A Fast and Intuitive Online Help System

The HP Help System provides application developers with the tools to create and integrate rich online help information into their OSF/Motif-based applications.

by Michael R. Wilson, Lori A. Cook, and Steven P. Hiebert

With the growing complexity of today's UNIX*-operating-system applications, and the desire to improve usability, media-rich information is becoming more pervasive in today's computing environments. Users expect some base level of online help to be provided from within the applications they are using. They expect online information to be intuitive and graphical, with growing expectations of direct audio and video support and interactive capabilities.

The HP Help System is a good start towards providing multimedia online information that is both fast and intuitive. It has become the standard online help system within HP and is used extensively by the HP VUE and HP MPower products and many other OSF/Motif-based products.

Background

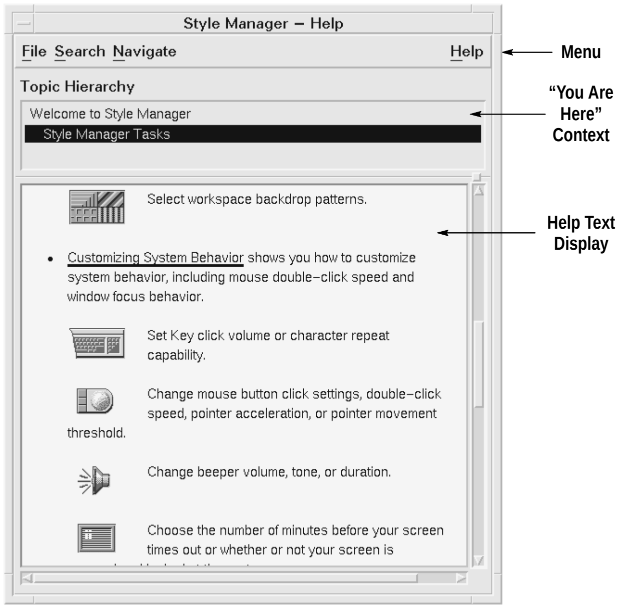
The current HP Help Developer's Kit is the second attempt by the HP VUE program to deliver an application help system for everyone. The first version, HP Vuehelp 2.0, while satisfying some of the HP VUE requirements as an application help system, failed to meet application developers' requirements for features, performance, and ease of integration.

One of the design goals for the HP Help System was to deliver a complete solution to developers for creating, integrating, and shipping rich online information with their OSF/Motif-based application, while keeping its presence (system resource use) to a minimum. There is a very fine line between providing the rich set of features that our customers require and maintaining our performance objectives. In 95% of the cases in which the issue of features versus performance came up, performance won.

Based on the knowledge and insight our team gained in doing the first version of the HP Help System, and a willingness to make radical changes in our second release, we basically started from scratch. We knew that our next-generation help system needed to provide a competitive set of base features and functionality that developers require, while producing little or no end-user-visible performance degradation to the hosting application.

The Help Developer's Kit

The HP Help Developer's Kit is a complete system for developing online help for any OSF/Motif-based application. It allows authors to write online help that includes graphics and text formatting, hyperlinks, and communication with the application. It provides a programmer's toolkit allowing developers to integrate this rich online help information into their client application. The help dialog widget (Fig. 1) serves as the main display window for the HP Help System. A second, lighter-weight help widget (quick help dialog) is also available in the toolkit.



Welcome to Style Manager
Style Manager Tasks

Select workspace backdrop patterns.

- Customizing System Behavior shows you how to customize system behavior, including mouse double–click speed and window focus behavior.

Set Key click volume or character repeat capability.

Change mouse button click settings, double–click speed, pointer acceleration, or pointer movement threshold.

Change beeper volume, tone, or duration.

Choose the number of minutes before your screen times out or whether or not your screen is

Fig. 1. Help dialog widget.

Following is a list of the components supported in the developer's kit:

- For Authors
  - The HP HelpTag markup language. This is a set of tags used in text files to mark organization and content of online help information. HP HelpTag is based on SGML (Standard Generalized Markup Language).
  - The HP HelpTag software. This is a set of software tools for converting the authored HP HelpTag files into run-time help files that contain the text for the help messages displayed in the help widgets.
  - The HelpView application. This is a program that allows an author to test a newly developed online help facility.
- For Programmers
  - The `Xvh` programming library. This library provides an application programming interface for integrating help windows into an application.

○ A demonstration program. This is a simple example that shows how to integrate the HP Help System into an OSF/Motif application.

General Packaging Architecture

An online help system needs to feel like it is part of the host application to the end user, not an appendage hanging off to the side. For developers to leverage a third-party help system, it must be delivered in such a way as to provide easy and seamless integration into their application. Furthermore, the effort and overhead of integrating and redistributing this help system along with their application must be minimal, while at the same time meeting the application's and end-user's requirements for help. Users should feel as if they have never left the application while getting help.

During our initial prototyping of the current HP Help System, we kept stumbling on the same two key issues: performance (how to make the system light and fast) and packaging (how to make it easy to integrate into any OSF/Motif-based application and redistribute with that application). Our initial help system suffered greatly in both of these areas. HP Vuehelp 2.0 was server-based, large, slow, and dependent on the HP VUE desktop. Any application using our help services had to run within the HP VUE desktop environment.

We addressed these two issues with our current release and ended up with a very different package. To fix the performance problems of slow startup times and heavy server-based implementation, we started copying functionality from the help server into the client via a linked-in help library. While prototyping this architecture, we quickly realized that we were duplicating services. However, we were also getting much better performance from the new client code. At that point we realized that if we moved everything to a help library we could fix our two biggest problems: performance and packaging.

By moving to a help library and removing our hard-wired dependencies on the HP VUE desktop and by bundling our product with HP's User Environment Developer's Kit, we cleaned up our previous dependency problems with HP VUE. Now developers can embed help directly into their application and ship a single executable that includes both. Developers only have to link their OSF/Motif application with the HP Help System library and they are ready to go. All dependencies on external system and desktop services have been removed. Fig. 2 shows an overview of our old and new help system architectures.

Following are some of the advantages we gained by moving from our initial HP Vuehelp 2.0 server-based implementation to an embedded client-side architecture.

- Integration Advantages:
  - ○ OSF/Motif-based application program interface (simple to use for developers familiar with OSF/Motif)
  - ○ Complete application control over the help system dialog management including creation, destruction, caching, and reuse
  - ○ Smooth transition into the help system via consistent resource settings between the application and the help system (e.g., same fonts and color scheme and quick response times)
  - ○ Immediate support for a tightly coupled application-to-help system environment (e.g., application-defined and processed help controls such as hypertext links)
  - ○ Application-specific customization of the help system.
- Packaging Advantages:
  - ○ Eliminates installation and version problems (Since applications are not sharing the help services, the contention between older or newer software versions does not exist.)
  - ○ Eliminates distribution problems. (The help system is linked directly into the host application, tested and shipped as one executable.)
- Performance Advantages:
  - ○ Requires less overall system resources (RAM and disk) for a single application using the help services
  - ○ Provides much faster initial response time when displaying help requests.

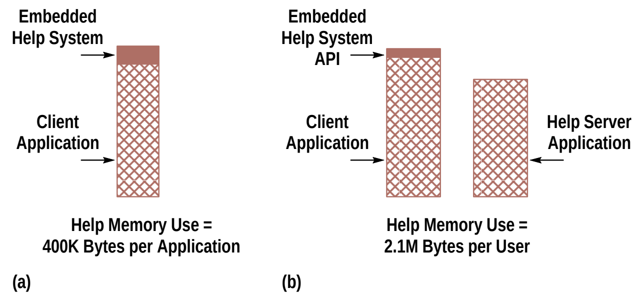


Fig. 2. Two different approaches to integrating online help into an application. (a) Client side embedded implementation using HP VUE 3.0 help. (b) Help server implementation using HP VUE 2.0 help. Our initial prototypes revealed that the client side embedded solution is better in terms of performance (rendering time) and memory use.

Integration Concepts, Practices, and Mechanisms

As mentioned in the previous section, the run-time help facility is made up of a collection of help dialog widgets and files containing online help information. The help widgets are linked directly into the client application via the help library libXvh.a* and instantiated by the client to display help information. While the help dialogs serve only as vehicles for displaying online help information, standard OSF/Motif, Xlib, and X toolkit (Xt) services provide the glue to integrate the dialogs into the application.

The online help files in the HP Help System are called help volumes. These volumes contain the text for the help topics that are displayed in the dialog widgets. The following files, or volumes are used by the HP Help System:

- volume.hv. This is the master help volume file accessed by the HP Help System when a user makes a request for help information. Information stored in this file is used to access the actual help topics stored in the help topic (.ht) files.
- volume.hvk. This is a keyword index file for the master help volume.
- volumeNN.ht. These are the help topic files, where NN represents file numbers (00, 01, 02 ...). If there are no chapter elements in a help volume, only a single topic file (e.g., volume00.ht) will exist. These are the files containing the help text.

The relationship between these files is shown in Fig. 3.

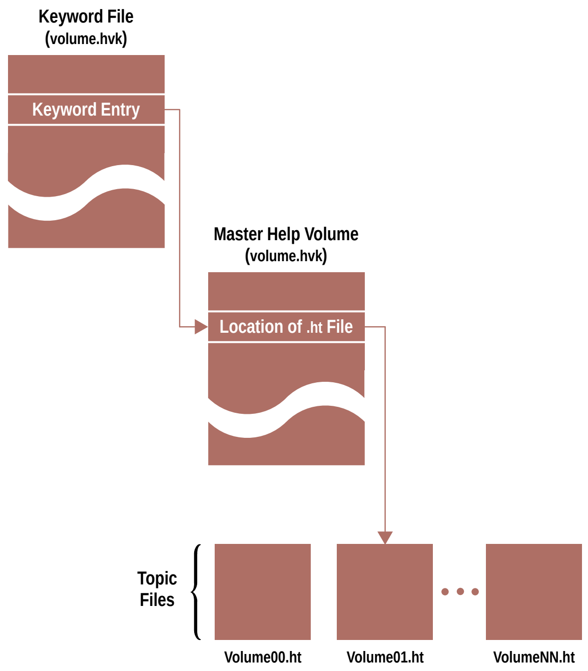


Fig. 3. The HP help file system.

There are two levels of integration with respect to the HP Help System: integrating help into an application and integrating a help-smart application into an HP VUE or HP MPower desktop environment.

Integrating Help into an OSF/Motif Application

Developers have many degrees of freedom with respect to how much or how little help capability they include in an application. If an application and its help information have very loose ties, there may be only a handful of topics that the application is able to display directly. In contrast, the application could provide specific help for nearly every object and task in the application. This requires more work, but it provides potentially greater benefits to the end user.

Help Dialogs. Two help widgets are supported in the help library: quick help dialog and general help dialog. They both support the same text, hypertext, and graphics display capabilities, but differ with respect to the remainder of the user interface. The quick help dialog (Fig. 4) is a very simple help dialog intended for displaying small blocks of text to the user. Quick help dialog is best used for handling things like error messages, version information, and object and item definitions.

The general help dialog, which is the most commonly used help dialog, has a few more user interface features than the quick help dialog widget. Most notably, the Topic Hierarchy list (see Fig. 1), which appears just above the help text display area, indicates the location of the current topic in the help topic hierarchy. The general help dialog also provides various navigational aids to assist the user in moving about the online help information space.

Standard Xt Paradigm. The programmer interacts with the help dialogs in the same manner as any other OSF/Motif widgets used by the application. The two types of help dialogs are defined by the following two widget classes:

- XvhQuickHelpDialogWidgetClass (for quick-help dialog)
- XvhHelpDialogWidgetClass (for general-help dialog).

Nearly every attribute of the help windows including the volume name and topic identifier are manipulated as widget resources. For instance, to display a new topic, an XtSetValues() call is made to set the volume, location identifier, and help type resources.

Creating Help Entry Points

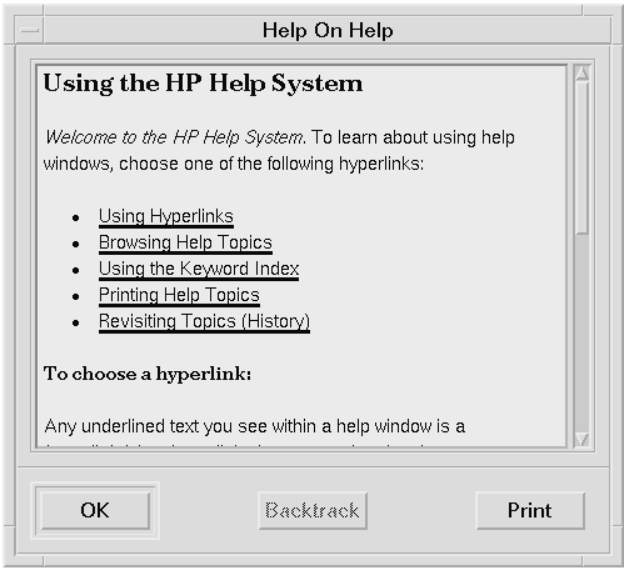
Each help topic that can be displayed directly as the result of a help request is called an entry point. That is, if there is at least one way to get directly from the application to a help topic, then that help topic is an entry point into help. Help menus and buttons in the application are the basic help entry points for an application. The types of help requests that provide entry points into the HP Help System are contextual help and item help.

Contextual Help. Contextual help provides help information about the item on which the selection cursor is positioned. Contextual help information provides users with information about a specific item as it is currently used. The information provided is specific to the meaning of the item in its current state. For example, suppose a user is running an application that uses an options widget that has four options. If the user requests information on the widget while it has option 1 selected, the user will get help information on the option widget in the context of its option 1 setting.

A selection cursor, which is a visual cue, enables users to indicate with the keyboard the choice with which they want to interact. It is typically represented by highlighting the choice with an outline box.

The OSF/Motif user interface toolkit, through its help callback mechanism, provides direct support for contextual help entry points. When a valid help callback is added to a widget, and the user presses the help key (**F1**) while that widget has the current keyboard focus (e.g., selection cursor), the widget's help callback is automatically executed.

From within the help callback, the application has the opportunity to display some help topic based on the selected


Help On Help

Using the HP Help System

Welcome to the HP Help System. To learn about using help windows, choose one of the following hyperlinks:

- Using Hyperlinks
- Browsing Help Topics
- Using the Keyword Index
- Printing Help Topics
- Revisiting Topics (History)

To choose a hyperlink:

Any underlined text you see within a help window is a

Fig. 4. Quick help dialog box.

widget, or the application could dynamically construct some help information based on the current context of the selected item.[1,2]

Any level of granularity can be applied when adding help callbacks to an application's user interface components. They can be added to all the widgets and gadgets within the application dialogs, the top-level windows for each of the dialogs, or any combination in between.

If the user selects **F1** help with the selection cursor over a widget or gadget that has no help callback attached to it, the OSF/Motif help callback mechanism has a fallback mechanism for providing more general help. The help callback mechanism will jump to the nearest ancestor that has a help callback assigned and invoke that callback. The theory is that if there is no specific help on that widget or gadget, then it is better to provide more general help than none at all. Application developers are responsible for adding their own help callbacks to the user interface components in their application. OSF/Motif sets these callbacks to NULL by default.

Item Help. Item help allows users to get help on a particular control (e.g., button, menu, or window) by selecting the item with the pointer. Item help information should describe the purpose of the item for which help is requested and should tell users how to interact with that item. An item help request does not provide context sensitive information like the current state of the selected item.

Item help is usually accessed via an application's Help menu under the On Item menu selection. Once selected, the cursor is replaced with a question mark cursor. The user can then select the item of interest.

The HP Help System API utility function, XvhReturnSelectedWidgetId() assists developers in providing item help within their application. This function provides an interface for selection of a component within an application. XvhReturnSelectedWidgetId() returns the widget identifier for any widget in the user interface that the user has selected via the pointer.

At any point while the question mark cursor is displayed the user can select the escape key (**ESC**) to abort the function call. If the user selects any item outside the current application windows the proper error value will be returned.

Once XvhReturnSelectedWidgetId() returns the selected widget identifier, the application can invoke the help callback on the identified widget or gadget to process the selected item. From the help callback the application can display some help topic based on the selected widget or dynamically construct some help information based on the current selected item.

Integrating a Help-Smart Application

There are no restrictions regarding where run-time help files are installed. However, a suggested practice is to package a help volume in a separately installable file set so that the system administrator can place them on a system file server. This will save local resources and make the help information available to a larger number of users. The default configuration is for the run-time files to be installed with the rest of the application's files.

Another important step in installing help files is registration (or symbolic links to help volumes). The registration process

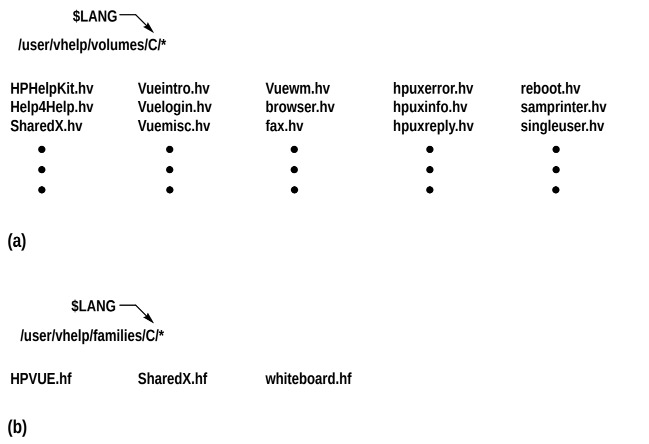


Fig. 5. Help directory contents. (a) A portion of a directory containing help volumes. (b) A directory containing product families.

enables two important features of the HP Help System: cross-volume hyperlinks and product family browsing.

Registering a Help Volume. After the run-time files have been installed, a help volume is registered by creating an HP-UX* symbolic link to the help volume's volume.hv file. The link is created in one of the directories that the help system searches for help volumes (Fig. 5a). For most help volumes, the appropriate place for the link is in the /usr/vhelp/volumes/$LANG/ directory, where $LANG represents the language of the help volume being registered. The default language for help files is usually English.

Registering a Product Family. A product family is a group of help volumes belonging to a particular product. To register a product family, a help family file (product.hf) must be created with the rest of the product's help files. The family file is registered by creating a symbolic link to the product-name.hf file. For most products, the appropriate place for the link is in the /usr/vhelp/families/$LANG/ directory (see Fig. 5b).

Family files can be read with the helpgen program (part of HP VUE), which creates a special help volume that lists the families and the volumes within each family installed on the system.

Access to Help Volumes

The HP Help System has a simple, yet extensible mechanism for transparent access to help volumes installed on the desktop. It supports both local and remote access to help volumes and works with any number of workstations. The only dependencies required are the proper help registration as discussed above, NFS services (i.e., remote systems mounted to the local server), and proper configuration of the help environment variables discussed below.

When an application creates an instance of a help widget the XmNhelpVolume resource can be set using either a complete path to the volume.hv file, or if the volume is registered, using the base name (i.e., the file name without the .hv attached) of the volume. When using the base name, the help system searches several directories for the volume. The search ends when the first matching volume.hv file is found. The value of the user's $LANG environment variable is also used to locate help in the proper language (if it's available).

The environment variable XVHHELPUSERSEARCHPATH specifies the user search path for locating help volumes. Whenever this resource is set to a relative path, the user's default home directory will be prepended to the value contained in the XmNhelpVolume resource. The default value used when the environment variable is not set is:

```
vhelp/%T/%L/%H.hv:\
vhelp/%T/%H.hv:\
vhelp/%T/%L/%H:\
vhelp/%H:\
vhelp/%T/C/%H.hv:\
vhelp/%T/C/%H:
```

where:

%L is the value of the $LANG environment variable
(C is the default)
%T is the type of file (volume or family) being searched for
%H is the help volume specified.

These path names are illustrated in Fig. 5.

The environment variable XVHHELPSYSTEMSEARCHPATH specifies the system search path for locating help volumes. The default value used when this environment variable is not set prepends the string /usr to the strings given above. For example, vhelp/%T/%L/%H.hv:\ becomes /usr/vhelp/%T/%L/%H.hv:\. Note that the user search path defined above takes precedence over the system search path.

Run-Time Help Volumes

The flexibility and power of this help system is largely placed in the author's hands. With the HP HelpTag markup language and a creative author, very different and interesting approaches can be taken with respect to presenting information to the end user. Documents can be organized in either a hierarchy with hyperlinks referencing the children at any given level, or in the form of a network or web, with a linear collection of topics connected with hyperlinks to related topics (see "Information Models for Online Help" on page 92 for more about these document organizations). It's up to the author to explore the many capabilities with respect to producing effective online help information for a particular system.

Help Volume Structure. A help volume is a collection of related topics that form an online book. Normally, the topics within a volume are arranged in a hierarchy. Developers usually create one help volume per application. However, for complex applications or a collection of related applications several help volumes might be developed. Topics within a help volume can be referenced by unique location identifiers that are assigned by the author. It is through these location identifiers that help information is referenced in the run-time environment.

Help Volume Authoring. Online help is written in ordinary text files. Special codes, or tags, are used to mark up elements within the information. The tags form a markup language called HP HelpTag. The HP HelpTag markup language defines a hierarchy of elements that define high-level constructs such as chapters, sections, and subsections, and low-level elements such as paragraphs, lists, and emphasized words (Fig. 6). The text files created by authors and any associated graphics files are compiled using the HP HelpTag software to create run-time help files (Fig. 7). These run-time files (or volumes) are installed with the application and accessed when the user requests help.

```
<!entity versionGraphic FILE "bike.bm">

<metainfo>

 <title>Helpdemo (Sample Application)
 <copyright>

    <graphic entity=versionGraphic><term nogloss|Helpdemo|, Version 1.0

    <image>
    &copy; Copyright Hewlett-Packard Company 1992
    All Rights Reserved.
    <\image>

    !!This program is for demonstration purposes only! !!

 <abstract>This online help volume is used with the 'helpdemo' program
         that demonstrates how to use the HP Help System in an OSF/Motif
          application.

<\metainfo>

<hometopic>Helpdemo Help Information
          <idx|introduction|

This is the home topic. This is the top level of your helpdemo
help information.

Choose one of the following links to find out more about the helpdemo
program.

<list bullet>
  * <xref chap1ID>
  * <xref chap2ID>
<\list>

<chapter id=chap1ID>An Application Help System

<list bullet>
  * <xref onApplicationMenu>
  * <xref sampleBtnOne>
  * <xref sampleBtnTwo>
<\list>


<s1 id=sampleBtnOne>Button One Help

Here's the help text for our sample button one.

<s1 id=sampleBtnTwo>Button Two Help

Here's the help text for our sample button two.

<s1 id=onApplicationMenu>Introducing Helpdemo

This is the topic displayed by choosing On Application from the Help menu.

<chapter id=chap2ID> Controlling The Application

The following links demonstrate how the HP Help System can control the
hosting application.

<list bullet>
  * <link hyperlink="100" type=AppDefined> Move the window UP <\link>
  * <link hyperlink="101" type=AppDefined> Move the window DOWN <\link>
  * <link hyperlink="102" type=AppDefined> Move the window LEFT <\link>
  * <link hyperlink="103" type=AppDefined> Move the window RIGHT <\link>
<\list>
```

Note: The text contained in the brackets "< >" is the tags that form the HP HelpTag markup language.

Fig. 6. A sample HP HelpTag volume that is distributed as part of the HP Help Developer's Kit.

Help Volume Compilation. The HP HelpTag compilation process performs the following tasks in generating a compiled run-time help volume:

- Syntax validation
- Conversion from the authored format to run-time format
- Location identifier map generation
- Topic compression
- Topic hierarchy map generation.

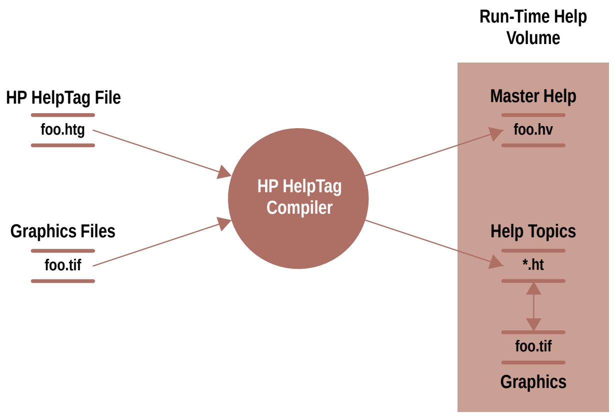


Fig. 7. The HP HelpTag compilation process.

While creating the help volume compilation process we developed techniques for improving the performance (speed and size) of our system. Our objectives were to create a run-time file format that supported very quick access (three seconds from request to display) for any requested help topic contained within a help volume, while at the same time keeping the overall size of the volume as small as possible. Topics with graphics can sometimes be slower than three seconds.

To solve the quick access problem, a scheme was devised that takes the physical hierarchy of authored topics within the volume and flattens the whole volume. During this process a jump table is generated that lists each topic name, the file that topic resides in, and the offset into the file that represents the beginning of the topic. With this flattened format and the jump table, topics can be quickly retrieved for display. The jump table is stored in the volume.hv file in X resource format, and parsed using standard xrm function calls in Xlib.

Help Volume Compression and Decompression. The size of compiled run-time help files was a real problem in the first prototypes. The solution we developed to solve this problem involves compressing the topics in the help topic (*.ht) files. This task was added as the last step in the compilation phase of an authored help volume.

The help data files are compressed on a per-topic basis. That is, each help file is read to determine the start and end points of each topic in the data file. This information is used to step through the data file, extracting each topic in turn, writing it out to a temporary file, and using the HP-UX utility compress(1) to compress the file. Unless a special option (–f) is given to compress, the utility will not compress a file if the file does not actually shrink. (The HP Help System is able to read uncompressed files.) After executing the compress utility, the resulting file is checked for a .Z extension to determine if compression has actually taken place.

If the topic is compressed, a null byte followed by three bytes making up a 24-bit number holding the size of the topic in bytes is written to a new version of the help data file followed by the compressed topic. If the topic is not compressed, it is simply copied to the new help data file. In either case, a new version of the index file (volume.hv file) is updated with the new starting offset of the topic based upon the size of the previous topics.

Since no uncompressed topic will ever start with a null byte, the leading null byte in compressed topics serves as a "magic number" to indicate to the run-time help viewer that the topic is compressed. The 24-bit size value written to the file in a fixed byte order allows help files to work on machines with varying byte orders in machine words and is used by the decompression algorithm in the run-time help viewer to determine when to stop expanding a topic.

After the entire help data file has been compressed, the new versions of the help data file and index file are moved to replace the old versions.

When the compression algorithm was first prototyped, we were pleasantly surprised by the results. The new scheme saved over 40% in disk use per volume while creating no end-user-visible performance degradation in rendering time. We use the standard HP-UX compress(1) command, which is called from the HP HelpTag program to handle the compression, and an embedded library version of uncompress for decompression at display time. We determined that the lack of performance degradation in using compressed topics is partly because most of the help topics are very small and the embedded decompression function enables fast retrieval.

Graphics Compression and Decompression. Support is also included for compressed graphics including X pixmaps, X bitmaps, and X windows, as well as JPEG compressed TIFF images, which are supported by the HP Image Library. See the article on page 37 for more about the HP Image Library. In most cases the best results, both for performance and disk use, are gained by using JPEG compressed TIFF images.

Help Dialogs

From the developer's perspective the help dialog is seen as a single widget. Widgets are created, managed, and destroyed as one single object. In reality our help widgets are not one monolithic help entity, but two separate very distinct components: the text display engine component and the widget component. The text display engine component as seen from the developer's perspective is the region in the help widget that renders the text and graphics, provides hyperlink access, and performs dynamic formatting of the text. The widget component consists of the OSF/Motif code that makes it a widget and the remainder of the user interface including topic hierarchy, menu bar, and supporting dialogs (print, keyword search, and history).

The Help Widget. By building the help dialogs as true OSF/Motif widgets (customized or part of a toolkit) and exposing this as our help API, we immediately gained an industry-standard format for our help functions. The syntax and use model for programmers is the same for every OSF/Motif widget created. This makes it very easy for developers familiar with OSF/Motif programming to understand and use the help widgets.

The OSF/Motif-based API supports the various controls we need to manage an instance of our help dialog. Through standard OSF/Motif and X toolkit and Xlib functions, programmers have direct access to the help dialog resources and can manipulate these values as they see fit. Developers can add callbacks via XtAddCallback(), set and modify resources via XtSetValues(), manage and unmanage the dialogs via XtManageChild and XtUnmanageChild, and free the resources when done via XtDestroyWidget().

The Display Area. Text formatting on the display allows different types of text. The author can specify dynamic text that will format or reformat text according to the window size. The author can also specify static text that will not be reformatted to adjust to different window sizes. For dynamic text a sequence of two or more spaces will be compressed into a single space and internal newlines (**Return**s or **Enter**s) will be changed into a space. For static text all spaces and internal newlines will be honored.

Help system users demanded the ability to resize help windows. While vertical scrolling is accepted as necessary when help topics are longer than the available screen space, horizontal scrolling is not. Therefore, dynamic reformatting is a must.

Foreign Language Format. Text formatting for foreign languages placed some special demands on our display area text formatting widget.

- European (8-bit) rules. For most European languages (including English), breaking a line of text at spaces is sufficient. The only other line-breaking rule applied is with hyphens. If a word begins or ends with a hyphen, the hyphen is considered a part of the word. If the hyphen has a nonspace before and after it, it is considered a line breakable character and anything after it is considered safe to place on the next line.
- Asian (16-bit) rules. For Asian language support, breaking a line of text on a space is unacceptable since some 16-bit languages do not break their words with spaces (i.e., Japanese and Chinese, but not Korean). With the Japanese language, the characters are placed one after another without any word breaking character because each character is considered a word. There is also the concept that certain characters cannot be the first character on a line or the last character on a line. English (8-bit) characters can be mixed with 16-bit characters.

Given these considerations, the following line-breaking rules for 16-bit languages are used:

1. Break on an 8-bit space.

2. Break on a hyphen if the character before and after the hyphen is not a space.

3. Break on an 8-bit character if it is followed by a 16-bit character.

4. Break on a 16-bit character if it is followed by an 8-bit character.

5. Break between two 16-bit characters if the first character can be the last character on a line and the other character can be the first character on a line.

Rather than hard code the values of those Japanese characters that can't start or end a line into our help system, a message catalog system is used. This provides a general mechanism for any 16-bit language. All language localization support people are required to determine which characters in their language cannot start or end a line and localize the appropriate native language support (NLS) file. The file /usr/lib/nls/%T/%L/fmt_tbl.cat contains the values of those characters that are used for rule 5 of the line-breaking rules. If this file does not exist for a given language, rule 5 is ignored and line breaking will occur between any two 16-bit characters.

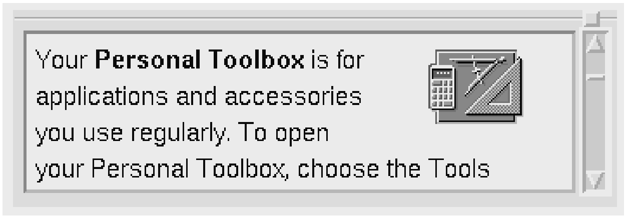

Your **Personal Toolbox** is for
applications and accessories
you use regularly. To open
your Personal Toolbox, choose the Tools

Fig. 8. Text flowing around a graphic.

Even using these line-breaking rules, sometimes the lines are still too long to fit in the available space. For this case or when static text pushes the boundary of the display area, the help system gives up and displays a scroll bar so that the user can see the information without having to resize the window.

The help system uses the same routines for processing English or multibyte documents. These routines determine what line-breaking rules to use based on the $LANG environment variable and the character set used in the document.

If a document specifies an ISO-LATIN1 character set, the display engine does not bother looking at rules 3 to 5 or using multibyte system routines. Only when the document and the $LANG environment variable specify a 16-bit character set are these rules and routines used. This minimally impacts the access and rendering time but allows the same binary to be used in the United States, Europe, and Asia.

Flowing Text. The help system display area also provides the ability to flow text around a graphic. This is seen as a space saving measure and a highly desired feature. The graphic can be placed on the left side or the right side of the display area and the text occupies the space to the side of the graphic. If the text is too long and does not fit completely in the space beside the graphic, the text wraps below the graphic. Fig. 8 shows an example of this graphics formatting technique.

Graphic Manipulation. Complaints about the text-only nature of HP Vuehelp 2.0 strongly demonstrate the truth of the adage "one picture is worth a thousand words." Therefore, the help system tries to allow as many standard graphic formats as possible. During the design of the help system the question was, which ones to allow? Eventually, it was determined to allow standard X graphics (plus the new X pixmap) and TIFF image formats. The reason for selecting the X graphic formats is that they are supported by standard Xlib routines for accessing graphics files, and the reason for choosing TIFF format is that the HP Image Library supports TIFF format. The formats supported by the help system include:

- X bitmaps
- X window files
- X pixmap files
- TIFF 5.0.

Graphic Compression. While JPEG compression schemes are common for use with TIFF files, no compression was being used with the X graphical formats. After numerous complaints from authors about how much space Xwd files

(continued on page 88)

WYSIWYG Printing in an X Application

The HP Help System provides several features that allow authors to specify different fonts in various sizes and combine them with graphics bitmap illustrations. These capabilities and others created special challenges when it came to WYSIWYG (what you see is what you get) printing. The HP Help System developed some sophisticated techniques for handling WYSIWYG printing.

What Is WYSIWYG?

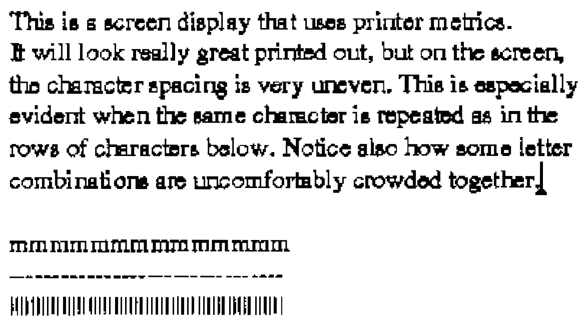
Our market research uncovered a strong demand for WYSIWYG printing without a clear definition of what WYSIWYG really meant. One obvious interpretation is the screen dump that accurately represents on paper what is on the screen. This is appropriate for some applications, but prototypes of this approach showed jagged characters crammed into a small 4-by-6-inch window on a large 8½-by-11-inch piece of paper. This satisfies no one. Another approach is to take a printed image and display it on the screen matching line breaks as they appear on the printer. This approach compromises readability on the screen, which is not acceptable for an application providing online information. Fig. 1 shows one interpretation of WYSIWYG printing.

The interpretation of WYSIWYG we chose for our help system is based on the notion that "WYSIWYG is a state of mind." True WYSIWYG is undesirable, but an appropriate illusion of WYSIWYG is the right answer. For the HP Help System, we give presentation on the screen and presentation on the printer separate consideration. The printed page uses the same fonts as the screen (sans serif for the body of the text and serif fonts for titles), but printer fonts are laid out and rendered at 300 dots per inch. The help topics are laid out to fit on the size of the printed page, and each page has page numbers. Thus, line breaks and page breaks on paper do not match line breaks and screenfuls on the display, but this is just what our customers are looking for. Fig. 2 shows a comparison of screen and printer output.

Font Availability

We took care to allow the HP Help System to work with any X server. This was important to us, since we knew our work was likely to be ported to other platforms besides the HP-UX operating system. What is more, the distributed nature of X means that even though the application may run on an HP-UX computer, it can be displayed on a totally different device, such as an HP Vectra PC running an X server under Microsoft® Windows. It is difficult to predict what fonts might be available to the server.

To enable the help system to use fonts available on any display server as well as those built into HP LaserJet printers, we developed a table that maps generic help fonts to specific fonts available on the target server or printer. After having studied a variety of HP documentation, we identified a need for three font families for help messages: a serif proportionally spaced family (like Times), a sans serif proportionally spaced family (like Helvetica), and a serif monospace family (like Courier). Each family contains four treatments: normal, bold, italic, and bold italic. Table I shows these fonts. For special characters, a symbol font is also included. Our discovery of this was no surprise. It corresponds very closely to the 13 typefaces typically found in PostScript™ printers and in the HP LaserJet III printer.



This is a screen display that uses printer metrics.
It will look really great printed out, but on the screen,
the character spacing is very uneven. This is especially
evident when the same character is repeated as in the
rows of characters below. Notice also how some letter
combinations are uncomfortably crowded together.

mmmmmmmmmmmmmmmm

Fig. 1. One interpretation of WYSIWYG printing.

Table I
HP Help System Generic Fonts

Typeface Category	Treatments			
	Normal	Bold	Italic	Bold Italic
Serif	A	**A**	*A*	***A***
Sans Serif	A	**A**	*A*	***A***
Monospace	A	**A**	*A*	***A***

For screen use, we mapped our generic fonts to fonts included in the standard X distribution, which is available on most if not all X servers. The distribution uses New Century Schoolbook, Helvetica, and Courier fonts. Should these fonts not be available, an administrator can change the mapping table. The table is in an X resource file. For situations in which all the fonts are not available, several generic fonts can be mapped to the same physical font. We did this with Asian languages in which all four typeface treatments are not available. For Japanese, the serif family is Mincho and the sans serif family is Gothic, but all treatments are mapped to the one treatment available.

For HP LaserJet III printers, we used the same scheme to ensure we used the built-in fonts CG Times, Univers, and CG Courier. Thus, the fonts on the printer are not identical to those on the screen, but they are close enough to create a WYSIWYG state of mind to our users. Table II shows various font family mappings.

As an example of how this mapping works consider an application that requests a sans serif bold 12-point font. This request will be honored with Helvetica on a European X server, Univers on an HP LaserJet III printer, and Gothic on a Japanese X server.

Table II
Font Family Mappings to Printer or Screen

	Fonts Available		
Typeface Category	X Server (European)	HP LaserJet III	X Server (Japanese)
Serif	New Century Schoolbook	CG Times	Mincho
Sans Serif	Helvetica	Univers	Gothic
Monospace	Courier	Courier	Mincho

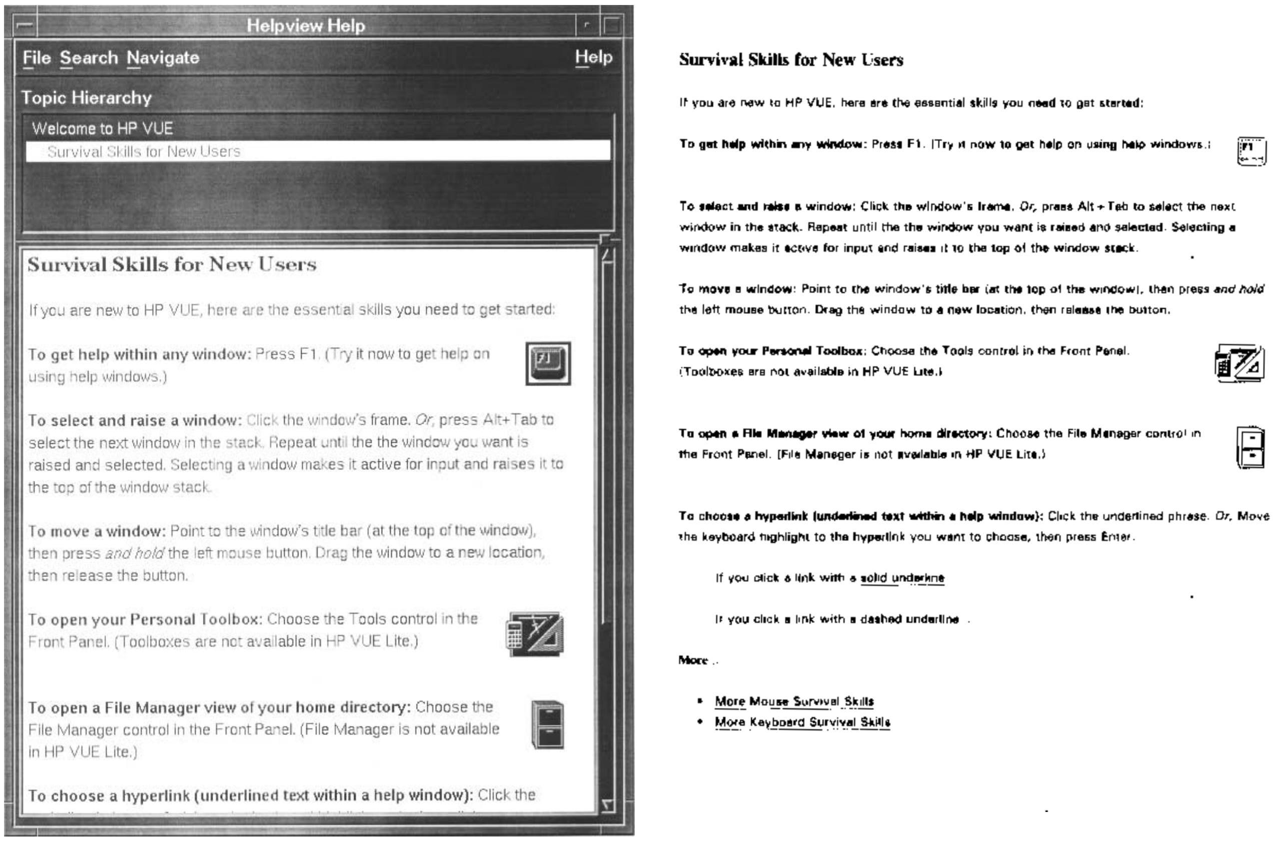
Use of generic fonts worked so well for us that we are now investigating ways to build this capability into X font servers so that applications can be assured of a reasonably rich set of fonts no matter what the platform or the current locale.

Xlib for Printing

The traditional approach to the implementation of printing in an application is to write the rendering code at a reasonably high level and then write specific drivers that convert from the higher-level code to the page-description language of the printer. With this approach an application might end up with a large number of drivers, one for each specific type of printer.

We addressed printing rather late in the development cycle. By the time we began developing printing support, the rendering code had already been written directly to Xlib, making it difficult to develop a higher-level tool kit as described above. We turned this vice into a virtue by experimenting with a new concept using Xlib functions.

Using some X toolkit and X motif routines we developed a clone of Xlib called Xvplib, which has the same functions as Xlib, but generates PCL instead of X protocol. Printing is done through a separate program called helpprint, which is called by the help widget. The helpprint program uses the same help library (Xvh) as the


Helpview Help

File Search Navigate Help

Topic Hierarchy

Welcome to HP VUE

Survival Skills for New Users

Survival Skills for New Users

If you are new to HP VUE, here are the essential skills you need to get started:

To get help within any window: Press F1. (Try it now to get help on using help windows.)

To select and raise a window: Click the window's frame. *Or*, press Alt+Tab to select the next window in the stack. Repeat until the the window you want is raised and selected. Selecting a window makes it active for input and raises it to the top of the window stack.

To move a window: Point to the window's title bar (at the top of the window), then press *and hold* the left mouse button. Drag the window to a new location, then release the button.

To open your Personal Toolbox: Choose the Tools control in the Front Panel. (Toolboxes are not available in HP VUE Lite.)

To open a File Manager view of your home directory: Choose the File Manager control in the Front Panel. (File Manager is not available in HP VUE Lite.)

To choose a hyperlink (underlined text within a help window): Click the

Survival Skills for New Users

If you are new to HP VUE, here are the essential skills you need to get started:

To get help within any window: Press F1. (Try it now to get help on using help windows.)

To select and raise a window: Click the window's frame. *Or*, press Alt+Tab to select the next window in the stack. Repeat until the the window you want is raised and selected. Selecting a window makes it active for input and raises it to the top of the window stack.

To move a window: Point to the window's title bar (at the top of the window), then press *and hold* the left mouse button. Drag the window to a new location, then release the button.

To open your Personal Toolbox: Choose the Tools control in the Front Panel. (Toolboxes are not available in HP VUE Lite.)

To open a File Manager view of your home directory: Choose the File Manager control in the Front Panel. (File Manager is not available in HP VUE Lite.)

To choose a hyperlink (underlined text within a help window): Click the underlined phrase. *Or*, Move the keyboard highlight to the hyperlink you want to choose, then press Enter.

If you click a link with a solid underline

If you click a link with a dashed underline .

More ..

- More Mouse Survival Skills
- More Keyboard Survival Skills

Fig. 2. A comparison between display (left) and printer (right) output from the help system.

display program, but links with XvpLib instead of Xlib for the printer driver code (see Fig. 3).

The helpprint application uses the Xlib call XOpenDisplay() to open a printer and then uses the XCreateSimpleWindow() routine to create a window on the print display. This window reflects the margins on the page. The help-rendering library (Xvh) then renders the current help topic into the window. The rendering process performed by Xvh involves retrieving the window size using XGetWindowAttributes(), loading fonts using XLoadQueryFont, and using XDrawImageString() to render text until done. The XvpLib library provides the Xlib functions and generates the PCL code required. Many pixel arguments such as width, height, x, and y are much larger to printer displays than to screen displays because of the 300-dpi resolution of the printer. However, calls such as XTextWidth(), which returns the space required to draw a string, still work fine. The XvpLib library supports both the HP LaserJet II (PCL 4) and the HP LaserJet III (PCL 5) printers. The primary difference between them is the selection of built-in fonts. The match from X to PCL was surprisingly close, but some X calls (such as XORing bitmaps) had to be interpreted liberally. XvpLib supports all 36 X functions used by the help topic library Xvh.

Using X as our print interface worked very well. Only about three lines of code had to be added to the help-rendering library (Xvh) to support printing. Common rendering code for the screen and printer made it easy for us to keep the layout of help topics consistent between screen and printer.

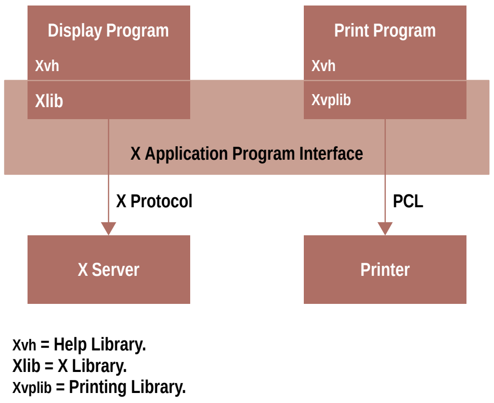


Fig. 3. The architecture for HP help printing drivers for the display and for the printer.

Printing for Special Printers

The printing approach described above works very well with PCL printers. However, HP customers in Japan presented us with special challenges. First, the Japanese language requires support for 16-bit text, and these customers typically have printers that support Asian page description languages such as Canon's LIPS and Ricoh's RPL.

To support such requirements, we took a different approach. Instead of using helpprint to render to the printer, we developed another rendering program called helpprintrst. This program creates an offscreen pixmap the size of a sheet of paper on the X server and and then calls Xvh to render into that pixmap. The print program retrieves the completed page using XGetImage() and sends it to the printing library (XvpLib) to turn the image into a PCL bitmap. Special filters in the print spooler convert the PCL bitmap into a bitmap suitable for the target printer.

Axel Deininger
Engineer/Scientist
Workstation Technology Division

require, the help system was modified to find and access compressed files. The author uses the same HP-UX compression utility mentioned earlier to compress the graphic files. The same internal decompression routine used to decompress topic files is used to decompress the graphic files into a temporary file. After decompression, the help system reads the graphic file as usual.

Using compression on X graphic files can impact access time. For very large graphic images or for a topic that uses many graphics, this impact can be noticeable. The trade-off between speed and disk space is one that the author and application engineer must address.

Color Degradation. Another obstacle encountered with handling graphics was color graphics on grayscale or black and white displays. Having to provide three different images for the three types of displays is not feasible because of disk use. Therefore, the help system has to degrade a color image for grayscale or black and white displays.

For X bitmaps, this is no problem since bitmaps are specified as using the foreground and background colors only. For TIFF images, the HP Image Library forces the image to the proper color set depending on the display type. For X pixmaps, the Xlib routines take the same approach depending on the display. This leaves the X window files to manipulate.

The task is to reduce an Xwd file from a full color image to a grayscale image containing no more than the maximum number of gray colors we have available. This number is kept in a variable called MAX_GRAY_COLORS. The HP Help System uses eight gray colors. The first step in this process is to map each of the X window color pixels to a grayscale luminosity value (① in Fig. 9). This is done using the NTSC (National Television Standards Committee) formula for converting RGB values into a corresponding grayscale value:

$$\text{luminosity} = 0.299 \times \text{red} + 0.587 \times \text{green} + 0.114 \times \text{blue}$$

where red, green, blue are the X window color values in the range of 0 to 255. This creates a grayscale luminosity value in the range from 0 to 255.

The next step is to determine the number of distinct luminosity values used in the image. This involves counting the total number of luminosity values computed above (② in Fig. 9). The idea is to group the grayscale luminosity values across the available gray colors so that the image can be evenly and reasonably rendered.

After determining the actual number of distinct luminosity values used in the image, a further calculation is done to determine which gray color to use for a given group of luminosity values (③ in Fig. 9). If the number of distinct luminosity values is greater than or equal to MAX_GRAY_COLORS, then each gray color will be used at least once. However, if the number of distinct luminosity values is less than MAX_GRAY_COLORS a calculation is performed to spread the color load among the gray colors instead of bunching them together. This provides the contrast in an image.

Finally, when the image is rendered to the display the shade of gray associated with a particular group of luminosity values is mapped to the appropriate display pixel (④ in Fig. 9).

If the system has to degrade the image to black and white, it first calculates the grayscale colors using the luminosity calculation described above. Then it dithers the image using a Floyd-Steinberg error-diffusion algorithm, which incorporates a Stucki error filter.[3]

The end user can force the whole environment including the help system to use grayscale or black and white by setting it via the HP VUE style manager's Color Use dialog. Also, if an image uses more colors than are available in the color map, the help system will automatically degrade the image until it is renderable.

Hard-Copy Support

A critical and difficult requirement of this help system was to provide WYSIWYG (what you see is what you get) printing support to match our text display capabilities. To ensure good performance and not tie up an application just to print out its online help, we implemented the printing mechanism as a separate application. When an end user requests to print one or more topics in the current help volume, the widget code packages the request and performs a vfork(2) to launch the print application.

The help system supports two different print applications: helpprint which is for HP LaserJet printers (i.e., PCL support) and helpprintrst for printing help volumes that contain multibyte characters such as those used in some Asian languages. The helpprintrst command operates just like the helpprint command except that its output does not depend on printer fonts. Instead, helpprintrst creates a page-size graphic image of each help topic.

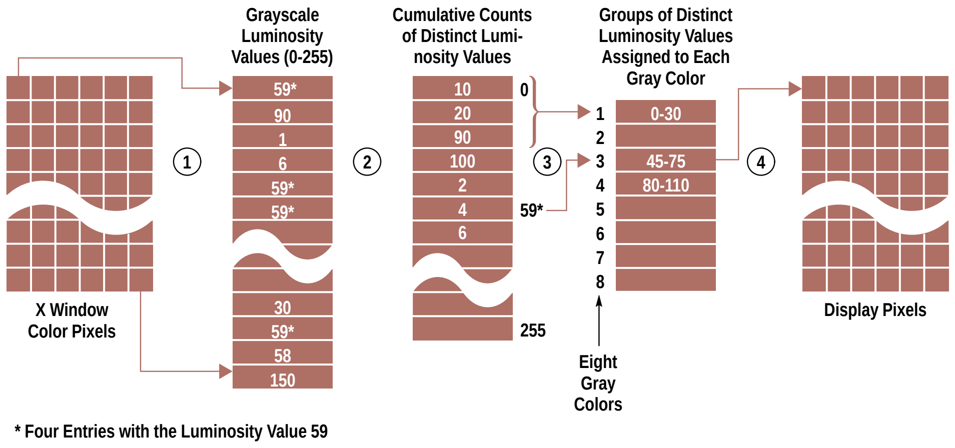
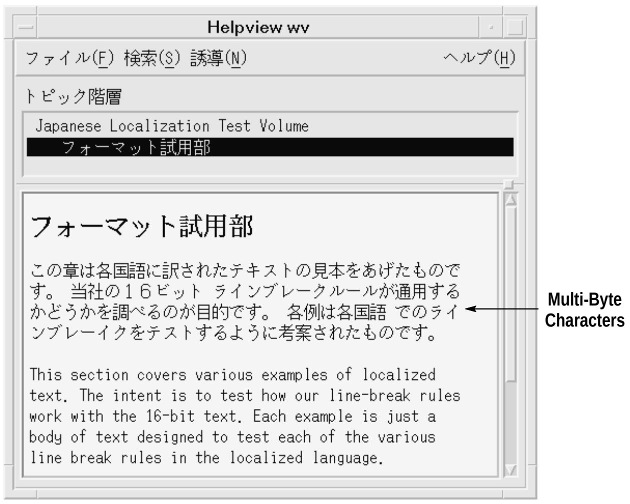

Grayscale Luminosity Values (0-255)
Cumulative Counts of Distinct Luminosity Values
Groups of Distinct Luminosity Values Assigned to Each Gray Color
59*
90
1
6
59*
59*
30
59*
58
150
10
20
90
100
2
4
6
0
59*
255
1
2
3
4
5
6
7
8
0-30
45-75
80-110
①
②
③
④
X Window Color Pixels
Display Pixels
Eight Gray Colors
* Four Entries with the Luminosity Value 59

Fig. 9. The process of reducing a full color image to a grayscale image.



フォーマット試用部

この章は各国語に訳されたテキストの見本をあげたものです。 当社の１６ビット ラインブレークルールが通用するかどうかを調べるのが目的です。 各例は各国語 でのラインブレーイクをテストするように考案されたものです。

This section covers various examples of localized text. The intent is to test how our line-break rules work with the 16-bit text. Each example is just a body of text designed to test each of the various line break rules in the localized language.

Fig. 10. A localized help window.

The goal in implementing a WYSIWYG printing solution for the help widgets was centered on creating a solution that could solve the printing problem for many applications, not just the HP Help System. Refer to “WYSIWYG Printing in an X Application,” on page NO TAG for a description of the printing solution used by the HP Help System.

Localizability

The HP Help System supports the authoring and displaying of online help in virtually any language. The online help information can be authored and translated in either single-byte or multibyte character sets. All the components within the developer’s kit are multibyte-smart and can parse and display the localized information.

The help widgets use the $LANG environment variable to determine which language directory to search to retrieve the requested help volume. For example, if $LANG = Japanese when the request to display help occurs, the widget code will attempt to open the Japanese localized version of that help volume. If one does not exist, then the default version, which is English, will be used.

When an authored help volume is compiled via HP HelpTag, the author sets the proper character set option. The character set information is used at run time to determine the proper fonts to use for displaying the localized help text. The default character set for the HP HelpTag compiler is the one defined for 8-bit character sets by ISO 8859-1. Currently only one character set per volume is supported.

Fig. 10 shows a sample of a localized help window.

Parsing Multibyte Characters. To make a single tool work for single-byte and multibyte character sets without constantly checking for the length of the current character, all characters are converted to wide characters on input. This input conversion is driven by either command line option, on an HP HelpTag entity file, or by the current setting of the locale. All internal processing of characters is done on wide characters with those wide characters being converted back to multibyte characters on output. Single-byte character sets are treated just like multibyte character sets in that the same input and output conversions always take place.

This scheme of doing all internal processing on wide characters has proven to be a very effective means for making one tool work for all languages. The scheme did require implementation of wide character versions of most of the string functions (e.g., strcpy, strlen) but those functions were all quite straightforward to create.

Localizing User Interface Components. The menus, buttons, labels, and error messages that appear in help dialogs also support full localization to native languages. The help dialogs read the user interface component strings from a message catalog named Xvh.cat. Different localized versions are supported by default and included with the developer’s kit. For languages not supplied with the help system, the developer must translate the message catalog /usr/vhelp/nls/C/Xvh.msg and then use the gencat command to create the needed run-time message catalog file.

Acknowledgments

The authors wish to acknowledge all those who contributed to the design and development of the HP Help Developer’s Kit, including the R&D, learning products, and marketing groups at the Open Systems Software Division in Corvallis, as well as the many other contributors throughout HP. Appreciation to Brian Cripe for his willingness to beat the issues into resolutions, to Axel Deininger for performing coding miracles with respect to WYSIWYG printing, and to the many managers that supported this project through to release: Bob Miller, Rick McKay, and Ken Bronstein. Special thanks goes to Dex Smith for helping our team understand the value of SGML and pushing us in that direction with respect to our product offering.

References

1. *OSF/Motif Style Guide,* Revision 1.2, Prentice-Hall, 1993.
2. *OSF/Motif Programmers Guide,* Revision 1.1 Prentice-Hall, 1991.
3. R. Ulichney, *Digital Halftoning,* MIT Press, 1988, pp. 239-244.