します。

- anthy
- canna
- prime
- skk
- freewnn

Ubuntu 10.04(lucid lynx)ではデフォルトで anthy が利用されます。また、ORCA プロジェクトでは glclient2 向けに anthy の辞書パッケージ orca-medic を提供しています。

Ubuntu 12.04 では Google 日本語入力のオープンソース版である Mozc が使用できるようになります。

3.6.6. 日本語入力方法

iBus と Anthy による日本語入力方法は以下に「日本語の入力方法」という資料がありますので参照ください。

- http://www.orca.med.or.jp/receipt/tec/lucid/doc/Lucid_Anthy.pdf

3.6.7. Takao フォント

Ubuntu 10.04 から日レセでは IPA フォントを派生させた Takao フォントを採用しています。また Ghostscript を使用しての印刷には、CID フォントに変換した「ORCA CID フォント」が使用されます。このフォントは JIS X 0213:2004（JIS 第 1?第4水準）までの文字が収録されています。字形も JISX0213:2004 に準拠しているため、これまでのフォントと一部字形が異なるフォントがあります。（例：「辻」）

- 参考：「JIS 漢字コード表の改正について-168 字の例示字形を変更」
  http://warp.da.ndl.go.jp/info:ndljp/pid/243519/www.meti.go.jp/kohosys/press/0004964//

一部の字形を変更前に戻した JISX0208（旧字形）フォントを ORCA プロジェクトで用意していますので、以下の URL を参照してください。

- http://www.orca.med.or.jp/receipt/download/lucid/jisx0208.html

3.7. X Window System

3.7.1. 設定ファイルと設定方法

Ubuntu では X Window System に X.org を採用しています。X.org は Linux の GUI フレームワークを提供しています。Gnome や KDE などのデスクトップ環境やウィンドウマネージャは X.org のアーキテクチャの上で稼働しています。

/etc/X11/xorg.conf

Ubuntu では xserver-xorg パッケージをインストールした時にハードウェアを自動認識できれば自動で設定ファイルが作成されます。 自動生成された設定から変更する場合には xorg.conf を編集します。

X フォワーディング

他ホストの X アプリケーション (クライアント) を手元の X の画面 (サーバ) で遠隔利用する際、X アクセス制限の設定が必要です。

ただし xhost+などによる設定はセキュリティの問題上するべきではありません。

特別な理由がない限り、sshd の X フォワーディング機能を使いましょう。

sshd の X フォワードを利用するには、sshd で X11Forwarding パラメータを yes に変更します。sshd を再起動した後、X の転送ができるようになります。

次に接続するクライアントから以下のように入力します。

```
$ ssh -l  ユーザ名  SSH ホスト  -X  <コマンド>
```

たとえば、リモートの 192.168.11.244 というホストの xeyes をローカルの X に表示する場合は以下のようにコマンドを実行します。

```
$ ssh -l orca  192.168.11.244  -X  xeyes
```

ターゲットのホストにターミナルでログインしてから X アプリケーションを転送することもできます。

```
$ ssh -l orca 192.168.11.244  -X
Enter passphrase for key '/home/oruser/.ssh/id_rsa':
$ xeyes  <-  手元のディスプレイにアプリケーションが表示される
```

3.7.2. 利用カードが対応していない

Ubuntu 10.04(lucid lynx)は Xorg version 7.5 を利用しています。

一部の最新のグラフィックカードには対応していないものや性能が十分に発揮出来ない場合があります。

restricted コンポーネントに収録されているフリーではないパッケージのドライバが利用で

きるかもしれません。Web を検索して情報を探してみましょう。

3.7.3. ログファイル

X サーバの自身のログは/var/log/Xorg.0.log に記録されます。また X 上でのユーザセッションに関するログは~/.xsession-errors などに記録されます。

X サーバの設定などに問題がある場合には前者にエラーメッセージが記録される。ユーザ設定（たとえば~/.xsession） に問題がある場合には後者になんらかのメッセージが記録されます。

3.7.4. 突然マウスやキーボードの反応がなくなってしまった場合の対応方法

単にキーボードやマウスのコネクタが緩んでいる場合もあります。

また、反応がなくなったからといってシステムが完全にフリーズしているとは限りません。別ホストからリモートログインして、X だけを停止して復旧できる場合もあります。

作業中のデータがディスクに保存されていない場合もあります。リセットボタンを押して強制的に再起動させるのは最後の手段にしましょう。

3.8. PostgreSQL

3.8.1. PostgreSQL の設定ファイルとディレクトリ構造

Ubuntu の PostgreSQL 8.4 パッケージのディレクトリ構成は以下のとおりです。

- /etc/postgresql/ <- 設定ファイルのディレクトリ
- /var/lib/postgresql/ <- データ格納ディレクトリ
- /var/log/postgresql/ <- ログファイルディレクトリ
- /usr/lib/postgresql/8.4/bin/ <- 管理用コマンドディレクトリ

PostgreSQL 8.4 の主な設定ファイルは以下にあります。

/etc/postgresql/8.4/main/pg_hba.conf

ホストベース認証の設定を記述するファイルです。

/etc/postgresql/8.4/main/pg_ident.conf

pg_hba.conf で ident 認証を指定した場合に Unix アカウントと PostgreSQL ユーザをマップする設定ファイルです。

/etc/postgresql/8.4/main/postgresql.conf

PostgreSQL サーバのディレクトリ構成やポート番号など、全体的な設定ファイルです。

/etc/postgresql/8.4/main/start.conf

init スクリプトを利用した自動起動を調整するファイルです。

3.8.2. 動作確認

postmaster プロセスが動いているからといって正しく動作しているとは限りません。少なくとも、psql コマンドを使ってアクセスしてみるなどのテストは必要です。もちろん実運用時と同じ方法でアクセスするのが最も望ましいでしょう。

3.8.3. ネットワークからのアクセス

PostgreSQL サーバへのアクセスは自ホストのみから行える設定にすることも多いですが、日医標準レセプトソフトでは 2 台での運用が前提になっているため、主サーバ、従サーバ「以外」のホストからのアクセスを適切に制限する必要があります。 アクセス制限を行うには 3.8.1 で紹介した設定ファイルの内容を調整します。

/etc/postgresql/8.4/main/postgresql.conf

このファイルはデータディレクトリ、認証設定ファイル、ポート番号、接続最大数などを調整できます。

postgresql パッケージをインストールした直後のデフォルトの状態では Unix ドメインソケットからのアクセスのみを受け付ける設定になっています。外部からのアクセスを許可するには「listen_addresses」パラメータを変更し、クライアントからの接続を受け付ける IP アドレスか、「*」を記述します。

/etc/postgresql/8.4/main/pg_hba.conf

クライアント認証を設定するために、接続を許可するホスト、データベース、PostgreSQL ユーザ名、アドレス範囲、認証方法を 1 行で定義します。記述方法は以下のとおりです。

ホスト種別　データベース名　ユーザ名　アドレス　メソッド　オプション

「ホスト種別」は 4 種類用意されています。Unix ドメインソケットを利用する「local」、SSL 暗号か暗号化せずに TCP/IP を利用する「host」、SSL で暗号化された TCP/IP を利用する「hostssl」、SSL で暗号化しない TCP/IP 接続の「hostnossl」のいずれかを指定します。

「データベース名」はデータベースの名前を指定します。データベース名には特殊な名前が三種類あります。すべてのデータベースを示す「all」、データベース名とユーザ名が同じ場合の「samename」、データベース名と同じ名前の role 名に所属するユーザがアクセスした場合の「samerole」です。それ以外の場合は、特定の PostgreSQL データベースの名前です。

「ユーザ名」はデータベースに接続できるユーザ名を指定します。データベースユーザ名以外に「all」はすべてのユーザ、名前の前に「+」をつければグループ名を指定できます。複数ユーザを指定する場合は「,」で区切って指定します。また、「@」を利用して複数ユーザリストを記述したファイルを指定することができます。

「アドレス」は IP アドレスと CIDR アドレスマスクを指定します。特定の一つのホストであれば「192.168.11.20/32」と、特定のネットワークであれば「192.168.11.0/24」と記述します。ここでは IPv6 の記述方は割愛します。

「メソッド」は認証する方法を定義します。認証方法は 11 種類用意されています。接続を無認証で許可する「trust」、どの接続も拒否する「reject」、 クライアントに MD5 パスワードを要求する「md5」、 crypt()による暗号化したパスワードを要求する「crypt」、クライアントに平文パスワードを要求する「password」、クライアントに GSSAPI を要求する「gss」、クライアントに SSPI を要求する「sspi」、クライアントに Kerberos 認証を要求す

る「krb5」、UNIX アカウントを ident 経由で取得し、データベースと一致するか検査する「ident」、PAM を利用したユーザ認証を行う「pam」か、LDAP サーバを利用して認証する「ldap」のいずれかを選択します。

なお、「password」を選択すると平文でパスワードを送受信するので安全でないネットワークで利用する場合は「SSL」の接続方式を選択してください。

参考:

- 2 台運用の設定 (設定手順)
  http://www.orca.med.or.jp/receipt/use/db_2/index.html
- MONTSUQI - PostgreSQL SSL 接続
  http://ftp.orca.med.or.jp/pub/data/receipt/use/montsuqi_SSL_PostgreSQL.pdf

日レセの二台構成では pg_hba.conf に以下の記述を追加しています。これは TCP/IP 接続で、すべてのデータベースに対し、データベースユーザ「orca」で 192.168.1.11 からのパスワード認証での接続を許可しています。

```
host   all   orca   192.168.1.11/32   password
```

/etc/postgresql/8.4/main/pg_ident.conf

pg_hba.conf の認証メソッドに ident を利用した場合のみ参照されます。引数に渡す名前から Unix アカウント名と PostgreSQL ユーザ名を関連付けます。

書式は以下のとおりです。

```
MAP    IDENTNAME    PGNAME
```

以下の例のように pg_hba.conf が設定された場合、対になる pg_ident.conf が必要です。pg_hba.conf ファイルはデータベース db_test に対し、Unix ドメインソケット接続で、usergroup に含まれるすべてのユーザに接続できます。このとき、usergroup に含まれるユーザは pg_indent.conf で指定されます。

```
local  db_test   all    ident     usergroup
```

上記の対となる pg_ident.conf は以下のように記述します。マップ名 usergroup は、Unix アカウント名 oruser と PostgreSQL ユーザ名が orca のユーザを示します。

```
usergroup       oruser    orca
```

上記のような PostgreSQL 独自の認証方法に加え、デフォルトでは 5432/tcp へのパケットフィルタでアクセス制御する事を推奨します。

3.8.4. データベースの作成

Ubuntu 10.04(lucid lynx)の PostgreSQL 8.4 からは、各データベースでロケール、エンコーディングを選択できるようになっています。そのため、Ubuntu 8.04(hardy helon)までは PostgreSQL インストール時に自動作成される初期クラスタを削除し、再作成していましたが 10.04 からは createdb 実行時にロケール、エンコーディングオプションをつけてデータベースを作成する手順に変更されています。

Ubuntu 10.04(lucid lynx)では、データの移行やバックアップからのリストア時にデータベースを作成する場合は必ず以下のように createdb にオプションをつけてください。

```
$ sudo -u orca created -lC -Ttemplate0 -EEUC-JP orca
```

createdb にオプションをつけなかった場合は、OSのロケールに従い初期データベースにロケール、エンコーディングともに「ja_JP.UTF-8」で作成されますのでご注意ください。

ロケールについて

PostgreSQL では、ロケールに ja_JP.UTF-8 を使用した場合、ソート順が異なったり、一部のSQLでインデックスが有効にならなかったりする動作が起こるため、日医標準レセプトソフトではロケールはオプションに「-lC」をつけて「C」を利用することを推奨します。

また、PostgreSQL では template1 とロケールが異なる場合は template0 をテンプレートとして利用する必要があります。

エンコーディングについて

Ubuntu 10.04(lucid lynx)からは、エンコーディングに従来の EUC_JP に加えて UTF8 を使用できるようになりました。EUC_JP では JISX0208:1983 の第一、二水準まで使用可能だったのに対し、UTF8 の場合は JISX0213:2004 の第一、二、三、四水準まで使用できます。

ただし、連携ソフトやクライアント等で JISX0213:2004 に対応できてない場合はあり、その際は従来通り EUC_JP でセットアップする必要があります。

エンコーディングの UTF8 使用については以下の「拡張漢字(JISX0213:2004)の使用」を参照してください。

- http://www.orca.med.or.jp/receipt/use/jisx0213/index.html

3.8.5. データベースのバックアップとリストア

データベースは定期的にバックアップを実行する事をお勧めします。pg_dump コマンドでデータベースのダンプを作成し、必要なときに最新のバックアップから復旧したシステムへリストアできることが理想です。

以下は Unix ユーザ名 orca で、データベース orca をダンプする例です。

```
$ sudo -u orca pg_dump -Fc orca > orca.dump
```

リストアはデータベースを作成してからデータをリストアします。

```
$ sudo -u orca pg_restore -O -d orca orca.dump
```

3.8.6. 定期保守

定期的にデータベースのバックアップを行う事をお勧めします。また、データベースには個人情報が含まれるため、バックアップファイルについては医院外部に漏洩することがないように細心の注意を払う必要があります。

バックアップファイルを数世代分保持する場合は、データ領域容量確保のために、定期的に古いダンプデータから削除するようにしましょう。

3.8.7. クラスタ

Ubuntu 10.04(lucid lynx)で利用される PostgreSQL は、以前のバージョンの 7.4 や 8.3 の PostgreSQL とのデータ互換性がない為、異なるバージョンの PostgreSQL が並行して稼働できるようにクラスタというアーキテクチャを採用しています。ここでのクラスタはデータベースの集まりを意味します。

Ubuntu ではクラスタを構築する以下のコマンドを用意しています。

pg_lscluster

クラスタをリスト表示します。

pg_createcluster

initdb のラッパーで、クラスタとその設定ファイルを作成します。

pg_ctlcluster

pg_ctl のラッパーで、postmaster サーバや pg_autovacuum デーモンのコントロールをします。

pg_upgradecluster

新しいメジャーバージョンへアップグレードします。

pg_dropcluster

クラスタとその設定ファイルを削除します。

3.8.8. PostgreSQL のアップグレード

基本的には、現在利用中のデータベースはそのままアップグレードできますが、扱っているデータの重要性を鑑みて自前でもバックアップをとっておくべきでしょう。

PostgreSQL をアップグレードするタイミングには以下が考えられます。

- PostgreSQL のセキュリティアップデートのリリース時
- Ubuntu のメンテナンスリリースで PostgreSQL のアップデートが含まれていた場合
- Ubuntu のバージョンをアップする場合

日々のメンテナンスでもアップデートされるパッケージに注意し、バックアップを行ってから適用するなどの対応が望ましいでしょう。

なお、クラスタ管理ツールを利用すれば比較的容易にアップグレードが行えます。

たとえば 8.3 のシステムから 8.4 のシステムにアップグレードする方法は以下のとおりです。繰り返しますが、実行前にデータベースのダンプを必ず取得してください。

すでに postgresql-8.3 のパッケージがインストールされており、8.3/main のクラスタが動いている環境に、postgresql-8.4 をインストールします。

```
$ sudo aptitude install postgresql-8.4
```

パッケージをインストールすると 8.4/main のクラスタを自動で作成してしまいます。始めにこのクラスタを削除します。

```
$ sudo pg_dropcluster --stop 8.4 main
```

つぎに pg_upgradecluster を利用して 8.3/main から 8.4/main にアップグレードします。

```
$ sudo pg_upgradecluster -v 8.4 8.3 main
```

3.9. カーネルとブートローダ

Linux カーネルとブートローダについて説明します。

3.9.1. カーネル

Ubuntu だけでなく、すべての Linux ディストリビューションは Linux カーネルを利用しています。ディストリビューションは Linux カーネル、libc ライブラリ、X Window System、GUI ツールキット、DBMS、その他各種アプリケーションを一纏めにしたものです。

Linux カーネルは Linux Foundation を中心に世界中のプログラマが開発を続けています。

http://www.kernel.org

開発元で配布するカーネルを「vanilla カーネル」と呼ぶこともあります。Ubuntu や Redhat など、ディストリビュータはそれぞれユーザの要望を叶えるため、セキュリティホールを埋めるためなどの理由で vanilla カーネルに独自のパッチを適用しています。

Ubuntu では、Server Edition と Desktop Edition で異なるパッチが適用されたカーネルパッケージが配布されています。

Ubuntu で配布されるカーネルパッケージは以下のとおりです。

linux-image

カーネルイメージ、System.map ファイルなどが含まれるパッケージです。

linux-headers

カーネルに関連する C 言語ヘッダファイルのパッケージです。

linux-source

カーネルのソースコードと、Ubuntu で利用するカーネルで適用しているパッチを集めたパッケージです。

linux-backports-modules-*

Ubuntu でディストリビューションのカーネル用にバックポートしているカーネルモジュールパッケージです。USB 無線 LAN アダプタ、無線 LAN アダプタ、ネットワークアダプタ、タブレットなどのドライバが含まれます。

カーネルは /boot 以下に vmlinuz-<カーネルバージョン>-<ビルド番号> のファイル名で保存されています。/boot 以下にはカーネルイメージの他に以下のファイルがあります。

/boot/System.map-2.6.32-24-server

カーネル内の重要な変数や関数のアドレスを記したテキストファイル。デバッグなどで利用します。

/boot/config-2.6.32-24-server

カーネルの設定テキストファイル。再ビルドする際に参考にします。

/boot/initrd.img-2.6.32-24-server

initramfs イメージファイル。HDD のルートファイルシステムをマウントする前にメモリに展開する小さな環境で、デバイスドライバなどが含まれます。

/boot/memtest86+.bin

メモリテストツール。

/boot/vmlinuz-2.6.32-24-server

カーネルイメージ本体。

3.9.2. ブートローダ

Ubuntu 10.04(lucid lynx)はブートローダに grub2 を採用しています。

grub はコマンドラインでのインターフェイスが用意されていたり、設定情報とブートレーコードが分離されている使い勝手のよさから現在のほとんどの Linux ディストリビューションでブートローダとして採用されています。

grub はバージョン 1.x の「grub-legacy」とバージョン 1.9 以降の「grub2」があり、10.04 では「grub2」を利用します。ブートローダの主要な設定ファイルは/etc/default/grub.conf、/boot/grub、/etc/grub.d/以下にあります。

* /etc/default/grub.conf

このファイルではブートするデフォルトイメージの番号、メニューを隠す時間とタイムアウト、カーネルのオプションなどを指定します。このファイルを編集した後は update-grub コマンドを実行する必要があります。

GRUB_DEFAULT=0

grub.cfg にある位置によってデフォルト OS を指定します。0 の場合は最初に記述されているカーネルイメージを示します。

GRUB_TIMEOUT=10

デフォルト OS が起動するまでの時間を指定します。-1 を指定するとユーザが OS を指定するまで待ちます。

GRUB_HIDDEN_TIMEOUT=0

コメントアウトのままではメニューは表示されません。有効にした場合は起動までの待機秒数を指定します。

GRUB_HIDDEN_TIMEOUT_QUIET=true

false を指定した場合、GRUB_HIDDEN_TIMEOUT で指定された秒数をカウントダウンします。

GRUB_CMDLINE_LINUX=""

通常起動・リカバリの際に Linux カーネル起動オプションを指定します。

GRUB_CMDLINE_LINUX_DEFAULT="quiet"
通常起動の linux カーネル起動オプションの最後に指定された文字列を付け足します。

* /boot/grub/grub.cfg
update-grub コマンドで生成されるファイルです。grub-legacy では/boot/grub/menu.list にあたるファイルで、起動する OS のメニューリスト情報を記述します。update-grub コマンドは/etc/grub.d/以下のスクリプトを実行し、その出力を/boot/grub/grub.cfg に出力します。 カーネルのアップデートを実施すると update-grub が実行されるので/boot/grub/grub.cfg を直接編集するのは推奨しません。

* /etc/grub.d/
update-grub を実行するとこのディレクトリ以下にあるスクリプトが順番に実行され、結果を/boot/grub/grub.cfg に書き込みます。実行はスクリプトファイル名の先頭数字の順番に行われます。各スクリプトは以下の目的があります。

00_header	必要なスクリプトなどを出力する
05_debian_theme	テーマ、背景などを出力する
10_linux	ブロックデバイスから Linux カーネルを検索する
20_memtest86+	memtest86+をメニューに追加する
30_os-prober	Linux 以外の OS をブロックデバイスから検索する
40_custom	ユーザカスタマイズ用スクリプト

3.9.3. カーネルのアップデート

カーネルのアップデートの際は以下の事に注意しましょう。

update-grub の自動実行

カーネルパッケージインストール後、/boot/grub/grub.cfg の生成を自動で実行したくない場合は/etc/kernel-img.conf の postinst_hook パラメータの行頭に「#」を追加してコメントアウトしましょう。

grub.cfg の default パラメータ

/boot/grub/grub.cfg には起動するイメージの番号を指定する「default」パラメータがあります。これをデフォルトの「0」以外に指定している場合、アップデート後の再起動で予期せぬイメージを読み込んで起動する場合があります。「default」パラメータはカーネルアップデート直後、再起動する前にチェックしましょう。

3.9.4. grub シェルの利用

なんらかの事故で grub 設定ファイルを更新した後にカーネルが起動しなくなることがあります。すでに起動実績のあるカーネルが存在する場合、grub のメニュー画面から別のカーネルを指定して起動することもできますが、grub パラメータを間違えた場合はパラメータを修正する必要が出てきます。

ここではgrubメニューに出てこないが、インストールされている別のカーネルをシェルから起動する手順を示します。起動するカーネルの情報を以下のように定義します。

- プライマリーのSATA HDDの第一パーティションに/bootがある。
- イメージのファイル名は vmlinux-2.6.32-24-generic
- ルートパーティションは/dev/sda1
- initramfsを利用し、イメージファイル名はinitrd.img-2.6.32-24-generic

1. grubによりカーネルが起動される前に「**SHIFT**」キーでメニューを表示します。

図 2 grub起動画面

2. grubメニューで「c」をタイプしてシェルを起動します。

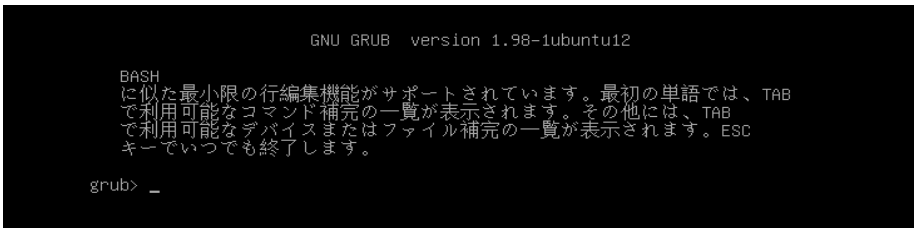

図 3 grubシェル画面

3. ルートデバイスを指定します。あらかじめ/boot/grub/grub.cfgを参照してルートデバイスはメモしておきましょう。

```
grub> root (hd0,0)
```

4. カーネルとルートパーティションを指定します。イメージファイル名を途中まで入力してTabをタイプすると可能であればファイル名が補完されます。

```
grub> linux  /boot/vmlinuz-2.6.32-24-generic root=/dev/sda1
```

5. initrdイメージを指定します。

```
grub> initrd /boot/initrd.img-2.6.32-24-generic
```

6. 「boot」とコマンドしてブートします。

```
grub> boot
```

カーネルパニックなどになってしまった場合に備えて grub シェルの簡単な使い方はマスターしておきましょう。

3.9.5. パスワードを忘れてしまった場合の対応方法

Ubuntu では root ユーザを利用しません。一般ユーザで sudo を利用して管理者コマンドを実行します。管理者権限を付与していたユーザアカウントのパスワードを失念する等、何らかの問題が発生した場合にはパスワードを再設定で対応します。

パスワードを再設定する場合は以下の手順で行います。

1. システムを再起動し、起動画面で「**SHIFT**」キーを押して grub メニューを表示します。

図 4 grub メニュー画面

2. 「(recovery mode)」の行を選択して起動します。起動後、以下のウィンドウが出てくるので「root」を選択すると root ユーザのシェルが起動します。

図 5 リカバリメニュー画面

日本語環境でインストールするとレスキュー画面が上記のように読めないことがあります。メニューには以下の項目があります。

・resume: recovery mode を抜ける。
・clean: apt-get clean を実行して空き領域を作ろうとする。
・dpkg: apt-get install -f や dpkg --configure などを実行して壊れたパッケージデータベースを修復しようとする。
・failsafeX: /etc/gdm/failsafeXServer を実行するして X を起動しようとする。
・grub: update-grub を実行して grub 設定ファイルを再作成する。
・netboot: dhclient を実行して sulogin でシングルユーザログインする。
・root: ネットワーク接続なし、sulogin でシングルユーザログインする。

3. 起動したシェルで passwd コマンドのオプションにパスワードを設定するユーザ名を指定して実行し、パスワードを変更します。

```
# passwd ユーザ名
```

また、意図的に grub の起動メニューから recovery mode を消している場合は以下の手順でも変更できます。

1. grub メニューの通常起動しているカーネルを「e」で選択します。

図 6 カーネル起動オプション表示画面

「linux=」で始まる行で「e」で編集モードにし、行末に「single」を追加します。

図 7 設定編集画面

2. 「root」を選べる画面が表示されるので、root シェルに移動し、パスワードを変更しましょう。

4. セキュリティ

4.1. セキュリティについてのヒント

ORCA プロジェクトでは医療の IT 化を推進するため、医療の現場におけるネットワークの活用を常に考えています。ORCA プロジェクトの最初の成果である日レセはネットワーク対応のレセコンです。しかし、ネットワークに接続できれば良いというわけではありません。レセコンで扱う情報には機密に相当する情報も含まれるため、情報そのものの取り扱いに注意する必要があることはもちろん、いわゆるクラック行為などに対する備えが必要となってきます。ここでは日レセの運用に必要となるセキュリティの考え方についての基礎的な知識について説明します。

4.2. 危険の分析

4.2.1. セキュアなシステム

システムを安全な状態に保つためには、どのような状態が望ましいのかを定義してみましょう。一般にセキュアな状態であるためには次の 3 つの要素が必要とされています。

- 機密性　見られてはいけない人に対して、見られては困るものを見せない。
- 完全性　情報が正しく、改変されることなく伝えられる。
- 可用性　必要とされる時には常にサービスを利用できる。

このうち、どれが欠けても望ましいとは言えません。

利便性のことばかり考えてセキュリティにまで考えが及ばないのも、セキュリティを高めようとばかりして、サービスを阻害してしまうのも、どちらも同じように望ましくない状態と言えます。

4.2.2. 様々なレベルでの「危険」

システムがセキュアな状態でなくなる要因は、数え切れない程あります。まずは危険因子を分類してみましょう。

「人」に起因するもの

「人」はコンピュータシステムに限らず一般の社会全般にひそむ危険といえます。

- 機密情報が書かれた書類、HDD などの記録媒体の取り扱い
- 電話などでパスワードなどの機密を聞き出すソーシャルエンジニアリング
- ニセ情報を与えて混乱をひきおこす
- 内部犯の可能性

ある意味では「人」がもっとも弱いと言えます。危機管理、身のまわりの危険に対する意識が勝手に向上することはありません。日々注意することが重要です。書類、記録媒体の破棄の前に破壊、怪しい情報は必ずソースを確認するなどしましょう。

ハードウェアの運用・管理に起因するもの

ハードウェアに直接手を触れられるところに行くことができれば、幅広い攻撃が可能となります。

- 電源を切る

- ハードウェアごと盗む
- システムを物理的に破壊する

攻撃者にとって、ソフトウェア的な特殊な技術領域に踏み込むよりも、このような「攻撃」の方が確実に攻撃対象にダメージを与えられるケースもあります。盗難に関しては盗難防止ベルトを設置する、USB メモリを刺されないように USB ポートをふさぐ、など対策が考えられます。

ソフトウェアの運用・管理に起因するもの

設定ミスや運用上の間違いにより、見せてはならないものを見せてしまったという事件は実際にあります。

- 設定ミス
- リンクをしないだけの「隠し」ファイルを web サイト上に置いてしまう
- ずさんなパスワード運用やアカウント管理
- 特権ユーザ (root) による作業が管理されていない
- P2P ツールの利用

セキュリティ侵害はグローバルネットワークから来るだけとは限りません。攻撃者は複合技で目的を達成しようとします。また攻撃されてなくても P2P を利用してしまったがために公開しなくていい情報を不特定多数に公開してしまうことも想定されます。

ソフトウェア自身の欠陥

ソフトウェアは人が作ったものであり、バグがないシステムは存在しないでしょう。しかし、バグにも以下の種類が存在します。

- 設定によって回避できるバグ
- システムの利用に影響するバグ
- セキュリティーホール

アプリケーションウィンドウのタイトルに typo がある程度では業務に影響はないでしょうが、レセプトが出力できないような問題では緊急度も上がります。

第三者が悪用できるバグ (セキュリティホール) が見つかった場合には早急な対応が必要になるでしょう。

情報サイトなどでソフトウェアの脆弱性情報が報じられますが、冷静に脆弱性のレベルを認識し、緊急度を判断するのがいいでしょう。

ネットワークからのアクセスに起因するもの

ネットワークに接続した瞬間から、コンピュータはネットワーク経由の攻撃にさらされると見て間違いはありません。現在の主流な攻撃方法は以下のとおりです。

- 許可した覚えのないアカウントでの外部からの「侵入」
- データ「破壊」やデータの「盗難」
- ウィルスの「侵入」
- スパムメールやボットの「踏み台」
- 各種の DoS (サービス妨害)

ネットワーク運用に起因するもの

セキュリティ対策を考える場合、こうした様々な面についての対策がそれぞれに必要となります。

- 事務所 LAN に接続されたハブの LAN ポートが丸見えで、容易に接続できる LAN
- MAC アドレスアクセス制御や暗号化などがされていない無線 LAN

アクセスコントロールの不備

たとえある特定の部分についての対策を強化したとしても、他に手薄な部分があるとシステム全体のセキュリティ的な強度はその手薄な部分に依存してしまいます。

また、やみくもに「対策」を進めても、結局はバランスを欠いたものになりがちです。

4.3. 情報セキュリティポリシーの作成

どのような方針でセキュリティ対策を考え、具体的にはどういったルールの下で運用するかといったことを定義したものを情報セキュリティポリシーと呼びます。

具体的な内容は組織ごとに違ってきますが、ポリシーの策定にあたっては NPO 日本ネットワークセキュリティ協会 (http://www.jnsa.org/) で作成され、公開されている「情報セキュリティポリシーサンプル」を参考にすることができるでしょう。

> ノート
>
> @IT (http://www.atmarkit.co.jp/) の Security & Trust に掲載された「情報セキュリティポリシー入門」という講座も興味深い内容で大変参考になります。

組織の規模によっては、あまり大がかりなポリシーを策定する必要のない場合もありますが、そうした場合でも基本的な方針を定め、その中で具体的な行動をとるようにすべきです。

逆に、規模の大きな組織においてはきちんとしたポリシーを設定し運用することが推奨されます。前述の「情報セキュリティポリシーサンプル」の他に、各種の書籍も出版されていますし、コンピュータセキュリティを扱う業者によってはポリシー策定の支援サービスを行っています。定めたポリシーが適正なものでなければ逆効果になることもあり得ますので、必要に応じてそうしたサービスが検討されることもあります。

階層的なポリシー

組織に設定するポリシーは必ずしも均一であるとは限りません。業務内容によって行動範囲や触れるべき機材・情報は違ってきますから、むしろ、業務区分ごとに適した細則を設定する方が普通と言えます。

ただし、そのようなケースでもコアとなるポリシーが設定され、それに逸脱しないように細則が設定されます。その意味ではポリシーは全体として階層構造をとることも少なくありません。その場合、以下のような構造とするのが一般的なようです。

基本ポリシー

その組織においてセキュリティへの取り組みを行うという宣言。組織全体で運用する必要があるため、組織の最高責任者による説明が必要とされる。

スタンダード

実際に守るべきルールを定める。どういう目的で何をやらねばならないのかということをわかり易く記述する。様々な角度からの視点が必要とされ、必要に応じて細分化されることもある。

プロシージャ

いわゆるマニュアルにあたる。スタンダードを運用するための、さらに具体的な行動規定を定める。この部分は一般に、業務内容や情報システムにおける役割に応じて細分化されるべきである。

体制作り

ポリシーの策定とそれに続く運用にあたり、なんらかの体制をととのえておいた方が良い

でしょう。

設定されたルールは運用しなければ意味がありませんが、運用主体のないルールがうまく機能することはあまりありません。罰則などが必要かどうかはその組織での判断によりますが、何らかの監査的な機構が必要になるケースもあるでしょう。

また、ルールを運用するためのチームと併せて、万一のときのための対策チームなどについても検討すべきかもしれません。

万一のために

万一のことが起こったときのために、最低でも次のことを明らかにしておくことをおすすめします。

- 通知先

組織内の対策チームや、場合によっては専門の業者になるでしょう。

- 一次対応の内容

いち早く連絡をするのは当然として、その場で何をすべきか、何をすべきでないかについてはわかりやすく示しておいた方がよいでしょう。

4.4. 啓発

システムを安全にするために人の教育や啓発、実践はやはり重要になってきます。

- 教育
- 啓発
- 運用
- フィードバック

上記の手順を、段階的に進めなくてはなりません。また、施策したところで終えてしまっては、本当に有効なのか、ユーザにとって過度に不便を強いていないか、などといった点を検証することができません。

後のフォローまで含めて計画を立てる必要があります。

4.5. チェックポイント

ここで、セキュリティに関するチェックポイントのいくつかを挙げておきます。ただし、ここですべてのポイントを網羅することはできませんので、あくまで参考にする程度にとどめてください。

> 注意
>
> 前述した通り、必要に応じてポリシーを定め、その上で細則を決めていく必要があります。また、その内容は実際の環境によって大きく変わり得ることにも注意してください。どこにでも通用する絶対的なマニュアルはありません。

4.6. 一般ユーザ

良いパスワードを使うことは基本中の基本です。

- できるだけ乱雑で、かつ、できれば覚えられるものを利用する。
- 辞書に載っていない。
- キー配列から類推できない。
- 記号を含める。
- pwgen コマンドなど、パスワード生成ツールを利用する。
- 覚えられるように簡単なものにするよりは、紙に書いておく方が良いということもある。そして誰にも見られてはいけない。
- 定期的に変更する。
- 複数のシステムで同じパスワードを使わない。

日常の操作

何気ない操作が危険につながることもあります。

- ログインしたまま離席しない
- 離席するならスクリーンをパスワードロックしておく
- 不審なプログラムを実行するようなことはしない

Linux というシステムでは一般ユーザの権限でできることは限られます。

しかし、日レセをはじめとする各種のアプリケーションの中には特権ユーザではない専用の一般ユーザの権限で動作させるようなものもあり、そうしたアプリケーションにとって専用ユーザの権限をうばわれることは一般的な Linux システムにおいて root をうばわれるのと同じです。

> ノート
>
> 何をすると (されると) 、どこに、どのような影響があるかを常に考える必要があります。また、ポリシーにおいてはそうした部分をカバーできるように設定できるとよいでしょう。

4.7. 管理者

ソフトウェアの管理・更新

ソフトウェアの管理・更新使用しているソフトウェアのパッケージを安定した最新のものに保つようにする必要があります。

ソフトウェアにはバグが付きもので、中にはセキュリティホールと呼ばれるようなものもあります。セキュリティホールが見付かると、開発者などから修正情報が (多くはパッチなどの形で) 発行されるとともに、問題点を修正した新しいバージョンがリリースされるのが一般的です。そうした情報は常にウォッチしておいてできる限り早く対応しなくてはなりません。その他、以下のような点に注意するとよいでしょう。

必要のないパッケージをインストールしない

出所のはっきりしないパッケージをインストールしない。

パッケージによっては、インストール前後に指定されたプログラムを実行するものがあります。出所のはっきりしないパッケージの中にはそうしたところに悪意のあるプログラムが仕込まれている可能性もあり、安易にインストールしてしまうことはおすすめできません。

アカウント管理

アカウントを発行することは、システムへの入口を作ることにつながります。

よって、アカウントの発行は慎重な管理の下で行われるべき作業と言えます。

- 必要のないアカウントを作らない
- 使われなくなったアカウントを放置しない
- テスト用のアカウントやデフォルトのアカウントに注意
- 基本となるパスワードは要注意、必要に応じてチェックツール (`john the ripper` など) を使用

4.8. root 権限の利用

可能な限り root 権限は使用しない。root 権限を不必要に多くの人に与えない事は重要です。Linux では pam を設定することにより、su コマンドで root になることの出来るグループを制限することもできます。以下は/etc/pam.d/su の該当行です。

/etc/pam.d/su

```
#Uncomment the following line to implicitly trust users in the "wheel" group.
#auth sufficient /lib/security/pam_wheel.so trust use_uid
# Uncomment the following line to require a user to be in the "wheel" group.
#auth required /lib/security/pam_wheel.so use_uid
```

sudo の活用

特定のユーザに、特定のコマンドだけ特権で利用することを許可する仕組みです。これにより、root のパスワードを教えることなく、限定したコマンドだけを root 権限での実行をさせることができます。

また-u オプションで実行時の権限を指定すれば、その権限においてコマンドを実行することもできます。

- root 権限で ls コマンドを実行

```
$ sudo ls
```

- ユーザ www-data の権限で ls コマンドを実行

```
$ sudo -u www-data ls
```

sudo を利用してコマンドを実行すると、そのログが syslog 経由で記録されるのも利点のひとつです。以下はユーザ dolphin が sudo apt-get update 実行時に残るログの例です。

```
  Jun 10 16:42:09 thehost sudo: dolphin : TTY=pts/2 ;
PWD=/home/dolphin ;  USER=root ; COMMAND=/usr/bin/apt-get update
```

visudo コマンドを使って/etc/sudoers ファイルを編集することにより、許可するユーザ/コマンドを設定します。

4.8.1. アクセス制限

アクセス制限の一般的なポリシー

「全てを許可しておいて何を禁止するか」ではなく「全てを禁止しておいて何を許可するか」を考える方が確実です。

OS のフィルタリング機能

フィルタリングはネットワーク間接続を行う場合、接続の窓口となる部分などで、ネットワーク間の通信を対象にした集中的に行ないます。

- IP アドレス、プロトコル、ポート番号、フラグなどで制御

どのホストがどういうサービスを別のホストに対して提供しているかということを常に把握しておく必要があります。あまりに制限の強すぎるフィルタは、本来、成立すべき通信までも遮断してしまうことにつながり問題となります。

NAT/NAPT とセキュリティ

フィルタリングとともに用いられるのが NAT や NAPT、あるいは IP マスカレードと呼ばれる機能です。IPv4 の本質的な問題であるとされる IP アドレスの割り当て数の少なさから、多くのサイトで用いられているアドレス変換技術です。

一般にフィルタリングと併用されますので、それよってセキュリティ面において一定の貢献をすることはありますが、本質的には NAT/NAPT を導入したからセキュリティが向上するということはありません。

ホスト間の信頼関係の確認

最後にネットワークに存在するホスト間の信頼関係について触れておきます。

- DHCP でだれでも接続できてしまわないか
- LAN ケーブルがあれば誰でも LAN に参加できてしまわないか
- 制限のない無線 LAN が運用されてしまっていないか
- 管理のあまいダイアルアップサーバはないか

万一、信頼しているホストに穴があると信頼関係を逆手にとられてしまう可能性があります。内部ネットワークであっても、本質的にその必要があるのであれば認証をかけるなどといった対策をとるべきでしょう。最終的には利便性の兼ね合いになってきますので、バランスをどこでとるかというのは難しい問題ですが、何らかの方針のもとに管理をする必要があると言えます。

TCP Wrappers

libwrap をリンクしたプログラムは/etc/hosts.allow と/etc/hosts.deny でアクセス制限の制御が可能です。例えば ssh、 sendmail、stunnel、rpc.mountd、portmap などで利用できます。また inetd から起動されるプログラムについては /usr/sbin/tcpd を介して起動するようにすることで、TCP Wrappers を使ったアクセス制限が可能となります。(アクセス制御ファイルの記述方法は「3.4.2 アクセス制御(TCP wrappers)」を参照)

アプリケーション独自対応

apache2 や proftpd など、独自のアクセス制限を備えたプログラムもあります。詳しくは各パッケージのドキュメントやマニュアルを参照してください。これらのアクセス制限の手法は排他的でないので、目的や状況に応じて組み合わせて利用することも可能です。

確認方法

フィルタリングやアクセス制御をした後の確認方法には、以下のコマンドなどが利用できます。

- nmap

nmap はポートスキャナです。自分の作った PC に対してポートスキャンを実施して不必要なポートが開いていないかを確かめられます。

- netstat
  netstat は接続状態や、経路テーブルなどの情報を表示するツールです。
  netstat -ta とすれば TCP の状態がわかります。LISTEN と表示されるものが開いているポートです。-ua とすれば UDP の状態がわかります。

- lsof
  lsof は list open file の略です。
  ポートで使ってるプログラムや、プロセスで利用しているファイルなどを表示することができます。不審なポートやプロセスがあればこのコマンドを利用して調査する事ができます。

```
$ lsof -i:22
$ lsof -p PID
```

- telnet
  telnet はリモートログインするためのサービスで利用されていましたが、今は ssh に置き換えられています。しかし、http サーバの状態確認など、デバッグ用に利用できます。

```
$ telnet localhost 80
 GET / HTTP/1.0
```

また SMTP サーバの転送状態を確認する事も比較的簡単に出来ます。

```
$ telnet localhost 25
 mail from: ryosuke@localhost
 rcpt to: ryosuke@example.jp
```

https の設定確認は openssl の s_client コマンドでも出来ます。

```
$ opnessl s_client -connect localhost:443
```

- Web ブラウザ
  http/ftp に telnet を使うのが面倒であればブラウザを使うのもいいでしょう。

4.8.2. ログの管理

syslog が作成するログファイルにはシステムの状態情報が記録されています。

ログは/var/log 以下に保存されます。

運用にもよりますが、/var を別パーティションにしている場合は、/var ファイルシステムのあふれに注意する必要があります。

以下は /var パーティションに関する注意です。

- インストール時に、/var を含むパーティションは、充分に大きくとっておく。
- logrotate を適切に設定しておくことにより一週間単位でログを回し、古いものを gzip 圧縮しておく。
- /var パーティションがあふれそうで緊急にファイルを削除しなければならない場合は、単純に削除するのではなく、必ず別ディスク、別ホストに移動すること。

4.8.3. ログの監視

システムの状態の把握や侵入を検知する為には、ログの監視が欠かせません。多くのサーバプログラムは syslog を利用してログを記録(apache など、一部独自の方法でログを記録するものもあります)。大量のログを全て管理人が逐一見ていくのは限界があるので、いわゆるログ監視ツールの活用が有効です。特定のパターンにマッチしたログ記録がある場合、メールなど指定した方法で報告してくれるツールを利用するのが一般的です。

swatch、 logcheck などはパッケージでも提供されているので適宜使用するとよいでしょう。

4.8.4. システム状態の監視

本来置き換わるはずのないシステムファイルが書き変わっている場合、侵入されている可能性が高いと考えられます。システムファイルの状態を監視し、何か変更を検知したら知らせる様なツールを定期的に実行し、監視しておくことも有用です。

例えば、代表的なシステム監視ツールである tripwire は、ファイル/ディレクトリのサイズや属性、タイムスタンプなどをデータベースに保存し、次回実行した際に変更があれば検出するというものです。同様のツールとして sxid というものもあります。

4.9. リモートメンテナンス

複数の医療機関を保守している場合は、さすがに毎回医療機関に伺うのは現実的ではない場合もあります。そのような場合に行われるリモートメンテナンスですが、こちらでも気をつけるべき項目があります。

VPN やリモートホストから医療機関に入るための「踏み台」のセキュリティは万全なのか。だれでもが踏み台を利用して医療機関に入れるなら、内部犯行による攻撃も起こりえます。また踏み台が外部からの攻撃で侵入された場合にも医療機関が危機にさらされるでしょう。

リモートログインのためのホストのセキュリティ確保と、そのホストが配置されているネットワークのセキュリティ確保をする事を勧めます。ここでは以下が重要と考えます。

- リモートメンテナンス実施者の確認体制の確立。
- リモートメンテナンス用ホストのセキュリティの確立と通常業務と隔離したネットワーク配置。

4.10. 情報処理の重要性

正しい運用をするためには正しくて鮮度の高い情報が不可欠です。とりわけセキュリティ分野はその傾向の強い分野で、日々問題点が発見されては解決され、解決されてはまた新しく発見されるということがくり返されています。

さいわい現在は情報が決定的に足りないという状況ではなく、求めさえすれば最新の情報が手に入ります。

しかし、逆に言えば情報があふれ返っている状況にあるとも言え、情報そのものの価値、信頼性などを判断し、整理する能力が必要とされるようになってきています。