

# Tokyo OpenSolaris Study Group



## 「IPS」のパッケージ作成入門

～だれでもcontributer計画～

**OpenSolaris Users Group Leader**

**ジャストプレイヤー株式会社**

**瀧 康史 / TAKI, Yasushi**

# Agenda

## JDS CBE (Common Build Environment)の作成

IPSについて

pkgコマンドの利用方法(逆引き的な利用方法)

各種レポジトリについて(主観)

## システム管理者、開発者のためのIPS

pkg/serverの立ち上げ方





# opensolaris™

## JDS Common Build Environment の作成

# configure && make よりも . . .

なぜパッケージ化するのか?

- 自分のため
  - インストールしたパッケージを再びインストールするときに楽。
  - 同じ環境を作り込むのが楽。
  - アンインストールが簡単
    - まあbeadm使えばconfigure && makeでもいいけどね
- 人のため
  - 誰か次にインストールする人が楽

IPSには従来のSVRのように.pkg形式というローカルに保存出来る形式がありません。

SVR4のPKGは非常にシンプルな構成であったので、configure && make && make install するよりも、SVR4のパッケージを作り、インストールした方が、後でアンインストールが容易です。

# 環境の準備

ビルド用、環境テスト用、レポジトリ用と3つの役割をもつサーバが必要になる。

- ビルド用
  - ビルドに必要なパッケージを入れる。
- 環境テスト用
  - 依存関係がちゃんと取れるかを確認用。
  - ほとんど何も無い状態でsnapshotをとっておき、何度でも構成できるようにする。
- ローカル用IPSサーバ
  - pkg.depodを起動するもの
  - ビルドサーバに1つたててしまっても十分です。

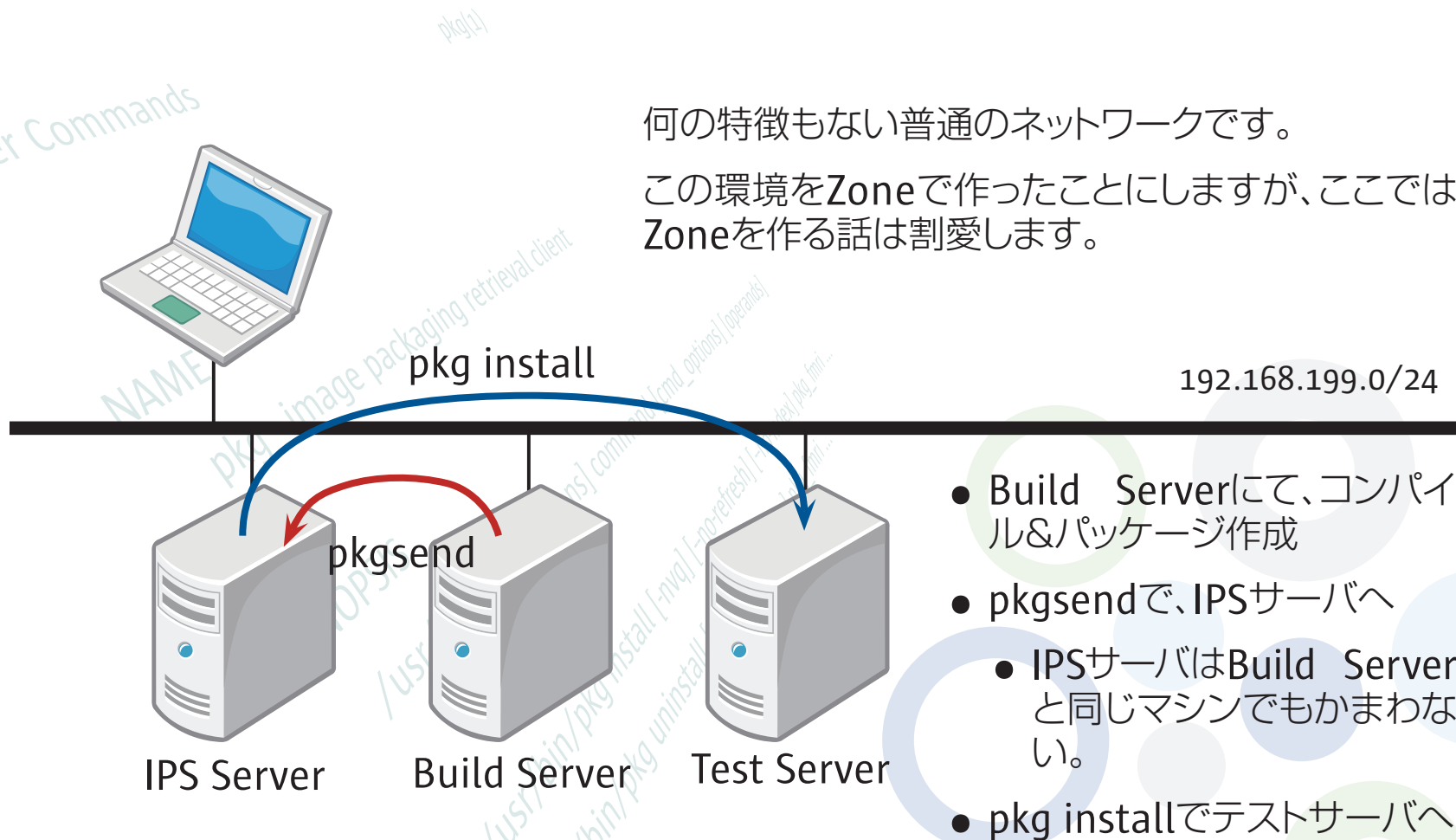
VirtualBoxで何台も作っても良いが、Kernelに關係するパッケージをIPSに入れない限り、Zoneでもよい。

Zoneはリソースもあるし、snapshotもとれるので便利でしょう。

# パッケージビルドファーム (造語)

何の特徴もない普通のネットワークです。

この環境をZoneで作ったことにしますが、ここではZoneを作る話は割愛します。



# IPS サーバの設定

pkg/serverは、SUNWipkgに含まれているので、pkgコマンドとともにインストールされています。

右は、起動の様子とログの状態です。

```
pfexec svcadm enable pkg/server
```

デフォルトでは、

- ポート80でListen
- 読み書き両用(pkgsend可能)

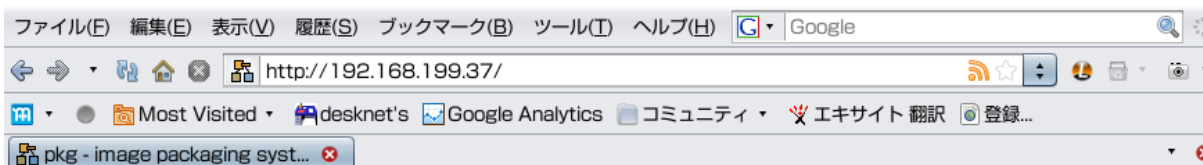
で、レポジトリが立ち上がっています。  
この状態は外からなんでもできるのでご注意。

```
root@test-ips:~# svcs -xv pkg/server
svc:/application/pkg/server:default (image packaging repository)
  State: disabled since Thu Apr 02 16:04:43 2009
  Reason: Disabled by an administrator.
  See: http://sun.com/msg/SMF-8000-05
  Impact: This service is not running.
root@test-ips:~# svcadm enable pkg/server
root@test-ips:~# svcs -xv pkg/server
svc:/application/pkg/server:default (image packaging repository)
  State: online since Thu Apr 02 16:05:32 2009
  See: /var/svc/log/application-pkg-server:default.log
  Impact: None.
root@test-ips:~# cat /var/svc/log/application-pkg-server:default.log
[ Apr 2 16:05:32 Enabled. ]
[ Apr 2 16:05:32 Executing start method ("/lib/svc/method/svc-pkg-depot start"). ]
ppriv -s A=basic,-file_link_any,-proc_info,-proc_session,net_privaddr -e /usr/lib/pkg.depotd -d /var/pkg/repo -p 80 -s 10 -t 60 --content-root=/usr/share/lib/pkg --log-access=none --log-errors=stderr
[02/Apr/2009:16:05:32] INDEX Search Available
[02/Apr/2009:16:05:32] ENGINE Listening for SIGHUP.
[02/Apr/2009:16:05:32] ENGINE Listening for SIGTERM.
[02/Apr/2009:16:05:32] ENGINE Listening for SIGUSR1.
[02/Apr/2009:16:05:32] ENGINE Bus STARTING
[02/Apr/2009:16:05:32] ENGINE Started monitor thread '_TimeoutMonitor'.
[02/Apr/2009:16:05:33] ENGINE Serving on 0.0.0.0:80
[02/Apr/2009:16:05:33] ENGINE Bus STARTED
```



# 立ち上がった IPS の確認

ブラウザでアクセスをすれば、立ち上がった、状態の確認ができます。



現時点では、なにも立ち上がっていないので、なにもありません。

## Statistics

Number of packages: 0  
Number of in-flight transactions: 0  
  
Number of catalogs served: 0  
Number of manifests served: 0  
Number of files served: 0  
Number of flists requested: 0  
Number of files served by flist: 0  
Number of packages renamed: 0

## 確認の様子

```
# netstat -an | grep LISTEN | grep *.80  
*.80 *.*
```

```
49152 0 LISTEN
```

※Listenプロセスを知りたい場合はPIDをしらべてpfilesする



Last Updated: None

## Catalog

FMRI	Info	Manifest
------	------	----------

完了 kohju YSlow 0.377s Proxy: なし



# JDS Common Build Environment

## ビルド環境の統一化(Common Build Environment)

- xxは簡単にビルドできるよ。ただし、僕の環境ではね…では、困る。
  - コンパイラ、環境変数、PATHの指定、パッチなどをみんな同じ環境で!
- すべてのポーター(Softwareをポーティングする人)が統一した環境下でビルドができる必要がある。
  - 誰かが後を次いでパッケージメンテナーになってくれるかもしれない。

## ポーター向けビルド方法のノウハウをレポジトリに(spec file)

- ダウンロード先のtar ballのありか、パッチ、様々なメタ情報等々を、統一した形式で管理したい。それをファイルに一元管理化させる。
- 元々は、Redhatの文化だったものを、SFE(Spec File Extra)という名前で、Solaris用に拝借してきた。コンパチブルではない。

## 共有のビルドマシンが欲しい(それがSource Juicer)

- 自分の環境は、たとえCBEであわせたとしても、何かが入ってしまうもの。
- ある担当者がビルド環境に知らずに手を入れていると困る。

# つまり・・・こんな流れです

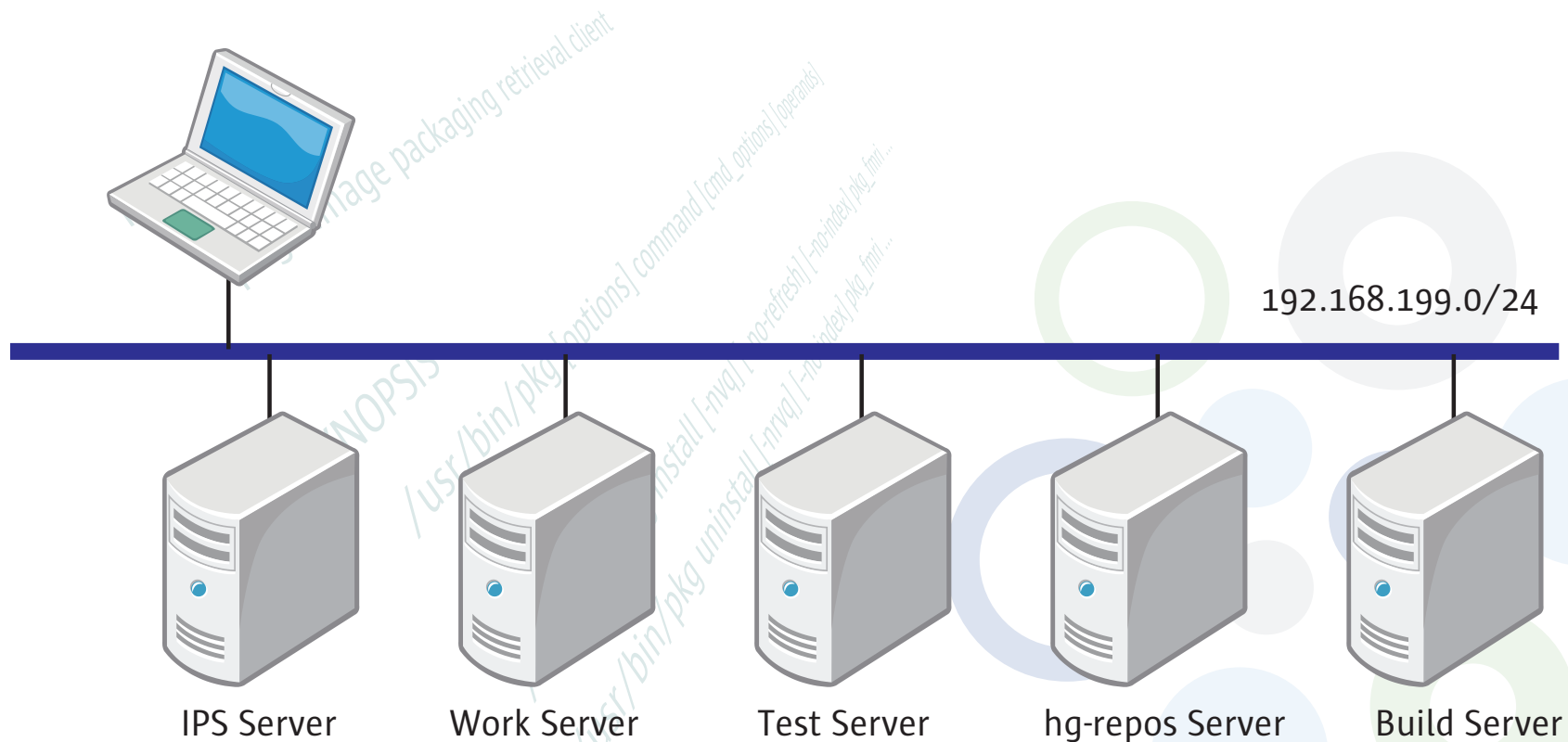
- ローカル環境
  - CBEを作る
  - specファイルを書く
  - ビルドに成功して、自分のマシンでインストールテストする。
    - インストールには自分用IPSサーバが、test serverから見えるようにし、publisher(authority)登録しておく。
- SourceJuicer
  - specファイル、Patch、コピーライトなどをsource juicerにアップロードする
  - aproverにコピーライト承認が降りるのを待つ。
  - ビルドグリッドが動いて自分のプログラムがコンパイルされるのを待つ!
  - PASSしたら、pending入りu
- テスト
  - <http://juicr.opensolaris.org/pending> をテスト環境のpkg publisher(authority)に登録。
  - インストールしてみる

# ビルドファームの例 (大)

たとえばこのような形のマシンが必要になります。

Work Server、Test Serverはコミッタの数だけあると便利です。

zone+zfsってすてきですね。



# spec ファイルのビルド環境作成 #1

## オペレータをPrimary Administratorに

まず、オペレーションユーザにPrimary Administratorの権限をつけておきます。この作業をした後は、シェルの再起動をする必要があります。

```
pfexec usermod -P 'Primary Administrator'
```

ユーザ名反映させるため、利用しているシェルの起動しなおしをします。

## 事前に入れておくべきもの

```
pfexec pkg install -v \  
  SUNWgtar \  
  SUNWxcu4 \  
  SUNWcar \  
  SUNWkvm
```

とりあえず、いれておきます。

# spec ファイルのビルド環境作成 #2

## 開発環境のインストール

```
pfexec pkg install -v \  
ss-dev \  
SUNWgnome-common-devel \  
SUNWperl-xml-parser \  
SUNWgnome-xml-root SUNWgnome-xml SUNWgnome-xml-share
```

## Studio12のインストール作業

コンパイラ(開発環境)は、gcc-dev、ss-dev(StudioExpress)、Studio12が選べます。しかし、現時点でON(OS+Network)がStudio12でビルドしてあるため、「私としては」Studio12をおすすめしたいです。ここではss-devの依存物も一緒に入れたいので、ss-devは念のためいれてあります。

Studio12は、こちらからダウンロードします。

```
http://www.opensolaris.org/os/community/tools/sun_studio_tools/sun_studio_12_tools/  
pfexec mkdir /opt/SUNWspro/  
cd /opt/SUNWspro/  
pfexec gtar zxvf sunstudio12-ii-20081010-sol-x86.tar.gz
```

※出たばかりのSPは未調査 :)



# spec ファイルのビルド環境作成 #3

## JDS CBE (Common Build Environment) 1.7+をインストール

このプログラムはSVR4のパッケージでインストールされます。

```
wget \  
http://dlc.sun.com/osol/jds/downloads/cbe/test/desktop-cbe-1.7.0-rc1-x86.tar.bz2  
gtar jxvf desktop-cbe-1.7.0-rc1-x86.tar.bz2  
cd desktop-cbe-1.7.0-rc1  
./cbe-install
```

質問がいっぱい出てきますが、あわてずにかげば、そう極端な事はかいていません。どうしても例が欲しい場合は、私のブログを参照(<http://kohju.justplayer.com>)。



# spec ファイルのビルド環境作成 #4

## pkgbuildをダウンロード

cbeに含まれているpkgbuildが今は古いので、最新版を<http://sourceforge.net/projects/pkgbuild/>からダウンロードするか、あるいはIPSサーバ「<http://jucr.opensolaris.org/pending>」から、インストールします。ここではIPSサーバから入れる方法を記載します。

1.3.1が入っているのでアンインストール

```
pfexec pkgrm SFpkgbuild
```

## SourceJuicerのpendingレポジトリを追加

※SourceJuicerのpendingレポジトリは、Test Serverにも追加します。

```
pfexec pkg set-publisher -0 http://jucr.opensolaris.org/pending/ sourcejuicer
```

juicerのレポジトリは開発用なので、「毎日」何かがカレントで書き換わる。次のようにして、レポジトリのメタ情報のリフレッシュは頻繁にすると良いでしょう。

```
pfexec pkg refresh juicer
```

pkgbuildのインストールをします。

```
pfexec pkg install -v pkgbuild
```

# spec ファイルのビルド環境作成 #5

## specファイルのincludeを用意する

spec-filesのincludeファイルと、いろいろなincludeファイルを取得します。下記の例ではtrunkを引っ張ってきます。

```
svn co svn+ssh://anon@svn.opensolaris.org/svn/jds/spec-files/trunk spec-files
```

基本のspec用includeファイルのコピー

```
cp spec-files/include/*.inc ~/packages/SPECS/
```

## ACLOCALの修正

/opt/dtbld/share/aclocal以下に、CBE m4のファイルが見つからないと、コンパイル時にACLOCALでエラーが出ます。この場合、/usr/share/aclocal/dirlistに、下記のものがあるか確認する必要があります。

```
/usr/sfw/share/aclocal  
/opt/dtbld/share/aclocal
```

aclocalという名で実行できないのでシンボリックリンクで回避します。

```
ln -s /usr/bin/aclocal-1.10 /usr/bin/aclocal
```



# spec ファイルのビルド環境作成 #6

## サンプルのSPECファイルのインストール

workディレクトリを決めて、svnからチェックアウトしてきます。

```
svn co svn+ssh://anon@svn.opensolaris.org/svn/jds/spec-files/trunk spec-files-trunk
svn co svn+ssh://anon@svn.opensolaris.org/svn/jds/spec-files/branches/gnome-2-24 spec-files-2-24
```

ディレクトリはこんな感じ。

```
~/work/spec-files-trunk
  /gnome-2-24 spec-files-2-24
```

以後、これをサンプルとして利用する。

## IPSサーバ

運用のためのものではないので、pkg/serverの起動するだけ

```
pfexec svcadm enable pkg/server
```

インストールのためにauthorityの登録をしておく。

```
pfexec pkg set-authority -0 http://localhost:80 mypkgs
```

# spec ファイルのビルド環境作成 #7

## 環境変数の設定

```
. /opt/dtbld/bin/env.sh
```

ビルドユーザの`.login`とか`.zshrc`とかに下記に追加もあり。

```
. /opt/dtbld/bin/env_include.sh
```

※`LC_ALL`とかはCのほうがよいでしょう。

## コンパイルの例

```
cd spec-files-2-24  
/opt/dtbld/bin/pkgtool --download --ips build-only SUNWTiff.spec
```

このようにすると、コンパイル後、IPSに自動的に登録します。

コンパイルの中間ファイルや結果は、`~/packages/`以下に保存されます。

`--ips`ではなく、`--svr4`にした場合、従来のSVR4のパッケージを作成します。

このようにすることで、**同じメタ情報とソースコードから、IPSとSVR4の両方のレポジトリの作成が可能**です。

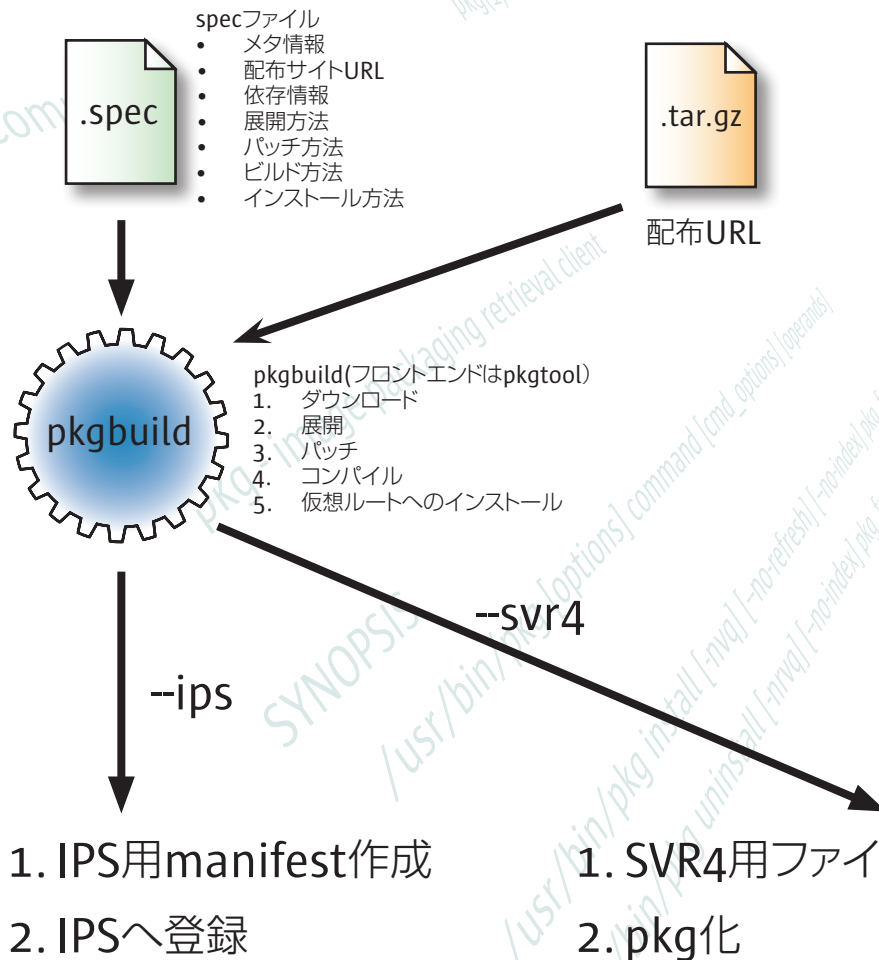
以上で、パッケージを制作するための、JDS CBE環境の構築は終わりです。



# opensolaris™

## spec ファイルを作成する

# spec から作られるもの



`pkgbuild` (フロントエンドは `pkgtool`) を利用することで、`spec` ファイルに記述したソース配布URLから自動的にファイルをダウンロードし、ビルド、IPSに登録、あるいはSVR4の `pkg` ファイルを作成することができる。

記述するファイルは、

- `spec`
- `copyright`
- パッチ (あれば)

程度である。

※ `rpm` のそれとは似て非なるもの!

# spec ファイルと manifest の関係

- specファイルはRedHat Linux系からきている考え方。
  - 移植品ではなく、perlでかかれた異なる実装 (一般的にexperimental)
- SolarisではSpec File Extra (SFE)という名前のパッケージ体系のものだった。
  - もともとのSFEはspecを配り、「svr4のパッケージを作ってインストールする」ためのものであった。
- IPSのManifestよりも、さらにバックヤードの考え方を持つ。
  - 配布サイトの場所
  - パッチ
  - ソースのビルド方法
  - パッケージの構成情報

manifest

.spec

ダウンロード

展開

パッチ

ビルド

テスト環境へのインストール

パッケージメタ情報作成

パッケージ化

どのレポジトリに登録するか?

# 開発環境の work ディレクトリ構造

ルールではないけれども、私が使ってるサンプルのディレクトリ構造です。

```
work/pkglabo
```

```
|-- bin          <- スクリプト群
|  |-- mk-Requires.pl
|  |-- pkgsend_manifest.sh
|  `-- specbuild.sh
|-- manifests    <- specファイルでは作れないmanifest群
|  |-- GNUmakefile
|  |-- JPCenvcmds.lst
|  |-- JPCenvcmds.manifest
|  ~割愛~
|-- specs        <- specファイル群
|  |-- eb.copyright
|  |-- eb.spec
|  |-- ebview.copyright
|  |-- ebview.spec
|  |-- lv.copyright
|  |-- lv.spec
|  |-- only-depend.spec.sample
|  |-- patches
|  `-- lv-01-kohju.diff
|-- xz.copyright
|-- xz.spec
```

```
specbuild.sh
```

```
#!/bin/sh
PKGTOOL=/opt/dtbuild/bin/pkgtool
SOURCES=~/packages/SOURCES/
SPEC=$1
if [ \! -f ${SPEC} ]; then
    echo specbuild.sh file.spec
fi
NODE=${1%.spec}
${PKGTOOL} build-only --patchdirs=`pwd`/patches
--sourcedirs=`pwd` --ips --download ${SPEC}
```



# spec のサンプル

specファイルのセクションについて

参考) [http://jucr.opensolaris.org/help/spec\\_file](http://jucr.opensolaris.org/help/spec_file)

基本構造は大体次の通りです。

- |                      |                         |
|----------------------|-------------------------|
| 1. メタ情報セクション         | メタ情報を記述する               |
| 2. %prep/%setupセクション | アーカイブを展開し、ディレクトリを作成する。  |
| 3. %buildセクション       | パッチや、ビルドのためのスクリプトを書きます。 |
| 4. %installセクション     | 仮想のroot '/'にインストールします。  |
| 5. %files セクション      | インストール構造を示します。          |
| 6. %changelogセクション   | 更新履歴を記載する               |

# メタ情報セクション

```
%include Solaris.inc
```

```
Name: nano
Summary: GNU nano text editor
Version: 2.0.9
License: GPLv2
Url: http://www.nano-editor.org
Source: http://www.nano-editor.org/dist/v2.0/{name}-{version}.tar.gz
Group: Editor
Distribution: OpenSolaris
Vendor: OpenSolaris Community
BuildRoot: ${_tmppath}/${name}-${version}-build
SUNW_Basedir: ${_basedir}
SUNW_Copyright: ${name}.copyright
%include default-depend.inc
```

パッケージの名前  
サマリー  
バージョン名  
ライセンス名 (意味コードではない)  
WEBサイト  
ファイル配布URL  
グループ  
参照)  
<http://opensolaris.org/os/community/sw-porters/contributing/ipsclass/>

```
# OpenSolaris IPS Manifest Fields
Meta(info.upstream): Chris Allegretta
Meta(info.maintainer): Peter Jones
Meta(info.repository_url): svn://svn.sv.gnu.org/nano/trunk/nano/
Meta(info.classification): Editor
```

sourcejuicerでは、まずライセンスチェックされます。英語サイトがない場合は、日本語のURLを書き、翻訳してあげると良いです。

ソースコード・メンテナー  
パッケージ・メンテナー  
レポジトリのURL  
IPS Class (Groupと一緒に?)

```
%description
```

```
GNU nano is an effort to provide a Pico-like editor, but also includes some features that were missing in the original, such as 'search and replace', 'goto line' or internationalization support.
```

説明文





# %prep/%setup セクション

```
%prep  
rm -rf %name-%version  
%setup -q -n nano-%version
```

%setupは、ワークディレクトリ作成、tarで展開、cdまでをしてくれます。

64bitも同時に作る場合は、下記のようなものを入れ、一度展開したものをリネームし、再び展開してあげると、-64付きのディレクトリと、そうでないものができます…。泥縄っぽいですが、こういう解決方法ばかりです。

```
%ifarch amd64 sparcv9  
mv %{name}-%{tarball_version} %{name}-%{tarball_version}-64  
gtar zxvf %SOURCE0  
%endif
```

# %build セクション

```
%build
export CFLAGS="%optflags"
export LDFLAGS="%{_ldflags}"
./configure --prefix=%{_prefix} \
            --bindir=%{_bindir} \
            --mandir=%{_mandir} \
            --infodir=%{_infodir} \
            --sysconfdir=%{_sysconfdir} \
            --enable-all

make
```

configure && makeを行うセクションです。ビルドのための変数を設定したり、gmakeの-j オプションの数を設定したりします。

実は、この直前にはpatchが入ったりすることもあり、一筋縄でいかないものでは、一番アツイことをしているセクションでもあります。

# %install セクション

```
%install  
rm -rf $RPM_BUILD_ROOT  
make install DESTDIR=$RPM_BUILD_ROOT  
rm -f $RPM_BUILD_ROOT%{_infodir}/dir
```

インストールコマンドを記載します。

通常の最近のconfigure && makeの仕組みでは「DESTDIR=仮のディレクトリ」を設定して置くことで、指定したWORKディレクトリにインストールを行います。

パッケージはそのディレクトリを起点に、パッケージを作成します。

ただし、ソース提供者がかならずこういうテストをしているわけではないので「しばしば」このDESTDIRがきちんと機能していないソースもあります。

そのような場合は、DESTDIRが動くように手でpatchを当ててるのです。

# %files セクション

```
%files
%defattr (-, root, bin)
%dir %attr (0755, root, bin) %[_bindir]
%[_bindir]/*
%[_infodir]/*
%dir %attr(0755, root, sys) %[_datadir]
%dir %attr(0755, root, bin) %[_mandir]
%dir %attr(0755, root, bin) %[_mandir]/*
%[_mandir]/*/*
```

ここも重要なセクションで、実際のディレクトリのインストール先の配置を決めます。

実は、`basedir`よりも上にはインストールはできません。通常のアプリは`basedir`が`/usr`になっているので、`/usr`以外にいれるものは、`-root`付のパッケージも同時に作ることによって実現するのです。

# %changelog

%changelog

- \* Thu Oct 23 2008 - andras\_dot\_barna\_at\_gmail\_dot\_com
  - new version, add --enable-all, add SFEncursesw for utf-8
- \* Wed Jul 5 2006 - laca\_at\_sun\_dot\_com
  - delete -share subpkg
  - update file attributes
- \* Fri Feb 3 2006 - mike\_kiedrowski (lakeside-AT-cybrzn-DOT-com)
  - Initial version

ここは、ログですね。



# ビルド作業

ビルドは一般ユーザで行います。

```
kohju@work-spec(3333)% ../bin/specbuild.sh xz.spec
INFO: IPS packages will be installed by default from http://localhost:80
INFO: Copying %use'd or %include'd spec files to SPECS directory
INFO: Processing spec files
INFO: Finding sources
INFO: Running pkgbuild -ba [...] xz.spec (xz)
INFO: xz PASSED
```

Summary:

NAME	package	status	details
	xz	PASSED	

失敗の時、ログは最後にまとめて出力されるので、**ERROR**が出た場合は参考にします。

```
INFO: IPS packages will be installed by default from http://localhost:80
INFO: Copying %use'd or %include'd spec files to SPECS directory
INFO: Processing spec files
INFO: Finding sources
INFO: Running pkgbuild -ba [...] xz.spec (xz)
ERROR: xz FAILED
INFO: Check the build log in /tmp/xz.log for details
```

Summary:

NAME	package	status	details
	xz	FAILED	pkgbuild build failed

(注意)

環境変数に依存するので、**env.sh**を再び実行するのもポイント

# ビルド後のディレクトリ

```
packages/
|-- BUILD
|  |-- CBEenv-1.7.0-rc1
|  |-- lv-4.51
|  |  `-- lv451
|  |     |-- GPL.txt
|  |     |-- README
|  |     |-- build
|  |     |  |-- Makefile
|  |     |  |-- README
|  |     |  |-- big5.o
|  |     ~割愛~
|  `-- xz-4.999.8
|     |-- xz-4.999.8beta
|     |  |-- ABOUT-NLS
|     ~割愛~
|-- PKGMAPS
|  |-- copyright
|  |  |-- lv-src.copyright
|  |  |-- lv.copyright
|  |  |-- xz-src.copyright
|  |  `-- xz.copyright
|  |-- depend
```

<- ビルドしているディレクトリ

<- copyright ファイルが保存される

<- depend ファイルが保存される (SVR4PKGの依存ファイル)

```

|-- CBEenv.depend
|-- manifests <- manifests ファイルが保存される (IPS用のmanifest)
|-- lv.manifest
|-- xz.manifest
|-- pkginfo <- pkginfo ファイルが保存される (SVR4PKGのメタ情報記述)
|-- CBEenv-src.pkginfo
|-- CBEenv.pkginfo
|-- lv-src.pkginfo
|-- lv.pkginfo
|-- xz-src.pkginfo
|-- xz.pkginfo
|-- proto <- proto ファイルが保存される (SVR4PKGのファイル一覧)
|-- lv-src.proto
|-- lv.proto
|-- xz-src.proto
|-- xz.proto
|-- scripts <- IPSなどに登録するために実際に動作するスクリプト
|-- CBEenv.preremove
|-- bchunk_ips.sh
|-- lv_ips.sh
|-- xz_ips.sh
|-- PKGS <- インストール後のDESTDIR root
|-- CBEenv

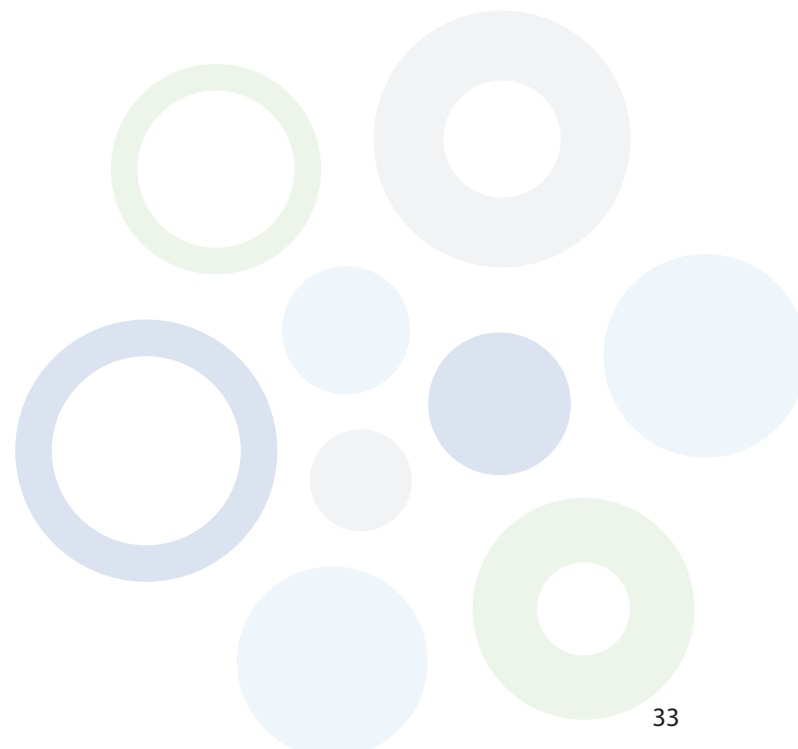
```



NAME

```
|-- install
|   |-- depend
|   |-- preremove
|-- pkginfo
|-- pkgmap
|-- reloc
|   |-- bin
|       |-- env.csh
|       |-- env.sh
|       |-- env_include.sh
|       |-- gendiff
|       |-- ld-wrapper
|-- lv
|   |-- install
|   |-- copyright
|-- pkginfo
|-- pkgmap
|-- reloc
|   |-- bin
|       |-- lgrep
|-- xz
|   |-- install
```

pkg(4)  
pkg - image packaging retrieval client  
/usr/bin/pkg [options] command [cmd\_options] [operands]  
/usr/bin/pkg install [-rvq] [-no-refresh] [-no-index] pkg\_fmri...  
/usr/bin/pkg uninstall [-rvq] [-no-index] pkg\_fmri...



```
|         | `-- copyright
|         |-- pkginfo
|         |-- pkgmap
|         `-- reloc
|         |-- bin
|         |   |-- amd64
|         |   |-- lzdiff
|         |   |-- lzgrep
|         ~割愛~
```

-- SOURCES

```
| -- env.csh
| -- env.sh
| -- env_include.sh
| -- gendiff
| -- ld-wrapper
| -- lv-01-kohju.diff
| -- lv.copyright
| -- lv451.tar.gz
| -- xz-4.999.8beta.tar.gz
| `-- xz.copyright
```

-- SPECS

```
| -- CBE.inc
| -- Solaris.inc
```

<- sourceアーカイブ、メンテナーが作ったパッチ等。

<- includeするspecファイル

```

|-- arch64.inc
|-- base.inc
|-- default-depend.inc
|-- options.inc
|-- prod.inc
|-- x86_sse2.inc
`-- SPKGS
    |-- CBEenv-src
    |   |-- pkginfo
    |   |-- pkgmap
    |   |-- reloc
    |   |-- share
    |   |-- src
    |   |-- CBEenv-1.7.0-rc1
    |   |-- SOURCES
    |   |   |-- env.csh
    |   |   |-- env.sh
    |   |   |-- env_include.sh
    |   |   |-- gendiff
    |   |   |-- ld-wrapper
    |   |-- SPECS
    |   |   |-- CBE.inc
    |   |   |-- CBEenv.spec

```

<- ソースパッケージ（ほぼ未使用？）



```
|
|-- lv-src
|   |-- install
|   |   |-- copyright
|   |   |-- pkginfo
|   |   |-- pkgmap
|   |   |-- reloc
|   |   |-- share
|   |       |-- src
|   |           |-- lv-4.51
|   |               |-- SOURCES
|   |                   |-- lv-01-kohju.diff
|   |                   |-- lv.copyright
|   |                   |-- lv451.tar.gz
|   |               |-- SPECS
|   |                   |-- Solaris.inc
|   |                   |-- arch64.inc
|   |                   |-- base.inc
|   |                   |-- default-depend.inc
|   |                   |-- lv.spec
|   |                   |-- options.inc
|   |                   |-- prod.inc
|-- xz-src
```

User Commands

NAME

pkg - image packaging retrieval client

SYNOPSIS

/usr/bin/pkg [-option] command [cmd\_options] [operands]



```
|-- install
|  |-- copyright
|-- pkginfo
|-- pkgmap
|-- reloc
  |-- share
    |-- src
      |-- xz-4.999.8
        |-- SOURCES
        | |-- xz-4.999.8beta.tar.gz
        | |-- xz.copyright
        |-- SPECS
        | |-- Solaris.inc
        | |-- arch64.inc
        | |-- base.inc
        | |-- default-depend.inc
        | |-- options.inc
        | |-- prod.inc
        |-- xz.spec
```



# 確認作業

```
% pkg list -a | grep mypkgs
JPCenvcmds (mypkgs)          1.0.0-0.111    installed  ----
JPCenvdev (mypkgs)         1.0.0          known     ----
JPCenvdtrace (mypkgs)     1.0.0          known     ----
JPCenveditors (mypkgs)   1.0.0          known     ----
JPCenvlangjp (mypkgs)    1.0.0          known     ----
JPCenvlibs (mypkgs)      1.0.0          known     ----
JPCenvperl584 (mypkgs)   1.0.0          known     ----
JPCenvsnmpd (mypkgs)     1.0.0          known     ----
bchunk (mypkgs)          1.2.0-0.111    installed  ----
eb (mypkgs)              4.4.1-0.111    known     ----
eb (mypkgs)              4.4.1-0.111    installed  u---
ebview (mypkgs)         0.3.6-0.111    known     ----
lv (mypkgs)              4.51-0.111     known     ----
lv (mypkgs)              4.51-0.111     installed  u---
tree (mypkgs)            1.5.2.2-0.111  installed  ----
xz (mypkgs)              4.999.8-0.111  installed  ----
```

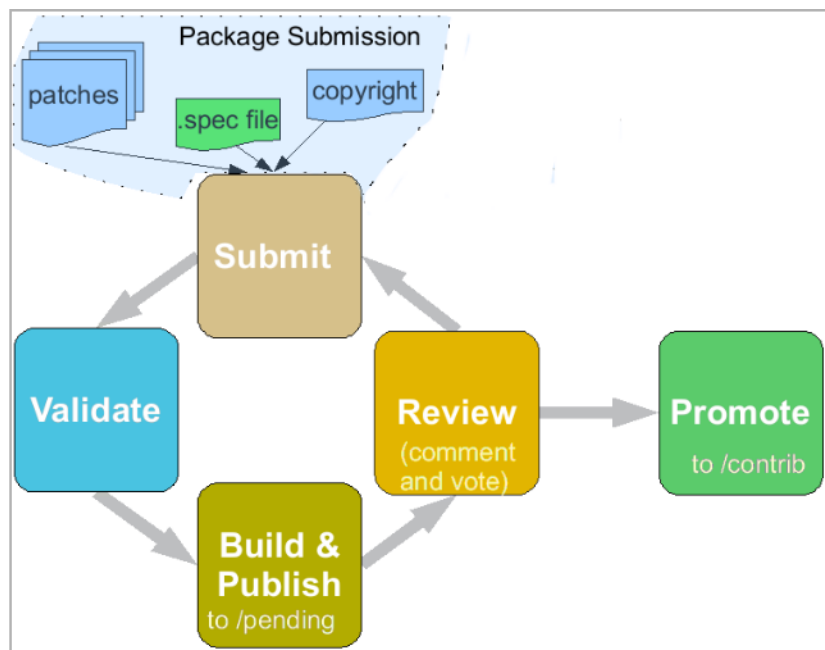
あえて書きませんが、後は普通にインストールしてみるだけです。



# opensolaris™

Contributor になろう！

# /contrib レポジトリまでの道



- spec/copyright/patch等を sourcejuicerにアップロードします (Contributer)。
- validate : copyrightファイルと、specのcopyright行と、配布元サイトのライセンス記載、アーカイブのlicenseなどを確認するようです (reviewersという特殊な役割を持った人っぽい)。
- build & publish : build gridでソースのビルドが始まります。成功すれば、勝手にpublishされます。
- review : 内容に関する投票。approversという人が行う様子。特殊なroleの人かわからない。普通に掲示板にvote(+1)してる。
- promote: /contribへ配布開始。

※pkg.opensolaris.org/pendingは廃止されたのかも？



# source juicer に登録

source juicerのURL


- <http://juicer.opensolaris.org/>

次の手順で行う。

1. [www.opensolaris.org](http://www.opensolaris.org/)にアカウントを作成する(バッチで連携するらしくしばらく時間がかかる)。
2. [juicer.opensolaris.org](http://juicer.opensolaris.org/)にログインする。
3. [spec/copyright/](http://spec.opensolaris.org/copyright/)があればpatchをアップロードし、何かしらの方法で騒ぐ。
4. 運が良いとreviewerの目にとまり、copyright checkが済む(人力作業)
5. build -> specアップロード -> build .....
6. 完成したら、voteしてもらう。
7. [pkg.opensolaris.org/contrib](http://pkg.opensolaris.org/contrib/)に入る!



# source juicer へのアップロード

 **OpenSolaris.org - log in**

**WARNING: THIS IS A BETA TEST SITE**

Only use this site for logging in to [hub.opensolaris.org](http://hub.opensolaris.org) or [juicer.opensolaris.org](http://juicer.opensolaris.org), NOT for registration. To register, please go to the [opensolaris.org registration page](#). After registering there it will take up to 24 hours before your account details appear on this site and you are able to log in here.

ANY REGISTRATIONS YOU MAKE HERE WILL BE IGNORED.

**User name**

**Password**

- Forgotten your password? [Reset it.](#)
- Got a new email address? [Change it.](#)
- Don't have an account? [Register.](#)

ログイン画面

opensolaris  
source juicer

home >> submit

Submit

Log out TAKI, Yasushi

**Identifier**  
  
(for example, "django pagination")

**Files**

/export/home/kohju/work/pkglabo/specs/ebvie

/export/home/kohju/work/pkglabo/specs/ebvie

/export/home/kohju/work/pkglabo/specs/patch

[Add another file](#)

Enter a short description that lets others easily identify what component you're submitting.

This is where you specify all the files required to build your component.

You must specify a **spec file** and a **copyright file**.

Additional files can be added by clicking the **Add another file** link, as required.

[\[more help\]](#)

[Important legal notice for employees of Sun Microsystems, Inc](#)

Use at Own Risk: As provided in the [Web Site Terms of Use](#), the Hosts may or may not pre-screen or perform compatibility testing on the Materials, and by using this repository You agree to assume all risks in Using the Materials. These risks include, but are not limited to, errors, viruses, worms, time-limited software that expires without notice, defamatory or offensive content, and the possibility that the Materials infringe or misappropriate the intellectual property rights of others.

For general discussions use [Software Porters Discuss](#) Report issues to [OpenSolaris Source Juicer Project](#)  
[Terms of Use](#) | [Privacy](#) | [Trademarks](#) | [Copyright Policy](#) | [Site Guidelines](#) | [Help](#)  
Your use of this web site or any of its content or software indicates your agreement to be bound by these Terms of Use.  
Copyright © 1995-2009 Sun Microsystems, Inc.

OpenSolaris Source Juicer: Version 1.2.0, Assembled 2009-05-26

spec、copyrightのアップロード



# My Juicer : Submissions






opensolaris  
source juicer

submit review build MyJuicer help

home >> myjuicr

My Submissions

Log out TAKI, Yasushi

Identifier	Submitter	Date	Files	Validated?	Comments	Votes
 <a href="#">ebview</a>	TAKI, Yasushi	2009-05-10	<a href="#">ebview.copyright</a> <a href="#">ebview-01-kohju.diff</a> <a href="#">ebview-02-kohju.diff</a> <a href="#">ebview-03-kohju.diff</a> <a href="#">ebview-04-kohju.diff</a> <a href="#">ebview.spec</a>	✓	7	+0 -0
 <a href="#">xz</a>	TAKI, Yasushi	2009-05-06	<a href="#">xz.spec</a> <a href="#">xz.copyright</a>	✓	4	+0 -0
 <a href="#">eb</a>	TAKI, Yasushi	2009-05-06	<a href="#">eb.copyright</a> <a href="#">eb.spec</a>	✓	11	+0 -0
 <a href="#">lv</a>	TAKI, Yasushi	2009-05-05	<a href="#">lv.spec</a> <a href="#">lv.copyright</a> <a href="#">lv-01-kohju.diff</a>	✓	1	+0 -0
 <a href="#">tree</a>	TAKI, Yasushi	2009-05-02	<a href="#">tree.spec</a> <a href="#">tree.copyright</a> <a href="#">tree-01-kohju.diff</a>	✗	5	+0 -0






自分でcommitしたものです。  
一つ「✗」があるのは、すでにdev  
に「tree」があったからです。



# My Juicer : My Comments

要するに、掲示板的な機能です。

## My Comments (Submissions I Have Commented On)

Identifier	Submitter	Date	Files	Validated?	Comments	Votes	
 <a href="#">ebview</a>	TAKI, Yasushi	2009-05-10	<a href="#">ebview.copyright</a> <a href="#">ebview-01-kohju.diff</a> <a href="#">ebview-02-kohju.diff</a> <a href="#">ebview-03-kohju.diff</a> <a href="#">ebview-04-kohju.diff</a> <a href="#">ebview.spec</a>	✓	7	+0	-0
 <a href="#">eb</a>	TAKI, Yasushi	2009-05-06	<a href="#">eb.copyright</a> <a href="#">eb.spec</a>	✓	11	+0	-0
 <a href="#">xz</a>	TAKI, Yasushi	2009-05-06	<a href="#">xz.spec</a> <a href="#">xz.copyright</a>	✓	4	+0	-0
 <a href="#">lv</a>	TAKI, Yasushi	2009-05-05	<a href="#">lv.spec</a> <a href="#">lv.copyright</a> <a href="#">lv-01-kohju.diff</a>	✓	1	+0	-0
 <a href="#">tree</a>	TAKI, Yasushi	2009-05-02	<a href="#">tree.spec</a> <a href="#">tree.copyright</a> <a href="#">tree-01-kohju.diff</a>	✗	5	+0	-0

# My Juicer :

## My Builds

JobID	Identifier	Submitter	Build Start	Build Finish	Install	Build Log	Status
<a href="#">1323</a>	 <a href="#">lv</a>	TAKI, Yasushi	May 05 22:05	May 05 22:05	<a href="#">Install</a> 	<a href="#">Log</a>	PASSED
<a href="#">1390</a>	 <a href="#">xz</a>	TAKI, Yasushi	May 09 13:05	May 09 13:05	<a href="#">Install</a> 	<a href="#">Log</a>	PASSED
<a href="#">1391</a>	 <a href="#">eb</a>	TAKI, Yasushi	May 09 13:05	May 09 13:05		<a href="#">Log</a>	FAILED
<a href="#">1403</a>	 <a href="#">ebview</a>	TAKI, Yasushi	May 11 04:05	May 11 04:05		<a href="#">Log</a>	FAILED

ビルドの状態です。Statusが微妙にバグっていて、ebはすでにbuildされていてinstallできません。

ebviewがbuildできないのは、ebviewがebに依存し、依存が必要なpackageが、まだ対応していないからです。

# 実践的な spec ファイル

```
#
# spec file for package eplib
#
# This file and all modifications and additions to the pristine
# package are under the same license as the package itself.
#
#
%include Solaris.inc

%define _prefix /usr
%define tarball_version 4.4.1

Name:                eb
Summary:             the library for accessing to the EPWING format Dictionaries
Version:            4.4.1
License:            Modified BSD
Url:                http://www.sra.co.jp/people/m-kasahr/eb/
Source:             ftp://ftp.sra.co.jp/pub/misc/%{name}/%{name}-%{tarball_
    version}.tar.bz2
Distribution:       OpenSolaris
Vendor:             OpenSolaris Community
SUNW_Basedir:      %{_basedir}
SUNW_Copyright:    %{name}.copyright
```

<- おきまりのもの

<- /usr以下に配置するパッケージ



```
BuildRoot:                %{_tmppath}/%{name}-%{version}-build
```

```
# OpenSolaris IPS Package Manifest Fields
```

```
Meta(info.upstream):      Motoyuki Kasahara <m-kasahr@sra.co.jp>
```

```
Meta(info.maintainer):   pkglabo.justplayer.com <pkgadmin@justplayer.com>
```

```
# Meta(info.repository_url): [open source code repository]
```

```
Meta(info.classification): System Libraries
```

```
%description
```

```
EB library is for accessing to the EPWING format Dictionaries
```

```
%include default-depend.inc
```

```
BuildRequires: SUNWzlib
```

```
BuildRequires: SUNWgnu-gettext
```

```
Requires: %{name}-root
```

```
Requires: SUNWzlib
```

```
Requires: SUNWgnu-gettext
```

<- 依存があるパッケージがあるときに書く。

<- ビルド時の依存のもの

<- 実行時の依存のもの

`%package root`      <- -rootパッケージは、/usr以外に作られるパッケージ。svr4的な概念。

`Summary:`      `%{summary}` - / filesystem

`SUNW_Basedir:`      /

`%include default-depend.inc`

`%prep`

`%setup -c -n %name-%version`

`%ifarch amd64 sparcv9`

`rm -rf %{name}-%{tarball_version}-64`

`cp -rp %{name}-%{tarball_version} %{name}-%{tarball_version}-64`

`%endif`

<- 64bitバイナリを作るためのもの

`%build`

`CPUS=`/usr/sbin/psrinfo | grep on-line | wc -l | tr -d ``

`if test "x$CPUS" = "x" -o $CPUS = 0; then`

`CPUS=1`

`fi`

`export CFLAGS=" $RPM_OPT_FLAGS"`

`export LDFLAGS=" %_ldflags"`

`export CC=cc`

<- コアの数

<- 環境変数の設定



```
cd %name%-tarball_version}
%ifarch sparc
%define target sparc-sun-solaris
%else
%define target i386-sun-solaris
%endif

./configure \
--prefix=%_prefix}\
--sysconfdir=%_sysconfdir} \
--libdir=%_libdir} \
--bindir=%_bindir} \
--includedir=%_includedir} \
--mandir=%_mandir}
make -j$CPUS

%ifarch amd64 sparcv9
cd ../name%-tarball_version}-64
export CFLAGS=" %optflags64"
./configure \
--prefix=%_prefix}\
--sysconfdir=%_sysconfdir} \
--libdir=%_libdir}/%_arch64} \
```

<- 32bitのconfigureオプション

<- 64bitのconfigureオプション



```
--bindir=%{_bindir}/%{_arch64} \  
--includedir=%{_includedir} \  
--mandir=%{_mandir}  
gmake -j$CPUS
```

```
%endif
```

```
%install
```

```
cd %{{name}}-%{{tarball_version}}  
gmake install DESTDIR=$RPM_BUILD_ROOT  
if test -d sun-manpages; then  
  cd sun-manpages  
  make install DESTDIR=$RPM_BUILD_ROOT  
  cd ..  
fi
```

```
%ifarch amd64 sparcv9
```

```
cd ../%{{name}}-%{{tarball_version}}-64  
gmake install DESTDIR=$RPM_BUILD_ROOT  
#rm -f $RPM_BUILD_ROOT/%{{_libdir}}/%{{_arch64}}/lib*  
cd ..  
%endif
```

<- インストール先をWORKへ

```
%{?pkgbuild_postprocess} %pkgbuild_postprocess -v -c “{%version}:{%jds_
version}:{%name}:$RPM_ARCH:(date +%Y-%m-%d):{%support_level}” $RPM_BUILD_
ROOT}
```

```
%clean
rm -rf $RPM_BUILD_ROOT
```

```
%files
%defattr (-, root, bin)
%dir %attr (0755, root, bin) %{_bindir}
%{_bindir}/*
%dir %attr(0755, root, bin) %{_libdir}
%{_libdir}/*
%dir %attr(0755, root, bin) %{_prefix}/include/eb
%{_prefix}/include/eb/*
%dir %attr(0755, root, bin) %{_prefix}/share
%{_prefix}/share/*
#%ifarch amd64 sparcv9
#%dir %attr (0755, root, bin) %{_libdir}/%{_arch64}
#%{_libdir}/%{_arch64}/lib*.so*
#%endif
```

<- ディレクトリ構造に関する定義

```
%files root
%defattr (-, root, bin)
%{_sysconfdir}/eb.conf
```

<- -rootパッケージ用のもの

```
%changelog
```

```
* Wed May 6 2009 TAKI, Yasushi <taki@justplayer.com>
- Initial Revision
```

.specファイルは、tar.gzに入っていたり、redhat等が配布されてるものがありますが、大抵はそのまま使えません。

理由は、環境に依存するものを記載するためのファイルだからです。

# porting のコツ

- 環境変数に依存するもの
  - Solarisは、Linuxに比べて、バックワードコンパチブルに対して厳密なので、古い物がそのまま残っています。
  - Solarisは、Linuxに比べて、環境の自由があるので、ライブラリやコンパイラの自由もあります。
- CPUに依存するもの
- コンパイラに依存するもの
  - Sun Studio 12を使うべきですが、gccでしかコンパイルできない物もあります。
- configureをちゃんと考えてないもの。
  - configureスクリプトを作るのはとてもめんどくさいので、DESTDIRがちゃんと動かないものもあります。その場合は、Makefile.inを修正します。
- ライブラリ、ヘッダに依存するもの
  - Socket系にあるぐらいで、互換ものをたまに作ることもあるかも……
- アレがない、コレがない(ライブラリ等)
- コンパイルしてパッケージ化してpendingに入れればOKです…といたいのが……

# その他の問題

- OSとはどこまでを示すのか?

- パッケージ依存の問題





# opensolaris™

## 公開用 ips サーバを設定する

# 公開用 IPS サーバ

IPSサーバのオプションは、`svccfg`を利用して設定します。

次のオプションがあります(参考:`man pkg.depotd`)

`pkg/content_root` ドキュメントルート。デザインを変更するときのコンポーネントはここに保存 (IPSは、`httpd`として起動している)。デフォルトは、`/usr/share/lib/pkg`

`pkg/inst_root` レポジトリの実体が入ります。レポジトリデータはこの場所にあるので、ここをコピーすることにより、レポジトリの複製が可能です。デフォルトは`/var/pkg/repo`。

`pkg/log_access` アクセスログ。

`pkg/log_errors` エラーログ。

`pkg/port` LISTENポート番号。

`pkg/proxy_base` Proxyサーバを経由したとき、実際にはどのURLになるのか。

`pkg/readonly` リードオンリー。`true`にすると、`pkgsend`できなくなる。

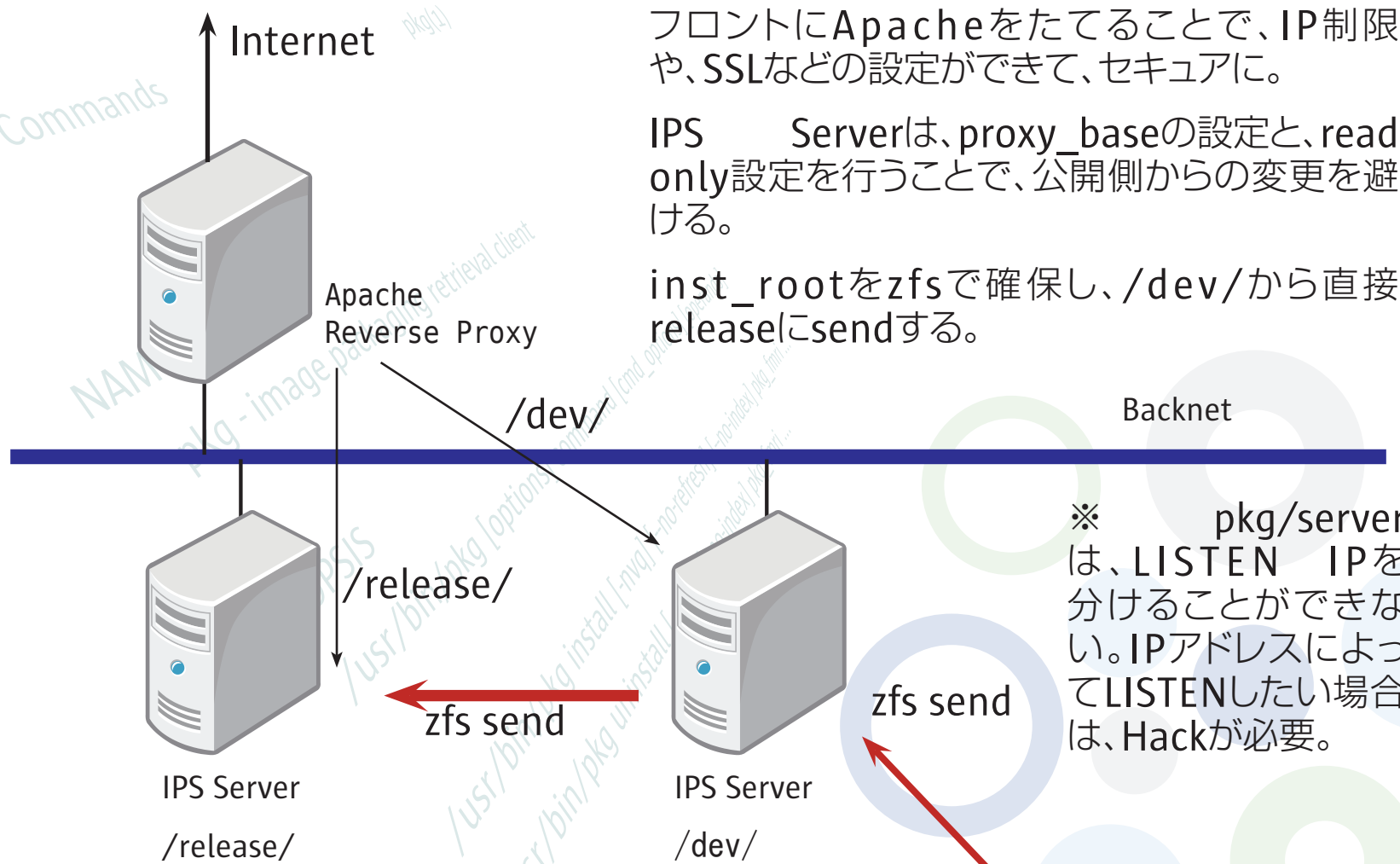
`pkg/threads` スレッドの本数。デフォルトは10なので最大同時10人しかアクセスできない。

設定変更例

```
pfexec svccfg -s pkg/server "setprop pkg/port=10000"
pfexec svcadm refresh pkg/server
pfexec svcadm restart pkg/server
```



# 公開用 IPS ネットワーク図



フロントにApacheをたてることで、IP制限や、SSLなどの設定ができて、セキュアに。

IPS Serverは、proxy\_baseの設定と、read only設定を行うことで、公開側からの変更を避ける。

inst\_rootをzfsで確保し、/dev/から直接releaseにsendする。

※ pkg/serverは、LISTEN IPを分けることができない。IPアドレスによってLISTENしたい場合は、Hackが必要。

開発用IPSより