2. Linux のインストールとパッケージ管理

2.1 ハードディスクのレイアウト設計

2.1.1 Linux インストールに必要なパーティション

ハードディスク，SSD は，**パーティション** という区画に分割して使うことができる．
Linux をインストールするためには，少なくとも次の2つのパーティションが必要になる．

- ルートファイルシステムに割り当てるルートパーティション
- スワップ領域

実際には，さらにいくつかのパーティションに分割する．

- 柔軟なシステム管理のため
- ディスクに障害が発生したときの被害を抑えるため
- 障害発生時にスムーズな復旧作業をするため

以下に示すディレクトリ群は，独立したパーティションに割り当てるのが一般的．

- `/home`
  - 一般ユーザが利用するファイルが格納される．
  - 多数のユーザが利用するシステムの場合は，専用のパーティションに分割する必要がある．
- `/var`
  - 各種ログファイルなど，更新頻度の高いファイルが格納される．
  - ログが大量に生成されログファイルが肥大化すると，ファイルシステムの容量を超えてしまう可能性がある．
    - `/var` を別パーティションにすることで，被害がシステム全体に及ぶ危険を減らすことができる．
- `/usr`
  - プログラムやライブラリ，ドキュメントが置かれる．
    - `/usr` を別パーティションにしていると，システムが正常に起動できなくなる場合がある．
  - NFS を使ってコマンドやプログラムを共有する場合は， `/usr` を読み込み専用でマウントしておくことでセキュリティを高めることができる．
- `/boot`

    - システムによっては，ディスクの先頭パーティションとして数百MB程度を `/boot` パーティションに割り当てたほうが良い場合がある．
        - e.g. RAID (複数のハードディスクを組み合わせる使用方法) を利用する場合，内蔵パーティションに `/boot` パーティションが必要とされることがある．
- EFI システムパーティション (ESP)
    - UEFI を使ったシステムでは，FAT ファイルシステムでフォーマットされた ESP が必要．
    - OS のブートローダやデバイスドライバなどが格納される．
- スワップ領域
    - 仮想メモリ領域として利用される．
        - 物理メモリが不足した場合に，ディスクの一部を一時的にメモリの延長として使うことができるようにする機能．
    - サイズの目安は，搭載されている物理メモリの 1 - 2 倍．
        - 十分な物理メモリを搭載していれば，スワップ領域を使わないようにすることもできる．
- `/` (ルート)
    - 上記以外は，ルートファイルシステム (`/` ディレクトリが格納されたパーティション) になる．
    - ファイルシステムに障害が発生したときの復旧を容易にするためにも，できるだけ小さくしたほうが良い

2.1.2 パーティションのレイアウト設計

ディスクのパーティションレイアウトを設計する場合に考慮すべき点:

- システムの用途
- ディスクの容量
- バックアップの方法

最近は，LVM (論理ボリューム管理) を使った柔軟なディスク管理が行われるようになってきている．

- LVM
    - パーティション上にそのままファイルシステムを作成するのではなく，ボリュームグループという仮想ディスクを作成し，その上に仮想的なパーティションを作成する仕組み．

2.2 ブートローダのインストール

- **ブートローダ**
  - ハードディスクなどのストレージからOSを読み込んで起動するプログラム．
  - 代表的な Linux のブートローダは，GRUB．
    - バージョン0.9x系のGRUB Legacyと，バージョン1.9x系のGRUB2がある．

2.2.1 GRUB のインストール

GRUB の特徴

- 多数のファイルシステムを認識可能
- シェル機能を搭載し，コマンドによる高度な管理が可能

ブートローダとして GRUB をインストールするには， `grub-install` コマンドを実行する．

例: `/dev/sda` の MBR 領域に GRUB をインストール:

```
grub-install /dev/sda
```

2.2.2 GRUB Legacy の設定

GRUB Legacy の設定ファイルは， `/boot/grub/menu.lst` である．
`/boot/grub/menu.lst` の設定パラメータ:

パラメータ	説明
`timeout`	メニューを表示している時間 (秒)
`default`	デフォルトで起動するエントリの番号
`title`	メニューに表示されるエントリ名
`root`	ルートデバイスの指定
`kernel`	起動するカーネルイメージファイルと起動オプションの指定
`makeactive`	ルートパーティションをアクティブ化
`chainloader`	指定されたセクタの読み込みと実行
`hiddenmenu`	起動時に選択メニューを表示しない

2.2.3 GRUB 2 の設定

GRUB 2 の設定ファイルは， `/boot/grub2/grub.cfg` であるが，GRUB Legacy とは異なり，直接ファイルを編集することはしない．
`/etc/default/grub` で設定を行い， `grub2-mkconfig` コマンドを実行すると，設定に基づき `/boot/grub2/grub.cfg` が生成される．

`/etc/default/grub` の例:

```
[root@localhost default]# cat grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto spectre_v2=retpoline rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rh
GRUB_DISABLE_RECOVERY="true"
```

注意: `=` の前後にスペースを入れない．

`/etc/default/grub` の主な設定パラメータ:

パラメータ	説明
`GRUB_TIMEOUT`	起動メニューがタイムアウトするまでの秒数
`GRUB_DEFAULT`	起動メニューがタイムアウトしたときに， デフォルトOSとして選択されるエントリ (saved: 保存された選択肢)
`GRUB_CMDLINE_LINUX`	カーネルに渡される起動オプション

設定後は， `grub2-mkconfig` コマンドを使って， `boot/grub2/grub.cfg` を生成する．

```
[root@localhost default]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-1160.45.1.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-1160.45.1.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10.0-1160.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-1160.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-5eeefb94cf964b18b1565579659d5df0
Found initrd image: /boot/initramfs-0-rescue-5eeefb94cf964b18b1565579659d5df0.img
done
```

2.2.4 ブートオプションの指定

ブードローダ起動時に，システムの動作を指定するためのブートオプションを指定できる．GRUB でブートオプションを指定するためには，起動時の画面で E キーを押す．
すると，次のような画面が表示される．

```
grub append > ro root=/dev/VolGroup00/LogVol00 rhgb quiet
```

ここでキーボードからオプションを入力できるようになる．

主なブートオプション:

パラメータ	説明
`root=デバイス`	ルートパーティションとしてマウントするデバイス
`nousb`	USBデバイスを使用しない
`single`	シングルユーザモードで起動する
`1 ~ 5`	指定したランレベルで起動する

注意: `/` ディレクトリが格納されたパーティションをルートパーティションという．

シングルユーザモードで起動したい場合は，ブートオプションとして `single` を追加する．

```
grub append > ro root=/dev/VolGroup00/LogVol00 rhgb quiet single
```

入力後に Enter を押すと，指定したパラメータが適用されてシステムが起動する．

2.3 共有ライブラリ管理

- **ライブラリ**
  - よく使われる機能をまとめ，他のプログラムから利用できるようにしたもの．
  - **静的ライブラリ**
    - プログラムの作成時にその実行ファイル内に組み込まれるライブラリ
  - **共有ライブラリ**
    - プログラムの実行時にロードされ，複数のプログラム間で共有されるライブラリ

2.3.1 スタティックリンクとダイナミックリンク

- **リンク**
  - プログラム本体からライブラリの機能を利用
  - **スタティックリンク**
    - コンパイルをする時点で，コンパイラがライブラリを実行ファイルに埋め込む．
    - 実行ファイル内にライブラリの機能が埋め込まれるということは，よく使われるライブラリの機能が，さまざまな実行ファイルに重複して入ってしまうことになる．
  - **ダイナミックリンク**
    - 実行ファイルへライブラリを埋め込むことはせず，実行時にライブラリの機能を呼び出す方法．
- **共有ライブラリ**
  - ダイナミックリンクによって呼び出されるライブラリ．
  - 通常， `/lib` あるいは `/usr/lib` ディレクトリに配置されている．
  - 共有ライブラリは， `lib*.so*` という名前が付けられている．
  - 共有ライブラリは通常， `/lib` あるいは `/usr/lib` ディレクトリに配置されている．

2.3.2 必要な共有ライブラリの確認

実行ファイルが必要としている共有ライブラリは， `ldd` コマンドで調べることができる．

`cat` コマンドが必要とする共有ライブラリ:

```
[root@localhost ~]# ldd /bin/cat
        linux-vdso.so.1 =>  (0x00007ffe38b88000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f6b7b21a000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f6b7b5e8000)
```

- プログラムの実行時には， `ld.so` リンカおよびローダが実行時にリンクする共有ライブラリを検索して，必要なライブラリをロードする．
  - `/lib` ， `/usr/lib` ディレクトリ以外のライブラリも検索する場合は，そのリストを `/etc/ld.so.conf` ファイルに記述しておく．
    - `/etc/ld.so.conf.d` ディレクトリ以下に複数の設定ファイルを配置し， `/etc/ld.so.conf` でそれらを読み込んでいる場合もある．
  - プログラムを実行する度にこれらのディレクトリを検索するのは非効率なので，実際にはバイナリのキャッシュファイルである `/etc/ld.so.cache` が参照される．
  - 共有ライブラリを変更した場合は， `ldconfig` コマンドを実行してキャッシュを更新する必要がある．
    - `ldconfig` コマンドは， `/etc/ld.so.conf` ファイルに基づき， `/etc/ld.so.cache` を再構築する．
  - その他のディレクトリも検索対象に加えたい場合は，環境変数 `LD_LIBRARY_PATH` にディレクトリリストを記述する．
  - `ld.so` リンカが共有ライブラリを検索する順序:
    1. 環境変数 `LD_LIBRARY_PATH`
    2. キャッシュファイル `/etc/ld.so.cache`
    3. デフォルトのパス `/lib` ， `usr/lib`

2.4 Debian パッケージの管理

- **パッケージ**
  - 実行プログラム，設定ファイル，ドキュメントなどを1つのファイルにまとめたもの．
  - パッケージ管理の方法はディストリビューションによって異なる．
    - Debian 形式
      - Debian GNU/Linux などで採用．
    - RPM 形式
      - Red Hat Enterprise Linux などで採用．

2.4.1 パッケージ管理とは

- **パッケージ管理システム**
  - パッケージのインストールやアンインストール，アップデート作業において，どのようなパッケージがどこにインストールされているのかを管理したり，パッケージ間の競合を回避したりする仕組みを提供．
  - インストール，アンインストール，アップデート作業を容易にする．
  - パッケージ管理システムは，依存関係や競合関係を監視し，依存関係や競合関係を損なうようなインストールやアンインストールに警告を発する．
- **パッケージの依存関係**
  - あるパッケージが別のパッケージに依存しているというような関係．
- **パッケージの競合関係**
  - パッケージ間で互いにぶつかり合うこと．
- Linux のパッケージ管理
  - **Debian 形式**
    - Debian 系のディストリビューションで利用されている．
    - パッケージ管理には， `dpkg` コマンドや APT ツールなどが使われる．
  - **RPM 形式**
    - Red Hat 系ディストリビューションを中心に利用されている．
    - パッケージ管理には， `rpm` コマンドが使われる．
  - Debian 形式と RPM 形式には互換性がない．
  - パッケージ管理システムを使って，コンパイル済の状態で配布されるバイナリパッケージを扱う場合，動作環境に依存するようになる．
    - ディストリビューションやバージョン，CPU アーキテクチャなどの動作環境が一致したパッケージを選択する必要がある．

2.4.2 dpkg コマンドを用いたパッケージ管理

Debian/GNU Linux や Ubuntu などの Debian 系ディストリビューションでは，パッケージ管理方法に Debian 形式 (deb 形式) が使われる．

Debian 形式のパッケージファイル名:

```
tree_1.6.0-1_i386.deb

tree: パッケージの名称
1.6.0: パッケージ番号
1: Debian リビジョン番号
i386: アーキテクチャ
deb: 拡張子
```

Debian 形式のパッケージを扱うには， `dpkg` コマンドを使う．

書式:

```
dpkg [option] action
```

option:

オプション	説明
`-E`	すでに同バージョンがインストールされていればインストールしない
`-G`	すでに新バージョンがインストールされていればインストールしない
`-R` `--recursive`	ディレクトリ内を再帰的に処理する

action:

アクション	説明
`-i` パッケージファイル名 `--install`	パッケージをインストールする
`-r` パッケージ名 `--remove`	設定ファイルを残してパッケージをアンインストールする
`-P` パッケージ名 `--purge`	設定ファイルも含め，完全にパッケージをアンインストールする

アクション	説明
`-l 検索パターン` `--list`	インストール済パッケージを検索して表示する
`-S ファイル名検索パターン` `--search`	指定したファイルがどのパッケージからインストールされたのか，表示する パターンにはワイルドカードが使える
`-L パッケージ名` `--listfiles`	指定パッケージからインストールされたファイルを， 一覧表示する
`-s パッケージ名` `--status`	パッケージの情報を表示する
`--configure パッケージ名`	展開されたパッケージを構成する
`--unpack パッケージ名`	パッケージを展開する (インストールはしない)

パッケージのインストール:

```
ai@ai-Standard-PC-Q35-ICH9-2009:~/Downloads$ ls
discord-0.0.16.deb
ai@ai-Standard-PC-Q35-ICH9-2009:~/Downloads$ sudo dpkg -i discord-0.0.16.deb
```

パッケージのアンインストール:

```
ai@ai-Standard-PC-Q35-ICH9-2009:~/Downloads$ sudo dpkg --purge discord
(Reading database ... 164852 files and directories currently installed.)
Removing discord (0.0.16) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
Processing triggers for mime-support (3.64ubuntu1) ...
```

インストール済パッケージの表示:

```
ai@ai-Standard-PC-Q35-ICH9-2009:~/Downloads$ sudo dpkg -l vim
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version      Architecture Description
+++-==============-============-============-=================================
un  vim            <none>       <none>       (no description available)
```

Debian パッケージのインストールでは，インストール時に対話的な設定が行われることがある．
`dpkg-reconfigure` コマンドを使うと，いつでも対話的な設定を実施することができる．

2.4.3 `apt-get` コマンド

- `apt-get`
  - APT (Advanced Packaging Tool) というパッケージ管理ツールに含まれるコマンド
  - 依存関係を調整しながらパッケージのインストール，アップグレード，アンインストールを行う．
  - インターネット経由で最新のパッケージの入手からインストールと依存関係の解決までできる．

書式:

```
apt-get [option] サブコマンド パッケージ名
```

option:

オプション	説明
`-d`	ファイルをダウンロードする (インストールはしない)
`-s`	システムを変更せず，動作をシミュレートする

サブコマンド:

サブコマンド	説明
`clean`	過去に取得し保持していたパッケージファイルを削除する
`dist-upgrade`	システムを最新にアップグレードする
`install`	パッケージをインストールまたはアップグレードする
`remove`	パッケージをアンインストールする
`update`	パッケージデータベースを更新する
`upgrade`	システムの全パッケージのうち， 他のパッケージを削除しないものをアップグレードする

`apt-get` コマンドでパッケージ管理を始めるには，まず `/etc/apt/sources.list` にパッケージを管理しているサイトの URL を記述する．

書式:

```
deb http://jp.archive.ubuntu.com/ubuntu/ focal main restricted

deb: deb (deb パッケージを取得) または deb-src (ソースを取得)
http://jp.archive.ubuntu.com/ubuntu/: 取得先のURI
focal: バージョン名
main restricted:
    main: 公式にサポートされるソフトウェア
    universe: コミュニティによってメンテナンスされるソフトウェア
    restricted: デバイス用のプロプライエタリなドライバ
    multiverse: 著作権もしくは法的な問題によって制限されたソフトウェア
    contrib: フリーではない依存関係のあるソフトウェア
    non-free: 利用と改変再配布に制限のあるソフトウェア
```

次に，設定したサイトに接続して最新のパッケージ情報を取得する．

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get update
```

パッケージをインストールするには， `apt-get install` コマンドを実行する．

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get install vim
```

これでパッケージがダウンロードされ，インストールされる．
取得したパッケージは， `/var/cache/apt/archives` 以下に格納される．
もしインストールしたいパッケージに必要なパッケージがインストールされていなければ，必要なパッケージが自動的にダウンロードされ，インストールされる．

パッケージをアンインストールするには， `apt-get remove` コマンドを実行する．

システムを一括して最新の状態にアップグレードするには， `apt-get dist-upgrade` コマンドを実行する．
このコマンドは，バージョンアップの際，重要度の高いパッケージをインストールするために既存のパッケージを削除することがある．
削除を伴わないパッケージのアップグレードを実行するには， `apt-get upgrade` コマンドを実行する．このコマンドは，システムの全パッケージのうち，他のパッケージを削除しないもののみをアップグレードする．

参考:
Ubuntu では， `do-release-upgrade` コマンドを使うと，メジャーバージョンの異なる大きなアップグレードをかけることができる．

2.4.4 `apt-cache` コマンド

`apt-cache` は，パッケージ情報を照会・検索するコマンド．照会・検索する対象はインストールされていなくても問題ない．

書式:

```
apt-cache サブコマンド
```

サブコマンド:

サブコマンド	説明
`search` キーワード	指定したキーワードを含むパッケージを検索する
`show` パッケージ名	パッケージについての一般的な情報を表示する
`showpkg` パッケージ名	パッケージについての詳細な情報を表示する
`depends` パッケージ名	指定したパッケージの依存関係情報を表示する

`apt-get` と `apt-cache` を合わせたようなコマンドが `apt` である． `apt` コマンドの利用が推奨されている．

書式:

```
apt [option] サブコマンド
```

option:

オプション	説明
`-c` 設定ファイル	設定ファイルを指定する デフォルトは `/etc/apt/sources.list`
`-d`	パッケージのダウンロードのみ行う `install` とともに
`-y`	問い合わせに対して自動的に yes と回答する
`--no-install-recommends`	必須ではない推奨パッケージはインストールしない
`--install-suggests`	推奨パッケージもインストールする

オプション	説明
`--reinstall`	インストール済のパッケージの再インストールも許可する

主なサブコマンド:

サブコマンド	説明
`update`	パッケージリストを更新する
`install パッケージ名`	パッケージをインストールする
`remove パッケージ名`	パッケージを削除する (設定ファイルは残す)
`purge パッケージ名`	パッケージを完全に削除する
`upgrade パッケージ名`	システムをアップグレードする (ファイル削除は伴わない)
`full-upgrade`	システムのメジャーバージョンを最新にアップグレードする
`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
`list`	パッケージのリストを表示する
`list --installed`	インストールされたパッケージを一覧表示する
`list --upgradable`	アップグレード可能なパッケージを表示する
`search キーワード`	指定したキーワードでパッケージ情報を全文検索する
`depends パッケージ名`	パッケージの依存関係を表示する
`autoremove`	必要とされていないパッケージを自動的に削除する

2.5 RPM パッケージの管理

RPM は Red Hat 社が開発したパッケージ管理システム．

2.5.1 RPM パッケージ

RPM パッケージのファイル名は以下のようになっている．

書式:

```
bash-4.2.46-30.el7.x86_64.rpm

bash: パッケージの名称
4.2.46: バージョン番号
30.el7: リリース番号
x86_64: アーキテクチャ
rpm: 拡張子
```

2.5.2 `rpm` コマンドの利用

`rpm` コマンドを使って RPM パッケージをインストールしたり，削除したり，アップデートしたりできる．
`rpm` コマンドにはいくつかのモードがあり，モードごとに多彩なオプションが用意されている．

- `rpm` コマンドの主なモード
  - インストール/アップグレードモード
  - アンインストールモード
  - 照会モード

インストール/アップグレードモード:

オプション	説明
`-i` パッケージ名 `--install`	パッケージをインストールする
`-U` パッケージファイル名 `--upgrade`	パッケージをアップグレードする (なければインストールする)
`-F` パッケージファイル名 `--freshen`	パッケージがインストールされていればアップグレードする

インストール/アップグレードモードで併用するオプション:

オプション	説明
`-v`	詳細な情報を表示する
`-h`, `--hash`	進行状況を `#` で表示する
`--nodeps`	依存関係を無視してインストールする
`--force`	既存のファイルを新しいものに置き換える
`--test`	実際にはインストールせずテストを実施する

アンインストールモード:

オプション	説明
`-e パッケージ名` `--erase`	パッケージをアンインストールする

アンインストールモードで併用するオプション:

オプション	説明
`--nodeps`	依存関係を無視してアンインストールする

照会モード:

オプション	説明
`-q パッケージ名`	指定したパッケージがインストールされているか照会する

照会モードで併用するオプション:

オプション	説明
`-a`, `--all`	インストール済のすべてのパッケージを表示する
`-f ファイル名`	指定したファイルを含むパッケージ名を表示する
`-p パッケージファイル名`	対象としてパッケージファイルを指定する
`-c`, `--configfiles`	設定ファイルのみを表示する

オプション	説明
`-d`, `--docfiles`	ドキュメントのみを表示する
`-i`, `--info`	指定したパッケージの情報を表示する
`-l`, `--list`	指定したパッケージに含まれるファイルを表示する
`-R`, `--requires`	指定したパッケージが依存しているファイルなどを表示する
`--changelog`	変更履歴を表示する

パッケージのインストール

- `-i` オプションを使う.
  - 経過を分かりやすくするために, `-v` オプションと `-h` オプションも併用するのが一般的.

パッケージのアップグレード

- `-U` オプションまたは `-F` オプションを使う.
  - 純粋にアップグレードのみを行うのが `-F` オプション.

例 (`~/rpms` ディレクトリ以下にある RPM ファイルをすべてアップグレード):

```
rpm -Fvh ~/rpms/*.rpm
```

パッケージのアンインストール

- `-e` オプションを指定する.
  - `--nodeps` オプションを使うと，依存関係を無視してアンインストールする.

パッケージ情報の照会

- `-q` オプションを指定する.

例:

```
[root@localhost ~]# rpm -qa | grep vim
vim-minimal-7.4.629-8.el7_9.x86_64
```

- `-qi`
  - 各パッケージの情報を表示する
- `-qip`
  - インストール前のパッケージ情報を表示する

- `-qf`
  - 指定したファイルがインストールされたパッケージを表示する

    ```
    [root@localhost ~]# rpm -qf /bin/bash
    bash-4.2.46-34.el7.x86_64
    ```

- `-qlp`
  - パッケージからどのようなファイルがインストールされるのか調べる
- `-qR`
  - パッケージの依存関係を調べる

パッケージの署名確認

- `--chacksig` または `-K` オプションを使う.

パッケージの展開

RPM パッケージをインストールせず，その内容を展開するには， `rpm2cpio` コマンドを使う.
使用時は，アーカイブを展開する `cpio` コマンドと組み合わせて使う.

例:

```
rpm2cpio tree-1.6.0-10.el7.x86_64.rpm | cpio -id
```

2.5.3 YUM

CentOS や Fedora では，APT ツールに相当するものとして **YUM** (Yellow dog Updater, Modified) がある．
YUM の設定は， `/etc/yum.conf` と `/etc/yum.repos.d` ディレクトリ以下のファイルで行う．

`/etc/yum.conf`：

```
[root@localhost etc]# cat yum.conf
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php?project_id=23&ref=http://bugs.centos.org/bug_repo
distroverpkg=centos-release


#  This is the default, if you make this bigger yum won't see if the metadata
# is newer on the remote and so you'll "gain" the bandwidth of not having to
# download the new metadata and "pay" for it by yum not having correct
# information.
#  It is esp. important, to have correct metadata, for distributions like
# Fedora which don't keep old packages around. If you don't like this checking
# interupting your command line usage, it's much better to have something
# manually check the metadata once an hour (yum-updatesd will do this).
# metadata_expire=90m

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

`/etc/yum.repos.d` ディレクトリ以下には，リポジトリ情報の設定ファイルが配置される．パッケージの入手先を増やしたい場合は，リポジトリ情報の設定ファイルを追加する．

```
[root@localhost etc]# ls /etc/yum.repos.d/
CentOS-Base.repo       CentOS-fasttrack.repo  CentOS-Vault.repo
CentOS-CR.repo         CentOS-Media.repo      CentOS-x86_64-kernel.repo
CentOS-Debuginfo.repo  CentOS-Sources.repo
```

YUM を使った管理は `yum` コマンドで行う．

書式:

yum サブコマンド

yum コマンドの主なサブコマンド:

サブコマンド	説明
check-update	アップデート対象のパッケージリストを表示する
update [パッケージ名]	指定したパッケージをアップデートする
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
remove パッケージ名	指定したパッケージをアンインストールする
info パッケージ名	指定したパッケージの情報を表示する
list	全パッケージ情報をリスト表示する
repolist	リポジトリー覧を表示する
search キーワード	パッケージ情報をキーワードで検索する
search all キーワード	パッケージをキーワードで検索する パッケージ名および説明文等すべて
groups list	パッケージグループをリスト表示する
groups install グループ	指定したグループのパッケージをインストールする

アップデート

- yum check-update
  - インストールされているパッケージの中で，アップデートパッケージが存在するパッケージのリストが表示される．
- yum update
  - インストールされている全パッケージが最新版にアップデート

インストールとアンインストール

- yum install
  - 指定したパッケージをネットワーク経由で取得し，インストールできる．
- yum remove
  - パッケージをアンインストールする．

パッケージ情報の確認

- `yum info`
  - パッケージ情報を表示する.
- `yum list`
  - リポジトリにあるすべてのパッケージ情報と，インストールされているかどうかを確認する.
- `yum search`
  - パッケージ情報をキーワードで検索する.

パッケージグループ単位のインストール

- `yum groups list`
  - どのようなグループがあるのかを表示する.
- `yum groups install`
  - グループのパッケージをまとめてインストールする.

2.5.4 `dnf` コマンド

これまで `yum` を採用していたディストーションでは， `yum` コマンドに代わって `dnf` コマンドが使われてきている (Fedora 等)．基本的には `yum` コマンドの使い方とほぼ同じ．

書式:

```
dnf サブコマンド
```

`dnf` コマンドの主なサブコマンド:

サブコマンド	説明
`check-update`	アップデート対象のパッケージリストを表示する
`clean`	キャッシュデータを削除する
`upgrade (update)`	システムの全パッケージをアップグレードする
`upgrade パッケージ名`	指定したパッケージをアップグレードする
`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
`remove パッケージ名`	指定したペッケージをアンインストールする
`info パッケージ名`	指定したパッケージの情報を表示する
`list`	全パッケージ情報をリスト表示する
`search キーワード`	パッケージ情報をキーワードで検索する
`history`	処理の履歴を表示する
`updateinfo`	パッケージのアップデート情報を表示する

2.5.5 `zypper` を使ったパッケージ管理

openSUSE でも RPM パッケージが使われるが，パッケージ管理には `zypper` コマンドを使う．

書式:

```
zypper [サブコマンド]
```

`zypper` コマンドの主なサブコマンド:

サブコマンド	説明
`install パッケージ名` `in パッケージ名`	指定したパッケージをインストールする
`remove パッケージ名` `rm パッケージ名`	指定したパッケージをアンインストールする
`info パッケージ名`	指定したパッケージの情報を表示する
`update` `up`	システムの全パッケージをアップデートする
`update パッケージ名` `up パッケージ名`	指定したパッケージをアップデートする
`list-updates` `lu`	アップデート対象のパッケージリストを表示する
`dist-upgrade` `du`	ディストリビューションをアップグレードする
`search キーワード` `se キーワード`	パッケージ情報をキーワードで検索する

2.6 仮想化のゲスト OS としての Linux

2.6.1 クラウドサービスとインスタンス

最近は，クラウドサービス上の仮想的な Linux マシン **インスタンス** を利用することが一般的になっている．
物理サーバを扱うよりも素早くインフラを整えたり，簡単にリソース (CPU，メモリ，ストレージ容量など) を拡張できる．

- **IaaS** (Infrastructure as a Service)
  - クラウドサービス提供者が，ネットワークから OS までをサービスとして利用者に提供する形態．
  - クラウドサービス提供者が管理する物理サーバ上で仮想マシンを起動し，利用者は仮想マシンを自由に扱うことができる．
  - ミドルウェアやアプリケーションは必要に応じて利用者自らが仮想マシンにインストールする．
  - 提供される個々の仮想マシンは **インスタンス** とも呼ばれる．
    - ソフトウェア的に構成されたコンピュータ．
- **PaaS** (Platform as a Service)
  - IaaS での提供部分に加えてアプリケーション実行環境まで提供される．
    - 開発環境やデータベース，プログラミング言語の実行環境などが提供され，すぐに開発に利用することが可能．
- **SaaS** (Software as a Service)
  - アプリケーションそのものが提供される．

クラウドコンピューティングの主な用語:

用語	説明
ブロックストレージ	仮想的なディスクストレージ．物理的なサーバとは異なり， 簡単に容量を追加できる．
ゲストOS	仮想マシンにインストールされたOS
OSイメージ	インスタンスのテンプレートとなるディスクイメージ
インスタンス	クラウド上で動作する個々の仮想マシン
コンテナ	独立したOSのように扱えるアプリケーション実行環境． 仮想マシンよりも消費リソースが小さく軽量

用語	説明
アパライアンスコンテナ	特定用途向けに あらかじめWebサーバやデータベースなどが組み込まれた状態で 用意されているコンテナイメージ
ゲストドライバ	仮想マシンがホストマシン上のデバイスにアクセスする際などに 使われるソフトウェア．仮想マシンにインストールして利用する．

2.6.2 インスタンスの初期化

クラウドでは，テンプレートから簡単にインスタンスを作成できる．
しかし，テンプレートをコピーしただけでは，本来はサーバごとに異なるはずのホスト名やSSH鍵が重複してしまう．

Cloud-init: インスタンスを初期化する仕組み．
テンプレートとなるOSイメージには Cloud-init が組み込まれており，インスタンスの初回起動時に，ユーザデータに基づき適切な設定が行われる．

Cloud-init で設定できる情報:

- ホスト名
- SSH 公開鍵
- 一般ユーザ
- インストールするパッケージ

個々のインスタンスの識別に **D-Bus マシン ID** が利用できる．
`/etc/machine-id` ファイルに格納されており，インスタンスごとに異なる値が割り当てられている．

```
[ai@localhost ~]$ cat /etc/machine-id
f85ca97c70e745e69225e2055709b116
```