HP-UX 11.x Patch Management Tutorial

Bob Campbell

Address:

Hewlett-Packard Company
3404 East Harmony Road, ms 57
Fort Collins, CO 80528-9599

Phone:

(970) 898-3558

Fax:

(970) 898-2838

Email:

bcampbell@hp.com

Introduction

This paper provides an overview of the basic concepts, tools, and processes related to HP-UX patch management. It is a prerequisite for another paper presented at Interworks 2001 titled "*HP-UX 11i Patching*". Details on patch selection using the IT Resource Center (*http://itrc.hp.com*) can be found within Ann Pakenhams paper "*HP-UX Patch Management*", also within the proceedings for Interworks 2001.

The document "HP-UX Patch Management, A guide to patching HP-UX 11.x systems" addresses these and other issues. It is available from *http://docs.hp.com*.

Basic HP-UX Patch Concepts

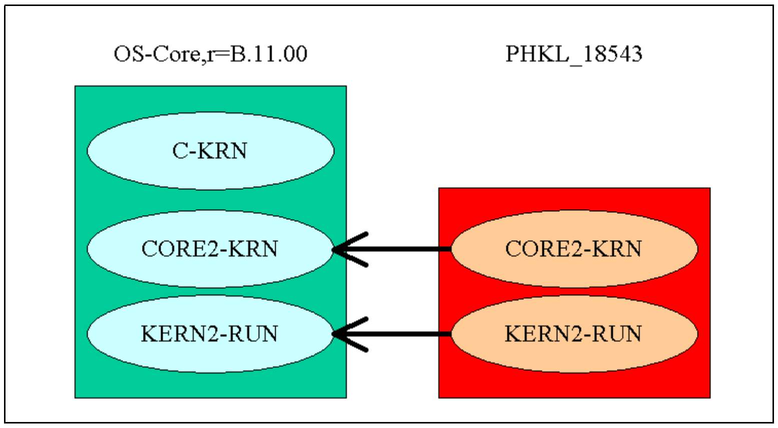
While HP-UX patches are managed through the SD-UX commands, they have certain properties that are not seen in the fully packaged product.

Ancestry

All patches for HP-UX 11.x software can only load in the presence of the original product. This is true even if the patch is a complete redelivery of all files in that product.

The original product is split into pieces known as **filesets**. All files delivered in a patch either introduce a new file or replace an existing file within the original product. All of the files within a patch that are destined to be delivered to a single product fileset will be packaged within a patch fileset of the same name. The original product fileset is said to be the **ancestor** of the equivalent patch fileset.

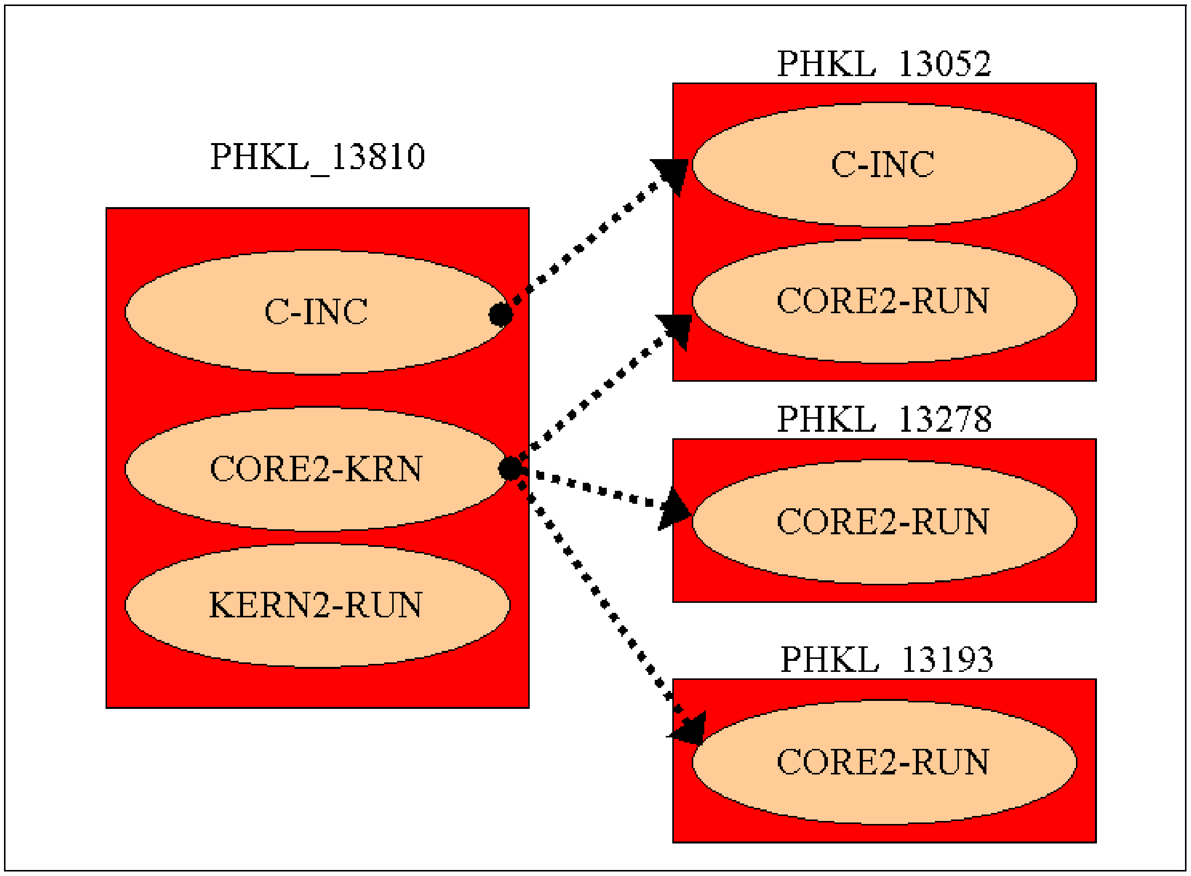
For example, let's look at the file `spinlock.o`. This file was originally delivered in the product fileset `OS-Core.CORE2-KRN`. A new version of this file was delivered within the patch `PHKL_18543`. In the diagram below, the new version of spinlock.o would be found within `PHKL_18543.CORE2-KRN`.



This diagram also shows that while the patch files will be delivered in a fileset that shares the name of its ancestor, every fileset in a product does not need to be found in one patch. Several patches may exist for the same ancestor fileset, but there are restrictions against the delivery of the same file in unrelated patches.

Supersession

While every patch fileset will specify a specific revision, there will never be another revision of that patch. When a new patch is created it provides all of the cumulative files and functionality of those it replaces. This new patch has a new patch ID, and may directly replace several preceding patches. This process is known as **supersession**.



In the preceding diagram, the newer patch `PHKL_13810` directly supersedes three other patches, `PHKL_13052, PHKL_13278`, and `PHKL_13193`. As PHKL_13810 is required to be cumulative regarding all superseded patches we can consider this diagram at the level of the patches themselves. The actual implementation of supersession occurs at the fileset level making it more precise to say that the fileset `PHKL_13810.C-INC` superseded `PHKL_13052.C-INC`. You can also see that the new `CORE2-KRN` fileset actually supersedes three others, and that `KERN2-RUN` is new and supersedes nothing.

Rollback & Commitment

When a patch is installed onto a system, by default it will preserve the original version of each file being delivered into a special save area. This allows the removal of the patch to return the system to the state before it was loaded in a process known as **rollback**.

Each patch is given a dedicated area, allowing several patches in a supersession chain to be installed and rolled back. While a useful feature, rollback comes at the cost of the disk space needed for the save area files. The disk space associated with a patch can be reclaimed by deleting the associated save area. A patch without a save area is said to be **committed**. A committed patch can only be removed through the removal or reinstallation of its ancestors.

Software Distributor

The Software Distributor (SD) tool set provides all of the utilities required to manage software products and patches on HP-UX. All utilities may be used via direct, command-line mode and several also have user interfaces built upon X-windows or terminal-based graphics.

SD Object Types

SD defines a hierarchy of objects. The following list describes the object types most useful within the realm of patches. This list is presented in order of increasing scope.

- Files
  A file is just that, an object to be delivered to a specific file system path on the target system.
- Filesets
  Filesets are a required object type. Files must be contained within filesets, and each and every product must have at least one fileset.
- Products
  A product contains one or more filesets. Each patch is packaged as a special type of product.
- Bundles
  A bundle is an optional container that may include one or more products. These are generally used as a delivery convenience. A bundle has no actual contents, but only includes products by reference.
- Depots
  All other objects are confined within depots. A depot can exist as a single file (tape-style) or as a directory structure. Patches are delivered individually as shar-archives containing a tape-style ".depot" file, while directory depots can be made available directly over networks.

SD Commands

The Software Distributor tools provide a rich and complex set of functionality that is beyond the scope of this paper to fully describe. For in depth documentation regarding these commands, consult the associated man pages or the "*Managing HP-UX Software With SD-UX*" manual available from the HP Technical Documentation web site (http://docs.hp.com).

swlist

The `swlist` command provides the ability to list the software installed on the system, including its current state and associated attributes.

1. List available depots on the system patchsvr

```
swlist –l depot @ patchsvr
```

2. List bundles installed on current system
```
swlist –l bundle
```
3. List all patches that have not been superseded on system
```
swlist –l product –x show_superseded_patches=false \
                                              *,c=patch
```
4. Invoke the swlist browser on the depot /MyDepot on the system MySystem
```
swlist –i –d @ MySystem:/MyDepot
```

swcopy

The `swcopy` command copies software from one depot to another. This allows several depots to be combined to enable installation in a single session, or for a frequently used depot to be relocated to faster media.

1. Copy the XSWGR1100 patch bundle from the CD to /MyDepot
```
swcopy –s /cdrom/XSWGR1100 XSWGR1100 @ /MyDepot
```
2. Combine the patch PHNE_21822 and its dependencies into /MyDepot
```
swcopy –s /tmp/PHNE_21822.depot \* @ /MyDepot

swcopy –s /tmp/PHCO_17622.depot \* @ /MyDepot

swcopy –s /tmp/PHCO_21596.depot \* @ /MyDepot
```

swinstall

The `swinstall` command installs software onto the local system from a local or remote depot.

1. Install all patches that match current system from the depot MyDepot on the remote system MySys without invoking the swinstall user interface:

   ```
   swinstall –s MySys:/MyDepot –x patch_match_target=true
        –auto_reboot=true
   ```

2. Install the patch PHSS_12345 from its .depot form directly to the committed state:

   ```
   swinstall –s /tmp/PHSS_12345.depot
        –x patch_commit=true \*
   ```

swremove

The `swremove` command removes software from the local system or depot.

1. Remove PHSS_12345 from the system (no kernel build or reboot anticipated):

   ```
   swremove PHSS_12345
   ```

swmodify

The swmodify command provides support for many advanced operations and should be used with caution. The exception to this rule is patch committal.

1. Commit all patches currently on the system:

   ```
   swmodify –x
   ```

swreg

The swreg command allows depots to be registered or unregistered.

1. Unregister the depot /PrivateDepot;

   ```
   swreg –l depot –u /PrivateDepot
   ```

Patch Documentation

Each patch is released with documentation that shares the patches identifier followed by the “.text” suffix. Known as the text file, it can be viewed through several different mechanisms:

- Text file
  The text file is delivered with each individual patch as a part of the original shar-archive. Generally static, this file will be modified to reflect a patch reposting or recall.

- **Patch Database**
  The Patch Database is the primary source for selecting and acquiring individual patches. A part of the IT Resource Center (*http://itrc.hp.com*), the Patch Database displays all available information related to a patch as of that date. This may include information not found in the text file.

- **The `readme` Attribute**
  When each patch is created, the original form of the text file is embedded within the product objects `readme` attribute. This form of the data is static and will never be updated. The contents of this attribute may be displayed using the swlist command:
  ```
  swlist –l product –a readme PHKL_18543 | more
  ```

Patch Text Fields

- **Patch Name**

  This identifier is used for the patch shar, text, and depot files as well as the patch product and in HP-UX 10.X releases the patch fileset. Format is PH*xx_yyyyy* where:

  PH = Patch HP-UX.
  xx =
  CO - general HP-UX commands.
  KL - kernel patches.
  NE - network specific patches.
  SS - all other subsystems: X11, Openview, etc.
  yyyyy = a unique number

- **Patch Description (1 Liner)**

  The patch description is a single text line that defines the architecture, HP-UX release, and function of the patch. For example:

  ```
  S700_800 11.00 Xserver cumulative patch
  ```

- **Creation/Post Date**

  These are the dates that the patch was created and when it was first made generally available.

- **Hardware Platform – OS Releases**

  The hardware platforms and HP-UX releases that this patch supports. After the introduction of HP-UX 11.0, all patches support both s700 and s800 platforms.

- **Products**

  The products and revisions that a patch is intended to modify. If the patch modifies part of the HP-UX core, this field is set to “N/A”.

- **Filesets**

  This section lists all of the ancestor filesets of the patch.

- **Status**

  The status field describes the current state of a patch within its supersession chain. The value of this field changes during the life of a patch, but is only modified within the patch database and text file.
  The status is a two-word field. The first word defines the patch distribution type with one of the following values:

  - **General**
    All general distribution patches were initially intended for usage by all customers.
  - **Special**
    Any special distribution patch was intended for use by a limited audience under the direction of HP support personnel. It was created to resolve some immediate need, and is not generally available or recommended.

  The second word describes the current release state of the patch. Current values are:

  - **Release**
    The patch is currently active. It has not been superseded or recalled.
  - **Superseded**
    The patch is acceptable for use, but has been replaced by a newer patch.
  - **Recalled**
    A critical defect was discovered within this patch and HP no longer recommends it for general use. The Warn field will contain detailed information regarding the issue and should be used to locally decide the best course of action.

- **Automatic Reboot**

  If this field contains "Yes", the patch requires a system reboot in order to be installed on the system. This generally indicates a kernel rebuild is necessary.

- **Critical**

  A critical patch is one which:

  - Causes the system (OS/kernel) to fail/crash/panic.
  - Causes a major application to fail such that the system's operation is severely impacted.
  - Causes data loss or corruption.
  - Delivers a fix related to processing dates in the year 2000 and beyond.

The critical field contains Yes or No, with each possibly followed by a list of all patches within the supersession chain that were marked as critical.

- **Category Tags**

  Patches may contain attributes known as category tags. Each patch is released from the factory with a certain number of predefined categories, and all patches are required to define the category of `patch`.

- **Path Name**

  The path name is the patch's relative storage location on the HP Electronic Support Center ftp server (*ftp://us-ffs.external.hp.com*). The full patch will be under the /superseded_patches directory for most superseded patches, and under /recalled_patches for all recalled patches.

- **Symptoms**

  The Symptoms field contains a description of the visible impact of the defects fixed by this patch.

- **Defect Description**

  The Defect Description field contains the detailed description on the nature of the defect or enhancement.

- **SR**

  The SR field contains all Service Request (SR) numbers that are addressed by this patch and all of its predecessors. An SR is a formal request from a customer to have a defect resolved or a feature added to HP software.

- **Patch Files/What Strings/Checksums**

  These fields define the contents and structure of the patch as well as the what strings and checksums that can be used to verify the proper installation of each patch file.

  If the patch replaces an object module in a library, the full path of the library is listed with the object module following in parentheses. For example, if a patch replaces the object module "vers.o" in the library "/usr/conf/lib/libhp-ux.a" the path listed would be "/usr/conf/lib/libhp-ux.a(vers.o)".

- **Patch Dependencies**

  This field is a list of any patches that are required for the proper operation of the patch. The policy of cumulative patches allows for the dependency requirements to be met by any patch that supersedes the listed dependency.

  Patches dependencies are not generally enforced for releases prior to HP-UX 11.11. Any patch that does not enforce dependencies on 11.11 and later releases shall be marked with the `manual_dependencies` category tag.

- **Hardware Dependencies**
  Specific system models that the patch is intended to address.

- **Other Dependencies**
  This field list dependencies that are conditional in nature, or are not a patch or system model.

- **Supersedes**
  This attribute is a list of all of the patch products replaced by this patch.

- **Special Installation Instructions**
  Patches may require user interaction or attention for a successful load. These actions are defined within the Special Installation Instructions section of the patch text.

- **Equivalent Patches**
  If similar patches have been created for other releases, they are documented here.

- **Repost**
  Occasionally a patch must be updated to reflect new information without any changes to the actual patch binary being required. In these cases the documentation will be updated and a Repost entry describing the change will be created.

- **Warn**
  If a patch is recalled, a Warn field is added that defines the issues that were responsible for the recall notice. The patch may include multiple Warn fields, and creation of a Warn field will cause the patch to be reposted.

Patch Depot Management

The ability to create depots and share them across the network provides the cornerstone of centralized patch management. The right group of patches for any given system will vary depending on application mix, risks and risk tolerance, and investment. This section will describe a process used at the fictional MD3E company.
A dedicated team is responsible for defining a recommended patch level every four months as well as providing critical fixes between these levels. Both the recommended and critical fix patches are kept in dedicated depots. All depots are built to include all dependencies, and are accessed using the `patch_match_target` option of `swinstall`.

Reactive Patch Depots

The critical fix depots are built in response to issues seen in the MD3E production systems. As issues arise, their progress is tracked by the team. When a set of patches have been found to resolve the original issue they are made available in two ways. A defect-specific depot is created, and the patches are also added to a critical fix depot. Administrators then have the option of installing the minimum change or of fixing all known issues when a problem is encountered. The critical fix depot allows the company to define the set of patches that are critical to its own systems and to share the lessons learned company-wide.

As these patches are being selected to resolve a critical issue, the patch managers are fairly aggressive. Unlike the proactive patches, they are allowed to include young, untested patches in the critical fix depot as long as they are seen to fix a locally critical issue.

Proactive Patch Depots

The triennial patch recommendations are created from a combination of standard HP bundles, ISV recommendations, and the most recent critical fix depot. The team subjects the proactive selections to a number of tests, followed by a limited release to certain production systems.

Once testing has been completed, these patches will become the internally recommended patch level for all systems. New systems are required to start at this level, and all other production systems are not allowed to be more than one year out of date.

Because this depot is being pushed out to systems across the company, MD3E has set several restrictions on its content:

- All must be recommended by HP (test level 2 or 3)
- None can be listed as recalled
- A minimum of 3 weeks internal testing must be completed without issues
- No operator interaction is required for installation (no special installation instructions require manual operations)

Once created, the proactive depot remains unmodified. Any fixes are provided in the critical fix depots.