



# PATCHING (OPEN)SOLARIS

**Ondřej Kubečka**  
Sun Microsystems, Inc.

# What makes patch being a patch

- Sparse package(s) delivering a fix.
- There is a patch identification system.
- It can be backed out.

## How does it work (high level)

- On `patchadd(1M)` every package is refreshed using `pkgadd(1M)`; for each package modified objects along with backout information are stored in undo package.
- On `patchrm(1M)` for each patched package original content is restored by installing the backout package on its top.

# Package structure review

- pkginfo(4)
- pkgmap(4)
- install/
  - > copyright
  - > checkinstall
  - > preinstall
  - > class action scripts (CAS)
  - > postinstall
- reloc/ tree

# Patch specifics

- Packages in patches need more in order to be able to:
  - > identify patch and how does it relate to other patches.
  - > handle backout package creation.
- Besides install scripts, there is a special set used for backout (*u.\**, *patch\_\**).
- *.diPatch* marker.
- There should be a *README.<patchid>* file in patch's top level directory.

# Patch specifics - pkginfo

- pkginfo(4):
  - > PKG, VERSION and ARCH need to be identical to patched package's values.
  - > SUNW\_PATCHID is required.
  - > Optional attributes: SUNW\_OBSOLETES, SUNW\_REQUIRES, SUNW\_INCOMPAT
  - > Don't forget to have correct value assigned with CLASSES attribute to ensure all objects are delivered.

# Patch specifics – install scripts

- install/checkinstall
  - > Checks it is OK to install patch.
  - > Prepares patch related pkginfo(4) entries and stores them in the response file.
- install/preinstall
  - > Initializes backout package directories and metadata (pkginfo and install/) and related variables.

# Patch specifics – install scripts (cntd.)

- `install/i.*` (CAS)
  - > Install file objects and storing their original content in backout package.
- `install/postinstall`
  - > Processes deletes file if found in the patch package (store in backout package, run `removef` and `purge` if safe).
  - > Create the undo package using gathered data.

# Patch specifics – backout scripts

- `install/patch_checkinstall`
  - > Check that package can be backed out.
  - > Construct patch backout related `pkginfo(4)` entries (`PATCHLIST`, `PATCH_OBSOLETES`).
- `install/u.* (CAS)`
  - > Used for delivery of objects stored in backout package.
- `install/patch_postinstall`
  - > Processes deletes found in backout package.
  - > Cleans up stored backout data.

# There are also...

- Patch level scripts:
  - > prepatch, postpatch
  - > prebackout, postbackout
- Patchinfo – patch metadata used for patch automation tools.

# Don't panic!

It is quite simple...  
...a practical example.

# Tips and tricks

- Use patch compatible package scripts.
- Utilize backout package for handling the roll-back.

# “e” file backout - i.CAS

```

# "e" backout magic definitions BEGIN
DIFF="/usr/bin/diff"
OLD_PATCH_SEPARATOR="__Old_File_Diff_Separator_aBcDeFgHiJkLmN__OjK__5924894_6548915__"
# "e" backout magic definitions END

# "e" backout magic prepare BEGIN
DST_EXIST="false"
if [ "${PATCH_NO_UNDO}" != "true" ]; then
  if [ -f "${dst}" ] ; then
    DST_EXIST="true"
  else
    DST_EXIST="false"
  fi
  if [ "${DST_EXIST}" = "true" ] ; then
    #Save original file
    $MKDIR -p "`dirname $Build_Path`"
    $CP "${dst}" "$Build_Path.Orig"
    $CP "${dst}" "$Build_Path.old_and_patch"
  fi
fi
# "e" backout magic prepare END

#define i_cas() above using the original package class action script

echo "${src}" "${dst}" | i_cas

# "e" backout magic store BEGIN

if [ "${DST_EXIST}" = "true" ] ; then
  echo "${OLD_PATCH_SEPARATOR}" >> "$Build_Path.old_and_patch"
  $DIFF -C 3 "$Build_Path.orig" "${dst}" >> "$Build_Path.old_and_patch"
  if [ "$?" -ne 0 ] ; then
    #Create a repository for original file and diff in the undo package
    $ECHO "e $Class $FS_Path.old_and_patch=$Build_Path.old_and_patch" >>$BUILD_DIR/prototype
    #Make sure this file does not remain on the system after the backout
    $ECHO "$FS_Path.old_and_patch" >>"$BO_Deletes"
  fi
fi
# "e" backout magic store END

```

# “e” file backout - u.CAS

```
#!/bin/sh
PATCH=/usr/bin/patch
SED="/usr/bin/sed"
OLD_PATCH_SEPARATOR="__Old File Diff Separator_aBcDeFgHiJkLmN__OjK__5924894_6548915__"
PATCH_SAVE_SUFFIX="pre${ACTIVE_PATCH}"

get_orig()
{
    cat $1 | \
        $SED -ne "1,/${OLD_PATCH_SEPARATOR}/${OLD_PATCH_SEPARATOR}/d;p;"
}

get_diff()
{
    cat $1 | \
        $SED -ne "/${OLD_PATCH_SEPARATOR}/,\${/${OLD_PATCH_SEPARATOR}/d;p;}"
}

while read src dst ; do
    realdst="\`echo $dst| /bin/sed -e 's/\.old_and_patch$//'\`"
    if [ "$realdst" != "$dst" ]; then
        cp "${src}" "${dst}"
        if [ -f "${realdst}" ] ; then
            get_diff $src | $PATCH -o /dev/null -R $realdst >/dev/null 2>&1
            if [ "$?" = "0" ] ; then
                get_diff $src | $PATCH -R $realdst >/dev/null 2>&1
            else
                get_orig $src > "${realdst}.${PATCH_SAVE_SUFFIX}"
                if [ -f "${realdst}.rej" ] ; then
                    rm -f "${realdst}.rej"
                fi
            fi
        else
            get_orig $src > $realdst
        fi
    fi
done

exit 0
```

# Drawbacks

- Output to console not available.
- pspool and why it does not work well with patches.
- Unable to determine file type/class of a files system object.
- SUNW prefix and shared patchid space.

## More than one patch...

- There cannot be multiple patches delivering same “f” type.
- Patches are cumulative.
- “Circular” dependencies do not work.

# Where to go for more information

- Application Packaging Developer's Guide:  
<http://dlc.sun.com/pdf/806-7008/806-7008.pdf>
- For packaging also lookup Jaromir's talk:  
[http://opensolaris.org/os/project/czosug/events\\_archive/](http://opensolaris.org/os/project/czosug/events_archive/)



# PATCHING (OPEN)SOLARIS

**Ondřej Kubečka**

[ondrej.kubecka@sun.com](mailto:ondrej.kubecka@sun.com)