Linux ディストリビューションのひとつ。
どちらかというと新しい技術にすぐ飛びつかず、数多くのハードに対応する。

debian fam
　http://debian.fam.cx/index.php?TopPage

2000 年頃のハードにインストールして使うなら debian がオヌヌメ。

インストール

ネットインストール用の CD から、なるべく最小の状態で debian をインストールし、足らない物を後から追加していくことにするので、ここでは netinst.iso を取得して CD に焼く。

Debian を入手するには
　http://www.debian.org/distrib/

下記の情報は debian-7.1 で確認したものなので、参考にする場合は注意のこと。

まずここで使うのは 2000 年頃のパソコンで、Pentium3 の 550MHz、512MB のメモリ、40GB のハードディスクという構成。

USB のキーボードを繋いで CD ブートしたら、netinst だとデバイスすら組み込んでくれないようで最初っから操作不能（汗）。
ps/2 の変換プラグが見つからないので泣く泣く古いキーボードを引っ張り出していざトライ。

まず起動画面から

```
Install
```

で進めて「日本語」を 3 ～ 4 つ連続で選ぶと追加コンポーネントのインストールが始まり、このコンポーネントで USB が認識するので、その後は USB キーボードも使えるようになる。

以降、ホスト名、ドメイン名、ネットワーク設定、ルートパスワード、ユーザー設定は割愛。
パーテーションも推奨のままでインストールを進める。

パーテーション設定後はベースシステムのインストールが開始され、タイムゾーン、リポジトリの選択～（略で大まかなソフトウェアの選択へ。

・SSH サーバ
・標準システムユーティリティ

だけを選んでインストールを進める。

コンソールで文字化け

インストール時に日本語を選んで進めると、コンソールでログインしたときに文字化けする。
調べたところによると、デスクトップ環境をインストールしてれば日本語表示に必要なパッケージがインストールされるらしい。

現在の状態は

```
# echo $LANG
ja_JP.UTF-8
```

以前は ja_JP.EUC-JP がデフォルトだった気がするが、debian-7.1 では ja_JP.UTF-8 となった模様。

それはさておき、どうしてもコンソールで日本語表示したければ

コンソールで日本語の表示をするには
http://debian.fam.cx/index.php?Japanese#x1589390

必要無ければ英語で再設定する。

```
# LANG=C
# dpkg-reconfigure locales
```

で en_US.UTF-8 を選択、ja_JP.UTF-8 を解除。次の画面で en_US.UTF-8 を選択する。

一旦 exit してログインし直し

```
# echo $LANG
en_US.UTF-8
```

になれば英語切替完了。

dpkg-reconfigure を使わずに手動設定するなら

Locales in Debian
http://people.debian.org/~schultmc/locales.html

IP の設定

インストール時はとりあえず DHCP で進めたので、固定 IP を割り振る。

現在の状態は

```
# ifconfig
```

設定を変更するには

```
# vi /etc/network/interfaces
```

で対象の eth を編集する。

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug eth0
#iface eth0 inet dhcp
auto eth0
iface eth0 inet static
address 192.168.0.xx
netmask 255.255.255.0
gateway 192.168.0.1
```

終わったら再起動。

```
# /etc/init.d/networking restart
```

以降は LAN 経由で SSH クライアント（puttyjp）から操作する。

IPv6 の無効化

sysctl に設定を追記。

```
# vim /etc/sysctl.conf
net.ipv6.conf.all.disable_ipv6 = 1
```

設定をリロード。

```
# sysctl -p
```

リロードは良いがリブートでダメな場合。
raspbian（wheezy）で再起動時に sysctl.conf が効かない（sysctl の設定が無視される）症状に出くわした。

After reboot debian box ignore sysctl.conf values
http://serverfault.com/questions/355520/after-reboot-debian-box-ignore-sysctl-conf-values

とりあえず例にならって

```
# /etc/rc.local
/etc/init.d/procps restart

exit 0
```

とすることで再設定。

SSH クライアントから接続

debian は初期設定でファイアーウォール（iptables）が無いので、puttyjp 等の SSH クライアントより、先に設定した IP アドレスに接続できるはず（多分）。

user の追加と削除

```
# adduser username
または
# useradd -m username

# deluser --remove-home username
```

.bashrc が読み込まれない

作ったユーザーで .bashrc が読み込まれないのでおかしいなぁと思ったら

```
$ echo $SHELL
/bin/sh
```

sh だったでござる（汗）。

```
$ chsh
Password:
Changing the login shell for ryusendo
Enter the new value, or press ENTER for the default
        Login Shell [/bin/sh]: /bin/bash
$ exit
```

でログインし直せば読まれるはず。

SSH の設定

root のログインを禁止し、ログイン可能なユーザーを限定（例では hogehoge のみ）にする。

```
# vim /etc/ssh/sshd_config

PermitRootLogin no
AllowUsers hogehoge
```

hogehoge でログインしてから su か sudo で root の作業を行うことになる。

sudo の設定

sudo がインストールされていない場合。

```
# apt-get install sudo
```

設定には visudo コマンドを使用する。

```
# visudo

Defaults        rootpw
hogehoge        ALL=(ALL:ALL) ALL
```

上記は hogehoge ユーザーが sudo で root 権限を持ち、パスワードは root のものを要求する。

```
ユーザー        ホスト =( 権限 ) コマンド
user            host=(user:group) command
%group...
```

みたいな感じ。

dpkg と apt-get と aptitude

最小構成の状態で

```
# free -m
             total       used       free
Mem:           502         46        456


# df -h
Filesystem    Size  Used Avail Use%
rootfs         36G  791M   34G   3%
```

みたいな状態。

ここに使いたいパッケージを随時インストールしていくのだが、インストールコマンドは以下の 3 つ。

・dpkg…パッケージ管理（依存関係はチェックしない）
・apt-get…dpkg を操作（依存関係をチェックする）
・aptitude…apt-get を操作（依存関係をチェックする）

第 2 章 Debian パッケージ管理
http://www.debian.org/doc/manuals/debian-reference/ch02.ja.html

```
# aptitude
```

とすると管理画面みたいなのが出てきて、キーボードでパッケージを選択して操作できるが、コマンドラインの apt-get と aptitude とは別のパッケージ管理（ログ？）を持っている模様。
これは使わない方が無難。

コマンドラインで

```
# apt-get install <package>
# aptitude remove <package>
# aptitude install <package>
# apt-get autoremove <package>
```

という感じに使っている分には大丈夫そう。

今後はコマンドラインから

```
# aptitude install <package>
```

でインストールを進める。

とりあえず apt-get と aptitude で update しておく。

```
# apt-get update
# aptitude update
```

下記は apt-get で玄人っぽく操作する場合のコマンドを少々（※ debian-7.1）。

パッケージ一覧を更新する。

```
# apt-get update
```

更新されたパッケージ一覧から <package> を検索。

```
# apt-cache search <package>
```

<package> をインストール。

```
# apt-get install <package>
```

インストールしたパッケージの更新。

```
# apt-get upgrade
```

<package> のみをアンインストール。

```
# apt-get remove <package>
```

<package> の依存関係をチェックしてアンインストール。<package> を指定しなかった場合は、どこからも参照されなくなったパッケージをすべて削除。

```
# apt-get autoremove <package>
# aptitude remove <package> ←同等
```

<package> のアンインストール時に設定ファイルも削除。

```
# apt-get purge <package>
```

ファイル一覧をカラーで表示させる

ls コマンドを使用した時、Fedora とかではディレクトリとファイルがカラーで区別されて見やすいが、debian は色が付かない。

ホームディレクトリの .bashrc にコマンドがあるのでコメントアウトする。

```
# You may uncomment the following lines if you want `ls' to be colorized:
export LS_OPTIONS='--color=auto'
eval "`dircolors`"
alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -lA'
```

vim で色が付かない

debian の最小構成でも vi が入っているが、色が付かないと見づらいので vim を入れた。

```
# aptitude install vim
```

インストールされたパッケージは下記のとおり。

- vim
- vim-runtime{a}

それでも色が付かなかったので調べたら、デフォルトでは色が付かない設定らしい（汗）。

/etc/vim/vimrc の syntax 行のコメントアウトを外す

```
syntax on
```

もしくはユーザー単位で設定したいなら

```
# cp /etc/vim/vimrc /home/user/.vimrc
```

のようにコピーしておいて syntax を有効にする。

iptables の設定

初期状態で iptables の設定が無い模様（iptables 自体はインストールされている）。

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

とりあえず必要なパッケージをインストール。

```
# aptitude install iptables-persistent
```

インストールされたパッケージは下記のとおり。

- iptables-persistent

/etc/iptables に設定ファイルができるので ip4 用の rules.v4 を編集。

内容は

iptables - Debian Wiki
https://wiki.debian.org/iptables

をパクる。
一時ファイル rules を作成。

```
# cd /etc/iptables
# vim rules
```

内容は下記のとおり。

```
*filter

# Allows all loopback (lo0) traffic and drop all traffic to 127/8 that doesn't use lo0
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -d 127.0.0.0/8 -j REJECT

# Accepts all established inbound connections
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allows all outbound traffic
# You could modify this to only allow certain traffic
-A OUTPUT -j ACCEPT

# Allows HTTP and HTTPS connections from anywhere (the normal ports for websites)
-A INPUT -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
-A INPUT -p tcp --dport 8080 -j ACCEPT

# Allows SSH connections
# THE -dport NUMBER IS THE SAME ONE YOU SET UP IN THE SSHD_CONFIG FILE
-A INPUT -p tcp -m state --state NEW --dport 30000 -j ACCEPT

# Now you should read up on iptables rules and consider whether ssh access
# for everyone is really desired. Most likely you will only allow access from certain IPs.

# Allow ping
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

# log iptables denied calls (access via 'dmesg' command)
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7

# Reject all other inbound - default deny unless explicitly allowed policy:
-A INPUT -j REJECT
-A FORWARD -j REJECT

COMMIT
```

保存したら iptables に読み込ませる。

```
# iptables-restore < rules
```

読み込んだ状態を確認。

```
# iptables -L
```

iptables-persistent で読み込まれる rules.v4 に設定を書き戻す。

```
# iptables-save > rules.v4
```

再起動。

```
# /etc/init.d/iptables-persistent restart
```

ちなみに iptables-persistentっていうのは、/etc/iptables/rules.v4 と rules.v6 を読み込む起動スクリプト。

あとは自分の環境に合わせて設定を行う。
※ debian8 では起動スクリプトが netfilter-persistent になった模様。

```
# /etc/init.d/netfilter-persistent restart
```

cron と ddns 設定

VALUE DOMAIN の DDNS で自鯖の IP アドレスを更新するコマンド。

```
$ wget http://dyn.value-domain.com/cgi-bin/dyn.fcg?d=[domain]&p=[password]&h=*
```

h=* の部分は value domain 側の設定と合わせること。

cron の設定は

```
# vim /etc/crontab
*/15 * * * * wget -O - 'http://dyn.value-domain.com/cgi-bin/dyn.fcg?d=sample.com&p=password&h=*' >
/dev/null 2>&1
```

初期状態で cron ログが無効になっているのでコメントアウトを外す。

```
# vim /etc/rsyslog.conf
cron.*      /var/log/cron.log
```

```
# /etc/init.d/rsyslog restart
```

apache のインストール

```
# aptitude install apache2
```

とやると apache2-mpm-worker をインストールしようとするので

```
# aptitude install apache2-mpm-prefork
```

と明示的に指示する必要がある（PHP5/libapache2-mod-php5 が prefork に依存しているため）。
インストールされたパッケージは下記のとおり。

・apache2-mpm-prefork
・apache2-utils{a}
・apache2.2-bin{a}
・apache2.2-common{a}
・libapr1{a} libaprutil1{a}
・libaprutil1-dbd-sqlite3{a}
・libaprutil1-ldap{a}
・ssl-cert{a}

debian の apache2 に ssl 接続できない

こんなエラーで接続不能（汗）。

```
SSL received a record that exceeded the maximum permissible length.
(Error code: ssl_error_rx_record_too_long)
```

ネット検索すると「VirtualHost にきちんとポート設定していないからだ」みたいな解説が多いが、debian-7.1 のデフォルトだと /etc/apache2/ports.conf に

```
NameVirtualHost *:80
Listen 80

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will also have to change
    # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
    # to <VirtualHost *:443>
    # Server Name Indication for SSL named virtual hosts is currently not
    # supported by MSIE on Windows XP.
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

/etc/sites-available/default と default-ssl に

```
<VirtualHost *:80>
        ServerAdmin ...


<IfModule mod_ssl.c>
<VirtualHost _default_:443>
        ServerAdmin ...
```

となっている。
で、いろいろ調べたら default-ssl が有効になっていなかった（汗）。
/etc/apache2/sites-enabled に default に対するシンボリックリンクがあり、これで初めて有効になる模様。デフォルトでは default-ssl のシンボリックリンクが無かった。

```
# a2ensite
Your choices are: default default-ssl
Which site(s) do you want to enable (wildcards ok)?
default default-ssl
Site default already enabled
Site default-ssl already enabled
```

同様に mods-available から ssl を有効にするには

```
# a2enmod ssl
```

とすると、mods-available の ssl.conf と ssl.load に対するシンボリックリンクが mods-enabled に作成される。

要するに debian の apache では

```
*a2enmod、a2dismod
*a2ensite、a2dissite
```

で設定するらしい。

設定後に再起動を忘れずに。

```
# /etc/init.d/apache2 restart
```

それにしても debian の apache は設定ファイルが変？に分割されててわかりづらい（汗）。

PHP のインストール

とりあえず最小構成でインストールしておく。

```
# aptitude install php5
```

インストールされたパッケージは下記のとおり。

- libapache2-mod-php5{a}
- libonig2{a}
- libqdbm14{a}
- php5
- php5-cli{a}
- php5-common{a}

勝手にリロードしてくれるので、/var/www/phpinfo.php を作成し、動作を確認する。

```
# vim /var/www/phpinfo.php
```

```
<?php
phpinfo();
```

LAN 上のクライアント（ブラウザ）から

```
http://192.168.0.xx/phpinfo.php
```

にアクセスして phpinfo が表示されれば OK。
ちなみに debian-7.1 の PHP は

```
# php -v
PHP 5.4.4-14+deb7u4 (cli) (built: Aug 26 2013 07:40:51)
```

mysql のインストール

```
# aptitude install mysql-server-5.5
```

・libaio1{a}
・libdbd-mysql-perl{a}
・libdbi-perl{a}
・libhtml-template-perl{a}
・libmysqlclient18{a}
・libnet-daemon-perl{a}
・libplrpc-perl{a}
・mysql-client-5.5{a}
・mysql-common{a}
・mysql-server-5.5
・mysql-server-core-5.5{a}

これだけだと PHP ～ MySQL が利用できないので

```
# aptitude install php5-mysql
```

とすると、php-mysql、php-mysqli、php-pdo_mysql が利用できるようになる。

・ php5-mysql

ちなみに debian-7.1 の MySQL5 は

```
mysql > status
Server version:         5.5.31-0+wheezy1 (Debian)
```

my.cnf は

```
/etc/mysql/my.cnf
```

my.cnf のサンプルは

```
/usr/share/doc/mysql-server-5.5/examples
```

あたりにインストールされる。small 以外は gz 圧縮されているので解凍のこと。

```
# gzip -d my-medium.cnf.gz
```

tomcat のインストール

```
# aptitude install tomcat7
```

・authbind{a}

・ca-certificates-java{a}
・dbus{a}
・default-jre-headless{a}
・icedtea-6-jre-cacao{a}
・icedtea-6-jre-jamvm{a}
・java-common{a}
・libavahi-client3{a}
・libavahi-common-data{a}
・libavahi-common3{a}
・libcommons-dbcp-java{a}
・libcommons-pool-java{a}
・libcups2{a}
・libdbus-1-3{a}
・libecj-java{a}
・libgeronimo-jta-1.1-spec-java{a}
・libjpeg8{a}
・libnspr4{a}
・libnss3{a}
・libnss3-1d{a}
・libpcsclite1{a}
・libservlet3.0-java{a}
・libsystemd-login0{a}
・libtomcat7-java{a}
・openjdk-6-jre-headless{a}
・openjdk-6-jre-lib{a}
・tomcat7
・tomcat7-common{a}
・tzdata-java{a}

```
http://192.168.0.xx:8080/
```

にアクセスして It Works! の画面が表示されれば OK。

solr のインストール

検索エンジンの solr をインストール。

```
# aptitude install solr-tomcat
```

・curl{a}
・fontconfig{a}
・hicolor-icon-theme{a}
・icedtea-netx{a}
・icedtea-netx-common{a}
・javascript-common{a}
・libapache-mime4j-java{a}
・libapache-pom-java{a}
・libasound2{a}
・libasyncns0{a}
・libatk-wrapper-java{a}
・libatk-wrapper-java-jni{a}
・libatk1.0-0{a}
・libatk1.0-data{a}

・libavalon-framework-java{a}
・libcairo2{a}
・libcommons-beanutils-java{a}
・libcommons-codec-java{a}
・libcommons-collections3-java{a}
・libcommons-compress-java{a}
・libcommons-digester-java{a}
・libcommons-fileupload-java{a}
・libcommons-httpclient-java{a}
・libcommons-io-java{a}
・libcommons-lang-java{a}
・libcommons-logging-java{a}
・libcommons-parent-java{a}
・libcurl3{a}
・libdatrie1{a}
・libexcalibur-logkit-java{a}
・libffi5{a}
・libflac8{a}
・libgdk-pixbuf2.0-0{a}
・libgdk-pixbuf2.0-common{a}
・libgeronimo-stax-1.2-spec-java{a}
・libgif4{a}
・libglib2.0-0{a}
・libglib2.0-data{a}
・libgtk2.0-0{a}
・libgtk2.0-bin{a}
・libgtk2.0-common{a}
・libguava-java{a}
・libhttpclient-java{a}
・libhttpcore-java{a}
・libhttpmime-java{a}
・libice6{a}
・libicu4j-4.4-java{a}
・libjasper1{a}
・libjaxp1.3-java{a}
・libjbig0
・libjs-jquery{a}
・libjson0{a}
・libjsr305-java{a}
・libknopflerfish-osgi-framework-java{a}
・liblog4j1.2-java{a}
・liblucene3-contrib-java{a}
・liblucene3-java{a}
・libmsv-java{a}
・libogg0{a}
・libpango1.0-0{a}
・libpixman-1-0{a}
・libpng12-0{a}
・libportlet-api-2.0-spec-java{a}
・libpulse0{a}
・libregexp-java{a}
・librelaxng-datatype-java{a}
・librtmp0{a}
・libservlet2.5-java{a}
・libslf4j-java{a}
・libsm6{a}

・libsndfile1{a}
・libsolr-java{a}
・libssh2-1{a}
・libstax-java{a}
・libstax2-api-java{a}
・libthai-data{a}
・libthai0{a}
・libtiff4{a}
・libtomcat6-java{a}
・libvorbis0a{a}
・libvorbisenc2{a}
・libwoodstox-java{a}
・libx11-xcb1{a}
・libxcb-render0{a}
・libxcb-shm0{a}
・libxcomposite1{a}
・libxcursor1{a}
・libxdamage1{a}
・libxerces2-java{a}
・libxfixes3{a}
・libxft2{a}
・libxi6{a}
・libxinerama1{a}
・libxml-commons-external-java{a}
・libxml-commons-resolver1.1-java{a}
・libxrandr2{a}
・libxrender1{a}
・libxtst6{a}
・libxz-java{a}
・openjdk-6-jre{a}
・shared-mime-info{a}
・solr-common{a}
・solr-tomcat
・tomcat6{a}
・tomcat6-common{a}
・ttf-dejavu-extra{a}
・wwwconfig-common{a}

どうやら tomcat6 の solr だったようで、以下は

```
     Remove the following packages:
1)     libtomcat7-java
2)     tomcat7
3)     tomcat7-common

Accept this solution? [Y/n/q/?] q
```

だそうな（汗）。
でも処理を進めたら 4 つ消えたけど良いの？

・libservlet3.0-java{u}
・libtomcat7-java{a}
・tomcat7{a}
・tomcat7-common{a}

動作確認は

```
http://192.168.0.xx:8080/solr/
```

Welcome to Solr! 画面が表示されれば OK。
Solr Admin から Info で確認すると

```
Solr Specification Version: 3.6.0.2012.06.21.14.28.59
```

solr4 が使いたい場合は自分で入れるしかなさそうです（汗）。
ちなみにダウングレードされた tomcat のバージョンは

```
cd /usr/share/tomcat6/bin
./version.sh

Server version: Apache Tomcat/6.0.35
```

でした。
ついでに tomcat の動作確認もかねて jsp を作成。

```
# cd /var/lib/tomcat6/webapps/ROOT
vim version.jsp
```

プログラムは下な感じ。

```
<%
out.println(application.getServerInfo());
%>
```

ブラウザからアクセスして表示されてば OK。

```
http://192.168.0.xx:8080/version.jsp

Apache Tomcat/6.0.35
```

postfix のインストール

debian では既に exim4 という smtp サーバーが入っている模様。
設定とかわからないので、慣れた postfix に変更する。

```
# aptitude install postfix
```

・postfix{b}

競合するということで、下記が削除される。

・exim4
・exim4-base
・exim4-config
・exim4-daemon-light

で、設定ファイルとかも削除されたのか疑問だったので

```
# aptitude purge exim4
```

を実行。その後 find で確認してみたら

```
# find / -name "exim*"
/var/lib/dpkg/info/exim4-config.postrm
/var/lib/dpkg/info/exim4-config.list
/var/lib/dpkg/info/exim4-daemon-light.postrm
/var/lib/dpkg/info/exim4-base.postrm
/var/lib/dpkg/info/exim4-base.list
/var/lib/dpkg/info/exim4-daemon-light.list
/var/lib/exim4
/var/spool/exim4
/var/log/exim4
/usr/share/wwwconfig-common/exim-trust.sh
/usr/share/vim/vim73/syntax/exim.vim
/etc/default/exim4
/etc/logrotate.d/exim4-paniclog
/etc/logrotate.d/exim4-base
/etc/ppp/ip-up.d/exim4
/etc/init.d/exim4
/etc/exim4
/etc/exim4/exim4.conf.template
/etc/cron.daily/exim4-base
/run/exim4
/run/exim4/exim.pid
```

残骸がいっぱい残ってるみたい（汗）。
横着せずに全て purge してみる。

```
# aptitude purge exim4 exim4-base exim4-config exim4-daemon-light

# find / -name "exim*"
/usr/share/wwwconfig-common/exim-trust.sh
/usr/share/vim/vim73/syntax/exim.vim
```

とりあえずほぼ消えたっぽいので OK とする。

パッケージ管理

apt でダウンロードされたパッケージの保存先。

```
/var/cache/apt/archives
```

任意のパッケージがインストール済みかどうか調べる。

```
$ dpkg -l | grep mysql-common
ii  mysql-common                5.5.44-0+deb8u1             all          MySQL database common
files, e.g. /etc/mysql/my.cnf
```

パッケージのインストール先を調べる。

```
$ dpkg -L libshout3-dev
```

パッケージの情報を表示。

```
$ dpkg -s libshout3-dev
```

apt-cache

solr というワードが含まれたパッケージを検索する。

```
$ apt-cache search solr
chef-solr - manager for search indexes of Chef node attributes using Solr
dovecot-solr - Solr full text search support for Dovecot
libsolr-java - Enterprise search server based on Lucene - Java libraries
libwebservice-solr-perl - Perl interface for the Solr (Lucene) web service
python-pysolr - lightweight Python wrapper for quering Apache Solr
solr-common - Enterprise search server based on Lucene3 - common files
solr-jetty - Enterprise search server based on Lucene3 - Jetty integration
solr-tomcat - Enterprise search server based on Lucene3 - Tomcat integration
```

パッケージのバージョンを調べる。

```
$ apt-cache policy solr-tomcat
solr-tomcat:
  Installed: (none)
  Candidate: 3.6.0+dfsg-1+deb7u1
  Version table:
     3.6.0+dfsg-1+deb7u1 0
        500 http://ftp.jaist.ac.jp/raspbian/ wheezy/main armhf Packages
        500 http://mirrordirector.raspbian.org/raspbian/ wheezy/main armhf Packages
```

パッケージの詳細を調べる。

```
$ apt-cache show solr-tomcat
Package: solr-tomcat
Source: lucene-solr
Version: 3.6.0+dfsg-1+deb7u1
Installed-Size: 63
Maintainer: Debian Java Maintainers <pkg-java-maintainers@lists.alioth.debian.org>
Architecture: all
Depends: solr-common (= 3.6.0+dfsg-1+deb7u1), tomcat6
Conflicts: solr-jetty, solr-tomcat6
:
```

ライブラリの中の関数一覧を確認する

```
$ nm libhoge.so
```

目的の関数名でフィルタ

```
$ nm libhoge.so | grep some_function_name
```

任意のディレクトリの使用容量を調べる

```
$ du -m /path/to
```

オプション m で MB。

ポートスキャン

対象ホスト（または IP）のポート（例では 10-100）をスキャンする。

```
# netcat -z -v my.domain.jp 10-100
my.domain.jp [xxx.xxx.xxx.xxx] 80 (http) open
my.domain.jp [xxx.xxx.xxx.xxx] 53 (domain) open
my.domain.jp [xxx.xxx.xxx.xxx] 22 (ssh) open
my.domain.jp [xxx.xxx.xxx.xxx] 21 (ftp) open
```

netcat コマンドは nc（短縮）も可。
インストールされていないようなら

```
# apt-get install netcat
```

通常版？の traditional がインストールされる。
高機能版？の bsd 版もある模様。

netcat で簡易クライアント

```
# echo -e "GET /path/to/file.php HTTP/1.0\r\n" | nc my.domain.jp 80
```

とかすればレスポンスが返ってくる。

USB メモリを使う

USB ポートに USB メモリを差し込んだら fdisk で確認。

```
$ fdisk -l

   :
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *          47      507391      253672+   6  FAT16
```

/dev/sda1 として認識したようなので /mnt にマウント。マウント先のディレクトリを作成しておく。

```
# mkdir /mnt/usbmem
# mount -t vfat /dev/sda1 /mnt/usbmem
```

あとは /mnt/usbmem に対して cp したり mv したり rm したり ... みたいな。

NTFS でフォーマットされている場合は

```
# mount -t ntfs /dev/sda1 /media/usbmem
```

しかし権限（パーミッション）の設定が生きてるようで、書き込みできなかった（汗）。

外すときは

```
# umount /mnt/usbmem
```

Windows で入力した mp3 のタグを UTF-8 に変換する

Windows でタグを入力すると文字コードが sjis になるため、Linux では文字化けする。
Linux 側で一括変換するのに mid3iconv を使うには、python-mutagen をインストール。

```
$ sudo apt-get install python-mutagen
```

その後、mp3 を置いているディレクトリで

```
$ mid3iconv -e SHIFT_JIS -d *.mp3
```

変換結果が -d オプションにより画面上に出力されるため、問題なく表示されれば OK。

Youtube やニコニコ動画をダウンロードする

youtube-dlってのを使うとコマンドラインでダウンロードできる。

```
# apt-get install youtube-dl
```

オプションとか詳細は --help で確認のこと。

Youtube 動画から音楽だけ抽出して mp3 で保存するには

```
$ youtube-dl -A -x --audio-format mp3 https://www.youtube.com/watch?v=nr688jE72mA
```

ニコ動からの場合は -u でメールアドレス、-p でパスワードを設定。

```
$ youtube-dl -u mail -p password -A -x --audio-format mp3 http://www.nicovideo.jp/watch/sm23144598
```

mp3 に変換するのに結構時間がかかる。