開
放
的
열린
مفتوح
libre
मुक्त
ముక్త
livre
libero
ಮುಕ್ತ
开放的
açık
open
nyílt
⠠⠠⠠⠠⠠
פתוח
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை




USE



IMPROVE



EVANGELIZE

# Making Your Own Distro

Peter Dennis
Sun Microsystems

# Making Your Own OpenSolaris Distro

- Why would you want to do this ?
  - Because you can and it is interesting ?
  - Build a pre-configured image used for install/testing/ diagnostics ?
- OpenSolaris provides redistributable packages and so you they can be used to create distros

# Approaches to doing this

- Two methods
  - Top Down: install the system with all the required software that you need, make modifications as required (new software repos, new grub menus, remove unwanted software components)
  - Bottom Up: look for the set of software required build a list and then see if the system starts with this list
- Top Down is going to be easier – start with a working system and add/remove what is required.

# Construction OverView

- Identify the required software
- Build the image
- Construct the miniroot
- Build the grub infrastructure
- Create the iso image
- Create the usb image (if required)
- Test the images

# Two Construction Methods

- There are two construction methods
  - Manually running commands via a script or the cli and building the required image
  - Using the distro_const(1M)

# Common Elements to both Methods

- Identify the software you want to use and where to get it from – using the top down or bottom up method
- What modifications are required post install (for example grub menu settings), network settings ?

# Manually Building a Distro

- Cool thing about IPS – it allows you to build an 'image area' for the IPS packages to be placed:
  - Pkg image-create -F -a *authority ${ROOTDIR}*
  - Now install the required packages

    pkg -R *${ROOTDIR}* install *pkgname*
- Wait time.....packages to down load and install into the image

# Manually Building a Distro

- Configure the root information:
  - Initialise the smf(5) repository
  - Set up the etc/vfstab to mount the ramdisk
  - hostname/timezone/network configuration
  - Anything that might be useful (ssh ?)
- Build the miniroot
  - The miniroot is the image that is used for the boot archive by grub(5)
  - Create a file, lofiadm -a it, newfs it, mount it
  - Copy the configured root above into it
  - Umount and compress it

# Manual Creation

- Test the image, fix if required (it will) but start again or run commands again and hope the manual image has not been damaged
- It is error prone (indeed twice I wipped out my system's menu.lst because it is a link)

# And now the easy way.....

- Distribution Constructor!
- Not installed by default

  *pkg install SUNWdistro-const*
- Consists of a central command, helper commands, xml files and scripts
- It allows for the easy creation of Live media adding in IPS packages, SVR4 packages

# Building the first distro

- Build the OpenSolaris.YYYY.MM LiveCD

```
cd /usr/share/distro_const/slim_cd
distro_const build ./slim_cd_x86.xml
```

- This will result in two image files:

```
/rpool/dc/media/OpenSolaris.iso
/rpool/dc/media/OpenSolaris.usb
```

- These are the two Live images, one for a CD and one for a usb device
- That is it!

# How to use it ?

- Everything is driven from the manifest file
- The manifest is an XML file that describes the:
  - The packages that make up the distribution
  - The IPS configuration
  - Scripts that run once the image has been created
- The package comes with some predefined/ example files:
  - /usr/share/distro_const/slim_cd/*.xml
  - /usr/share/distro_const/auto_install/*.xml

# The Manifest File

- The example one for the OpenSolaris.YYYY.MM contains lots of text describing each option
- The file contains
  - The name of the resultant image
  - The list of packages that will make up the image
  - The authority to pull the packages from
  - The files that are used to make up the miniroot
  - The IPS authorities to setup once the system is installed
  - Where to build the image and associated files

# More Manifest File

- Packages to remove once the image has been built
- Finalizer information
  - A finalizer is a program (shell script, C program) that takes some arguments and does 'something' to the built image.

# Example Modifications

- Change the source package repository

  - take a copy of the example xml file and edit it
- Look for `<pkg_repo_default_authority>` and the associated value - "`pkg.opensolaris.org/release`" change it to "`pkg.opensolaris.org/dev`"
- Save and rerun the command – the packages will be taken from the 'dev' repository

# I Want Other Packages Installed

- Which repository do the packages come from ?
- If not in <pkg_repo_default_authority> then add authority to <pkg_repo_addl_authority>
- Add in extra packages add them to the <packages> list
- Add the authority to <post_install_repo_addl_authority>
- Rerun the distro_const command

# What about SVR4 packages ?

- This is slightly harder than the IPS case
- Requires a Finaliser script to do the work

# Finalizer Details

- A finalizer is a program that is executed by the distro_const program
- Five standard arguments are passed to each one:
  - MFEST_SOCKET: socket to query  manifest values
  - PKG_IMG_PATH: package image area location
  - TMP_DIR: temporary directory
  - BR_BUILD: bootroot directory
  - MEDIA_DIR: directory for the resultant media
- Plus any further defined ones in the manifest

# Finalizer Example

- Pass in arbitrary arguments in <argslist>

```
<script name="/pete/new-package/grub_setup.py">
    <checkpoint
         name="my-grub-setup"
         message="My Grub menu setup"/>
    <argslist>
         "my-splash-image.png"
    </argslist>
 </script>
```

- In this case the image is passed to the finalizer as the sixth argument

# What is this 'checkpoint' word ?

- Checkpointing is a facility that allows for the build process to be stopped/started/resumed at arbitrary points
- Checkpointing is enabled within the manifest file
- Checkpointing uses ZFS snapshots and so must be run on ZFS

# Checkpointing command line

- Various options:
  - -R start from the last successful checkpoint
  - -r start from the named checkpoint
  - -p stop at the named checkpoint
  - -l List available checkpoints

```
 # distro_const
Usage:
        distro_const build -R <manifest-file>
        distro_const build -r <step name or number> <manifest-file>
        distro_const build -p <step name or number> <manifest-file>
        distro_const build -l <manifest-file>
```

# Checkpoint listing

```
# distro_const build -l add_package.xml
/usr/share/distro_const/DC-manifest.defval.xml validates
/tmp/add_package_temp_8873.xml validates

Step            Resumable Description
--------------- --------- -------------
im-pop              X     Populate the image with packages
add-pack                  Adding Custom Packages
im-mod                    Image area modifications
slim-im-mod               Slim CD Image area Modifications
br-init                   Boot root initialization
slim-br-config            Slim CD boot root configuration
br-config                 Boot root configuration
br-arch                   Boot root archiving
slim-post-mod             Slim CD post bootroot image area modification
my-grub-setup             My Grub menu setup
post-mod                  Post bootroot image area modification
iso                       ISO image creation
```

# Check pointing

- Stop at a particular point:
  - `distro_const build -p br-init add_package.xml`
- Restart from the last checkpoint:
  - `distro_const build -R add_package.xml`
- Restart from a particular checkpoint:
  - `distro_const build -r in_mod add_package.xml`

# Debugging the process...

- Log files are generated during the build process:
  - <build_area>/logs/simple-log-...timestamp...
  - <build_area>/logs/detail-log-...timestamp
- build_area is defined in the manifest file (default is rpool/dc)
- Simple log just contains the output sent to the stdout
- Detail contains all the details including the package installation pieces

# Debugging Finalizers

- They are scripts that do stuff – they query the manifest to get data and so can you...
- /usr/bin/ManifestRead
- This allows for the querying of the manifest via the socket passed to the script to query the values
- `ManifestRead /tmp/ManifestServ.9171 "..."`

  where "..." is a parameter in the manifest:

```
distro_constr_params/pkg_repo_default_authority/main/url
```

# More debugging

- Use dtrace – I traced the pkg client calls

```
#!/usr/sbin/dtrace -qs

proc:::exec
/execname == "pkg"/
{
        printf("%s\n", curpsinfo->pr_psargs);
}
```

- Set debug options in the scripts (set -x for shell) output goes to the stdout

# Supplied Finalizers

- Locations of the finalizers in the manifest
- pre_bootroot_pkg_image_mod
  - Tidies up the image by removing some files
- bootroot_initialize.py
  - Creates the mini root area by reading the files in the bootroot contents in the manifest, creates directories
- slimcd_bootroot_configure
  - Sets the image up for the slim install (adds the user jack, sets up gdm)

# Supplied Finalizers

- bootroot_configure
  - Device, coreadm configuration, sets the name of the machine, initialises smf repo
- bootroot_archive.py
  - Creates the compressed miniroot archives and puts it in boot/boot_archive
- slimcd_port_bootroot_pkg_image_mod
  - Tidies up more files, cleans the platform tree
- grub_setup.py
  - Sets up grub menus

# Supplied Finalizers

- post_bootroot_pkg_image_mod
  - Builds the compressed zlib's on the Live image (solaris.zlib, solarismisc.zlib, pkg.zlib)
- create_iso
  - Creates the iso image using mkisofs and the pkg_img_path
- create_usb
  - Creates the usb image based off the ISO one **so** to build an usb image you must build the ISO

# Testing the Images!

- Use virtual box to test the ISO
- Write the USB image to a USB device with /usr/bin/usbcopy <path to usb image>

# References

- Home Page
  - http://opensolaris.org/os/project/caiman/Constructor
- Source code
  - http://src.opensolaris.org/source/xref/caiman/slim_source/
- Wiki
  - http://wikis.sun.com/display/OSOLInstall/Home
    - Includes example finalizer scripts
- Discuss Alias
  - caiman-discuss@opensolaris.org
- Manual Distro construction
  - http://alexeremin.blogspot.com/

開
放
的
열린
مفتوح
libre
मुक्त
ముక్త
livre
libero
ಮುಕ್ತ
开放的
açık
open
nyílt
פתוח
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை


# Thank you!

Peter Dennis

peter.dennis@sun.com

“open” artwork and icons by chandan:
http://blogs.sun.com/chandan