**SUN/188.6**

Starlink Project
Starlink User Note 188.6

R.F. Warren-Smith & P.W Draper

20th January 2010

---

# HTX
# Hypertext Cross-Reference Utilities
# Version 1.3-1
# User's Manual

---

## Abstract

This document describes a set of "Hypertext Cross-Reference Utilities" (HTX) which are designed to help maintain large documentation sets whose constituent documents are written using the *Hypertext Markup Language* (HTML).

The central part of HTX is a *hypertext linker*, `hlink`. This allows hyper-links (or cross-references) to be established between related documents in such a way that it is easy to maintain their integrity as individual documents are updated. Information produced by this linking process is also used by other HTX utilities to provide document search facilities and the ability to randomly access any part of a documentation set. This latter capability forms a basis for constructing hypertext help systems for use by other software.

The expected readership of this document includes those who read hypertext documentation, those who write it, and those who maintain it, especially those who write and maintain Starlink documentation. Software developers may also be interested in the possibilities for hypertext help that HTX provides.

## Contents

# 1 INTRODUCTION

## 1.1 What Does HTX Do?

HTX is a set of utilities that allows you to maintain and access a collection of dynamic multi-page hypertext documents that refer to each other. Its main element is a *hypertext linker* which can be used to establish cross-references between documents and to re-establish these whenever changes occur to individual documents.

The information generated by the linking process is used by other HTX utilities to provide access to hypertext documentation, permitting document searches and the rapid display of selected parts of documents.

Other software can also make use of these facilities to obtain random access to any part of a cross-linked documentation set, forming a basis for on-line or context-sensitive hypertext help systems.

See §A for descriptions of each of the utilities that HTX provides.

## 1.2 What is a Hypertext Document?

The documents that HTX is concerned with are written using the *Hypertext Markup Language* (HTML)[1].

Of course, any page of hypertext written using HTML could be considered a document in its own right. However, most documents of substance will consist of multiple pages of HTML, joined together using hypertext links. It is convenient to gather these related pages together and to consider them as a single multi-page hypertext document. This is the sense in which the term *document* is used here.

## 1.3 Why are Cross-References a Problem?

The hyper-links that join different HTML pages **within** hypertext documents are normally defined by the document's author. These are an internal matter – if the document structure changes, the author can change the internal links accordingly. If proper use is made of relative file names in establishing these links, it should also be possible to transport these documents as a whole to a different location without damaging the internal links.

HTX addresses the different problem of maintaining a collection of dynamic hypertext documents (probably written by many different authors) that have cross-references established between them. That is, each document may contain hyper-links to the contents of any other document in the collection.

In this situation, document locations may often need to change, and revision of individual documents will also result in the referenced material moving around within them. Since the links that implement HTML cross-references consist, essentially, of file names, they cannot survive this process and will rapidly end up pointing to the wrong places unless remedial action is taken.

[1]http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

In a large document collection, re-organising all necessary cross-references following every document change is not a task to contemplate doing by hand. Establishing and repairing these links is therefore a job for the HTX hypertext linker.

### 1.4 Mixing Local and Remote Documents

The inherent fragility of hypertext links is a problem faced by any project that needs to maintain a collection of related documents in hypertext form. The documentation for software projects typifies this. However, software documentation also suffers from an additional problem because it is usually distributed and installed along with the particular items of software it describes. This means that any sub-set of the complete documentation set must also be viable in its own right.

To give an example, a particular site may only have installed one or two items out of a complete collection of software, and so may only have a couple of documents that relate to the installed software available locally. This means that many of the cross-references that these documents make will be to other documents that do not exist on the local system. Ideally, these links should not simply fail.

The problem can be solved by using a document server to provide access to the missing documents from a central archive, where a copy of all the documents is maintained. A reader with any of the documents installed locally will then potentially have easy access to the entire documentation set, albeit with some time penalty when accessing non-local documents. One of the functions that HTX performs, therefore, is to generate appropriate requests for *remote documents* for processing by a central document server (see §4.4 §6).

## 2 DOCUMENT ORGANISATION

This section describes how you should organise your hypertext documentation to make use of HTX to maintain and use cross-references between documents.

### 2.1 Document Directories

HTX assumes that each hypertext document resides beneath a top-level directory and it recognises these *document directories* from their extension, which should always be ".htx". When it needs to locate the contents of a document, it will search inside document directories (to any required depth) looking for files with an extension of ".html" which contain the hypertext itself, in HTML format.

The first step in organising your hypertext documentation should therefore be to collect all the HTML files comprising each document together into a directory (or directory tree) and to give the (top level) directory a distinctive name such as:

```
design-study.htx
```

The "design-study" part of this is the document name by which it will be known to other authors, and the ".htx" part identifies it to HTX as a candidate for the various operations it performs.

Once you have made a document available to other people, you should not subsequently change its name, even if you move it somewhere else, because other documents may be referring to it and will still know it by its original name.

## 2.2 Document Libraries

Because it is concerned with collections of hypertext documents, HTX uses the concept of a *document library*. There is nothing particularly grand about this. It is simply a directory in which you store a collection of documents. You can call it whatever you like and populate it with the ".htx" directories for each of the documents in your collection. It is also a convenient place to keep other files related to the documentation set, perhaps even versions of the same documents in other formats.

The key thing about a document library is that it defines a place in which to look for documents. When you *link* a set of documents together (the term used for establishing the correct cross-references), HTX identifies the documents involved by searching in the document libraries you specify, looking for all the ".htx" directories they contain. By default, it will simply look in your current directory.

## 2.3 Multiple Libraries and the Library Search Path

You may not always want to modify the contents of existing libraries, even although you may be referring to their documents. For example, you may have an established set of documentation in one or more libraries and be developing a new document that will eventually form part of that set. Your new document will probably need to refer to the others while you are working on it, but you may not want to (or have permission to) modify any of the existing documents.

In this case, you can specify a search path, on which HTX will look for other documents to which you may be referring, but it will not attempt to link (*i.e.* modify) those other documents. The search path is specified as a colon-separated list of directories via the `HTX_PATH` environment variable, for instance:

```
setenv HTX_PATH $HOME/mydocs:/docman/newdocs:/docman/olddocs
```

This search path is used by all HTX commands when they need to find documents for which no explicit location has been given. Note that if two documents with the same name occur at different points on your `HTX_PATH`, only the **first** one will be used.

You can set `HTX_PATH` to search anywhere you like, but if you do not specify it yourself, it defaults to:

```
$INSTALL/docs:$INSTALL/help:$INSTALL/starjava/docs:\
$STARLINK/docs:$STARLINK/help:$STARLINK/starjava/docs
```

where `INSTALL` in turn defaults to `$HOME/star` and `STARLINK` defaults to `/star`. The values of these environment variables are evaluated **when the HTX software is installed** (not when you later make use of it). By default, therefore, HTX will search for the standard set of Starlink documents and on-line hypertext help, plus any others you may have installed locally under your own user name (in the location identified by the `INSTALL` environment variable).

# 3 ESTABLISHING CROSS-REFERENCES

Before you can establish cross-references between documents using HTX, you must export labelled targets for the cross-references to point at from each of the referenced documents. You must also enter links to implement the cross-references themselves. Both of these make use of standard HTML *anchors*, but in a special form that makes them recognisable to HTX.

## 3.1 Exporting Cross-Reference Targets

If one document is to cross-reference another, the referenced document should export a label that the other one can refer to. The position of this label identifies the point in the referenced document to which a reader should be transferred when the cross-reference is followed. If you understand HTML, you will be familiar with this idea, as it uses an HTML *destination anchor* for this purpose. For use with HTX, this should have a form such as:[2]

```
<A NAME="xref_conclusions">any text</A>
```

In plain HTML, this makes "any text" a potential destination for a hypertext link, to which any other piece of HTML may refer.

What makes this anchor special to HTX is the "xref_" prefix given to the value of the `NAME` parameter. HTX recognises anchors with this form as potential targets for document cross-references, and this distinguishes HTX targets from other HTML destination anchors. This is important, because it gives the document's author control over which points in the document other authors may reference.

Normally, many HTML destination anchors will be used for internal navigation within a document. When the document is revised, these may be subject to change, so they should not normally be referred to by other documents. However, destination anchors with the "xref_" prefix can be chosen carefully (and named appropriately) to identify stable components of a document, to which enduring references can be made from other documents. The author is, in effect, exporting a set of labels that tell other authors which parts of his document it is safe to refer to.

We will refer to an anchor with the form above as a *cross-reference target* and the characters that follow "xref_" as its *label* ("conclusions" in the example above). As in normal HTML, HTX labels are case sensitive. Unlike plain HTML, however, they must be unique within an entire hypertext document, not just within a single HTML file.

## 3.2 Referring to Cross-Reference Targets

To establish a cross-reference to another document you again use an HTML anchor, this time with the `HREF` parameter, in a form similar to the following:[3]

```
<A HREF="/star/docs/sun188.htx/node33.html#xref_appendix">any text</A>
```

[2] Any other HTML anchor parameters may also be present and the double quotes are optional if the enclosed characters are all alphanumeric.

[3] Again, any other HTML anchor parameters may also be present.

This is, of course, the standard HTML method of forming a hypertext link to another location identified by the *Uniform Resource Locator* (URL) given between double quotes. If this URL is correct (*i.e.* fully identifies the HTML file you want to reference), then this anchor will work as it stands, without any intervention from HTX.

What makes this link special to HTX is the particular form that the URL takes. Anchors containing such URLs are recognised by HTX and can then be repaired when the cross-reference target changes, by appropriately editing new values into the fields that are no longer valid. The key components of the URL that HTX recognises are:

```
/sun188.htx/
```

which identifies the name of the document being referenced (in this case "sun188", *i.e.* this document), and:

```
#xref_appendix
```

where "#xref_" is a flag that identifies this as an HTX cross-reference and "appendix" is the cross-reference label in the target document that identifies which part of that document is being referenced. Note that HTX will only allow you to refer to cross-reference labels that have been exported by the target document and will warn you if you use an invalid label.

Because HTX can repair URL fields that are wrong, it is quite acceptable to omit everything inessential and to strip the URL down to just:

```
<A HREF="/sun109.htx/#xref_conclusions">any text</A>
```

However, entering a full URL (or at least a guess at it) is often a good idea to start with, as it means the cross-reference may still work even if someone forgets to link your document.

## 3.3 Referring to an Entire Document

HTX recognises a cross-reference target with the form:

```
<A NAME="xref_">any text</A>
```

(*i.e.* with a blank label) as a special case, to be used to identify the "top" page of a hypertext document. This should be the point to which a reader is transferred if the entire document is requested and no particular part of it is specified.

Any other document can refer to this label, using an anchor with the form:

```
<A HREF="/any_where/docname.htx/any_file.html#xref_">any text</A>
```

but its inclusion as a cross-reference target in the referenced document is optional. If it is omitted, HTX will assume that the top HTML page in a document is named after the document itself. That is, it will use a URL such as:

```
/some_where/docname.htx/docname.html#xref_
```

If your document does not follow this convention, then you should include a cross-reference target with a blank label at the start of whichever HTML file serves the same purpose.

## 3.4 Linking your Hypertext Documents

Once you have set up a collection of documents with cross-references implemented using the special forms of HTML anchor recognised by HTX, the next step is to link these documents together. This consists of editing the URLs in HTX cross-references so that they identify the current true location of the referenced files. No other URLs are touched during this process.

Linking hypertext documents is straightforward. Simply go to the directory containing the document collection (the document library) and use the command:

```
hlink
```

**WARNING: this will modify your documents. If you are worried about this, then make a backup first.**

The `hlink` command will display some informational messages as it links your documents and will warn you if you have referred to any cross-reference labels that do not exist. When it completes, you should be able to read your documents using any WWW browser and to follow all the cross-references that they contain.

When you change any of your documents, you can re-link them to accommodate the changes simply by using the `hlink` command again. Because no essential information is discarded during linking, this process can be repeated indefinitely. You do not need a fresh copy of the documents each time.

If the documents you want to link reside in more than one document library, then simply list the library directories on the command line, as follows:

```
      hlink dir1 dir2 dir3
```

Remember that any other local documents that you refer to (but do not actually want to modify) should reside in libraries that appear on your `HTX_PATH` search path (see §2.3). If they do not, they will be regarded as remote documents and the URLs generated for them will invoke a remote document server (see §4.4 and §6). This is usually less efficient than referring directly to local files.

## 3.5 Using Cross-References from LaTeX

It is quite straightforward to insert HTX cross-reference anchors in HTML form directly into documents when writing them by hand. However, many substantial documents already exist in LaTeX format and can be converted relatively easily into hypertext by using the **LaTeX**2`HTML`[4] converter (see SUN/201). LaTeX is, in any case, a convenient format for the primary source of many documents.

A facility is therefore available for generating HTX cross-references from commands within LaTeX documents. The LaTeX commands involved are normally ignored, but are detected by the `star2html` command that provides Starlink additions to the standard **LaTeX**2`HTML`. To allow these commands to work, the appropriate document preamble must also be included. The procedure to follow is described in SUN/199, but an outline of the principles involved is also given here.

[4]http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html

To export an HTX cross-reference label from a LaTeX document that is to be converted into hypertext, the following would be used:

```
\xlabel{label}
```

which results in the HTML anchor:

```
<A NAME=xref_label>?</A>
```

where "?" is some invisible character that simply serves to mark the position to which links should point. Note that the name of the label chosen should be unique within a LaTeX document and must only contain alphanumeric characters (or be blank – see §3.3).

To insert a cross-reference to the target above into another document, you would use:

```
\xref{any text}{docname}{label}
```

which results in the HTML anchor:

```
<A HREF="http://www.starlink.ac.uk/cgi-bin/htxserver/docname.html?xref_label">any text</A>
```

This contains an initial guess at the required URL (a document stored on the RAL service), which will be recognised and modified as necessary (see §3.2) when the document is linked.

## 4 HOW HTX LINKING WORKS

This section describes the processes that occur when a set of hypertext documents is linked. It is intended to clarify the messages that appear and to give some background information to help solve problems and avoid inefficiencies.

### 4.1 Index Files

When you invoke the `hlink` command, it first looks to find all the documents that may need to be linked. This it does by searching in all the document libraries specified on the command line (or just the current directory by default), looking for directories with a ".htx" extension.

It then ensures that each document has an up to date *index file* (a file called "htx.index" stored within the document directory), creating one if necessary. Index files are text files that contain, amongst other things, details of all the potential in-going and out-going cross-references for a document and list the HTML files in which they occur. This is a summary of the cross-links the document might potentially be involved in and serves to reduce the time needed to re-link it on subsequent occasions. New index files are normally only re-generated when a document changes.

In addition to cross-reference information, index files also contain details of the title of each HTML page in the document and an indication of which is the "top" page (see §3.3). This information is used by other HTX utilities to perform documentation searches, *etc.*

### 4.2 Link-Editing

The linker identifies those documents that have changed since the library they reside in was last linked by looking for a file called "htx.log" which serves as a date-stamp in each document library (if this file is absent, all documents in the library are considered changed). The contents of the "htx.log" file also list the location of each local document when the library was last re-linked. Any that have moved to a different location are also regarded as "changed".

The linker identifies all other documents which refer to any of the changed documents. Since all of these are potentially involved in cross-references that may no longer be valid, they will all need to be re-linked.

A further search is then made of the libraries that appear on the `HTX_PATH` search path (see §2.3). If any of the documents that needs re-linking refers to a document on this path, then the latter document's index file is searched to identify the HTML file that is being referenced.

Finally, the affected HTML files are edited to insert the new, correct URLs. After this, any WWW browser should be able to follow any of the links between the documents and arrive at the correct location.

A document set will need linking again whenever a document is added or removed, or whenever one of its documents changes. However, because the linker can quickly identify which files are affected and can utilise existing index file information, subsequent linking operations are much faster.

### 4.3 Absolute and Relative Links

When establishing a link between two local documents, the HTX linker has a choice of using an absolute file name for the target, or a relative one. The rule used is that relative cross-references are always generated if two documents reside in the same document library (*i.e.* the same directory), otherwise an absolute name is used.

This means that if a document library is moved, all cross-links between documents within it will remain intact. Only those from documents outside the library will fail and require re-linking.

### 4.4 References to Remote Documents

In the interests of efficiency, the HTX linker is designed to generate references to locally-installed documentation whenever possible. If a referenced document cannot be found locally, however, it will instead generate a reference to a remote document server – which should have a complete set of documentation available. In this way, links to documents that are not available locally should not fail, and the reader should always see a complete set of documentation.

The linker regards a document as remote if it is referenced by one of the documents it is linking, but it cannot find it either in the document libraries being linked or on the `HTX_PATH` search path (see §2.3).

Note that the HTX linker will not look up a document on a remote machine. All it does is to insert the remote server reference. References to invalid labels in remote documents (or, indeed, to non-existent documents) cannot therefore be detected at link time and diagnosing this type of error is the responsibility of the server itself.

## 5 WHEN TO RE-LINK

The HTX linker is equipped to detect when documents have changed and to decide when re-linking is necessary. However, it takes a pragmatic approach to this in order to keep linking time to a minimum. As a result, there are occasions on which it may report that no documents need to be re-linked when this is not, in fact, the case.

This section is intended to help you understand when you need to re-link your hypertext documents and how to tell the linker to do this for you.

### 5.1 Accommodating Normal Document Changes

A documentation set may need to be re-linked whenever any file in any of its documents changes, or when any document is added or removed, or moved to a new location. The HTX linker detects changed documents by observing the modification dates of the document ".htx" directories, the associated index files, and both the modification date and contents of the "htx.log" date-stamp file in each document library.

This process is effective, except that it is sometimes possible to modify an HTML file within a document without changing the modification date on the ".htx" directory that contains it. This is particularly likely if the HTML file resides in a sub-directory within the document. If such modifications are carried out, a simple remedy is to use the UNIX `touch` command to update the document's ".htx" directory and record the change before running the linker.

As an alternative, the `hlink` command may be invoked with the `-d` flag to specify a *deep dependency* test of all document files. This causes it to check the modification dates of all the HTML files in all the documents in order to detect changes. This is a thorough way of detecting changes, but may be rather too slow for regular use on large document collections.

You should also remember that re-linking can be rendered necessary not only by changes in document contents, but also by changes in document location. Thus, if documents or libraries are moved, it is always wise to re-run `hlink`.

### 5.2 Forcing a Re-Link

There are several ways to force a re-link of a document or library.

To re-link a particular document, or set of documents, from scratch, their index files may simply be removed, using a command such as:

```
rm *.htx/htx.index
```

The linker will then re-generate these indices and consider the affected documents to have changed. They, and all other documents that refer to them, will then be re-linked as necessary. Alternatively, to re-link **all** documents from scratch in this way, the `-a` flag may be given on the `hlink` command line.

Removing the "htx.log" date-stamp file from a document library will also force the documents in that library to be re-linked but without unnecessarily re-generating any of its index files. This can be somewhat faster if the index files are known to be up to date.

### 5.3 Suppressing Re-Linking

It is often convenient to use the `hlink` command as part of software-maintenance procedures, such as in makefiles, since this allows your documentation set to be re-linked automatically whenever software is installed or de-installed. However, in a large software installation this can take some time. Rather than re-linking after each item of software is changed, it can sometimes be more efficient to wait until a sequence of changes has been completed. A grand re-link of the entire documentation set can then be performed.

To permit this, you can set the `HTX_NOLINK` environment variable (to any value). While this variable is set, the `hlink` command will cease to function normally. Instead. it will simply produce a message saying that hypertext linking has been suppressed. You can then perform your software updates without waiting for re-linking to occur. When you have finished, you can un-set the `HTX_NOLINK` environment variable and use `hlink` to bring the cross-references in the associated documentation up to date.

## 6 REMOTE DOCUMENT SERVERS

This section explains how a remote document server is specified and the role it performs.

### 6.1 Specifying a Document Server

If HTX detects that a document has been referenced but cannot find it locally (see §4.4), it will generate a reference to a remote document server in place of a link to the local document. The URL it generates in this case takes the form:

```
$HTX_SERVER/docname.htx/some_file.html?xref_label
```

That is, it replaces any directory path at the start of the URL with the URL of a remote document server, obtained by translating the `HTX_SERVER` environment variable. It also replaces "#xref_" with "?xref_" so that the server will receive the label field of the URL as a *query string* which it can interpret.

You may set the `HTX_SERVER` environment variable to identify any document server you like (for instance, one that you write yourself). However, if you do not specify one explicitly, the default is to use the Starlink HTX document server at RAL, whose URL is:

```
http://www.starlink.ac.uk/cgi-bin/htxserver
```

Once a document has been linked, the choice of server is fixed, until it is linked again.

### 6.2 The Document Server's Role

It is the remote document server's responsibility to interpret the URL it receives when the cross-reference is followed and, typically, to return the correct URL (or the document itself) to the browser. The server will therefore have to search for the document in whatever form it takes on the remote machine (it may not necessarily be in HTML format). This may involve similar steps to those that the HTX linker would have performed had the document been available locally (*e.g.* looking the reference up in an index file), except that they are performed at run-time rather than in advance when the document is linked. The server should also issue any necessary diagnostics – about invalid labels or documents that cannot be found, for example.

## 7 GAINING ACCESS TO DOCUMENTS

In previous sections we have concentrated on how to write hypertext documents and organise them into an inter-linked documentation set. In this section, we turn our attention to accessing those parts of the documentation set that we want to read.

### 7.1 Displaying Documents by Name

The simplest form of document access is provided by the `showme` command, which simply displays the document you name. For instance:

```
showme sun188
```

would display the "top" page of the document called "sun188" (*i.e.* the one you are reading now). Using this command is normally easier than entering a full URL for the document.

The `showme` command displays the document using a WWW browser and will make use of one you already have running if possible. You can specify which browser to use, if required (see §B.1).

If you do not give any directory information, `showme` will search for the document using the `HTX_PATH` search path (see §2.3), but you can specify explicitly where the document is if you prefer, as in:

```
showme ~/mydocs/galaxy_survey
```

If `showme` cannot find the document locally, it will try and fetch it from the remote document server instead(see §6). You can suppress this behavior using the `-l` switch if you prefer:

```
showme -l document
```

in which case failure to find the document locally will simply result in an error. You can also specify that a remote copy of the document is required using the `-r` switch, in which case any local copy will be ignored and the remote document server will be asked to supply the document.

### 7.2 Displaying Parts of Documents by Label

The `showme` command also allows you to read selected parts of documents, specified using the cross-reference labels they contain(see §3.1). To do this, you simply add the label as an extra argument. Thus:

```
showme sun188 hlink
```

will display the description of the `hlink` command in this document.

Using this feature requires that you know the names of the cross-reference labels present in the document, or at least requires that they be easily guessable. If this is not the case, you may prefer to select which parts of documents to view by means of a document search instead(see §8).

## 8 SEARCHING FOR INFORMATION IN DOCUMENTS

As well as allowing you to access documents directly by name (which is normally the fastest method if you know where to find the information you want), HTX also allows you to search for information by keyword.

### 8.1 Performing Keyword Searches

Keyword searching is performed using the `findme` command, which is simply illustrated:

```
findme HTX
```

This command will search your documentation set for the string "HTX" and will then display a list of the documents found using your WWW browser, with each entry in the list being a hyper-link to the document in question. It is then a simple matter to follow the link to the document you want to read (in this example there will probably only be one document to choose from – this one).

The way in which the `findme` command performs its search is explained in the next section, but in essence it attempts to find information about major topics quickly by searching only the main titles of documents. It then goes on to consider more detailed (and time consuming) searches only for more obscure topics that can't be found readily. The progress of the search is displayed on your terminal, so you can interrupt it if you fail to find what you want quickly and don't want to wait.

The more level of detail `findme` needs to consider, the more detailed will be the list of results it generates, with individual HTML pages being listed if appropriate. This strategy of performing progressively deeper searches can be observed if you ask for information on something a little more obscure, like:

```
findme findme
```

which makes `findme` search for information on itself, or even something very obscure, like:

```
findme HTX_PATH
```

which will (probably) only be found in the body of the text of this document.

## 8.2 Controlling the Depth of Search

The `findme` command allows you to find information at the level of detail you want by searching any of four categories of information associated with hypertext documents:

(1) **The document name (-n switch)**
This is given by the name of the directory that contains the document(see §2.1). For example, the name of the document you are reading now is "sun188" because its hypertext version resides in a directory called "sun188.htx". The name of a document may indicate what category of information it contains and is very quick and easy to search.

(2) **The document title (-t switch)**
This is extracted from the "top" HTML page of a document (see §3.3). and consists of the text that appears between the <`TITLE`> and <`/TITLE`> tags in the HTML header section for that page. The title of a document is obviously a good place to search for important topics and this can be done quite quickly.

(3) **The document's page headings (-h switch)**
These are extracted in the same way as the document title (above), but from all the other HTML pages in the document, excluding the "top" page. If you have converted your document from a format such as LaTeX (see §3.5 and SUN/199), then these will be the section headings that appear in the printed form of the document. This is normally a fruitful place to search for slightly more specialised topics and can be done without a serious time penalty because HTX caches this information in a document's index file(see §4.1).

(4) **The lines of text in the document (-l switch)**
These consist of the contents of all the HTML files in the document (including all their HTML tags, URLs, *etc*). This is the ultimate place to search for information, but this can take quite a while if the documentation set is large.

If none of the switches shown above is used when `findme` is invoked, its default action is to:

(1) Search the document titles

(2) If that fails to find a match, search the page headings

(3) If that fails to find a match, search the lines of text

However, if one or more of the `-n`, `-t`, `-h` or `-l` switches is used, then *only* the specified categories of information will be searched, and this will be done in a single pass through all the documents. For instance:

```
    findme -n sun
```

will cause only the document names to be searched (for the string "sun"), while:

```
    findme -t sun
```

would search only the document titles, and:

```
findme -t -h sun
```

would search both the titles and page headings in a single pass.

## 8.3 Searching Specific Documents

By default, the `findme` command will search your entire documentation set, consisting of all the documents found on the `HTX_PATH` search path (see §2.3) [5].

However, you can restrict the search to specific documents by listing them after the keyword you are searching for, thus:

```
findme targets sun188
```

would find information about "targets" in this document. This is often a useful way of finding reference information once you are reasonably familiar with a document's contents. Restricting the search to a specific document will also make it far more rapid.

You may specify as many documents to search as you want. If you do not give explicit directory information, the `HTX_PATH` search path will be used to locate them.

## 8.4 Other Search Options Available

The `findme` command has a number of options that allow you to fine-tune the search that it performs. These include:

- Performing case-sensitive searches (by default, keyword matching is case insensitive)
- Searching for whole words (by default the keyword you give will match any string, even if it is only part of a word)
- Matching patterns in text by including regular expressions in the keyword string
- Abbreviating the output list by suppressing information about individual HTML pages and only displaying document names and titles
- Sorting the output list into order according to the significance or number of matches found in each document.
- Including information about the number of matches found.

See the `findme` command description in §A for full details.

[5] It will also include the names and titles of documents found in catalogue files (see§10)

## 9 USING HTX FACILITIES FROM OTHER SOFTWARE

### 9.1 Providing Hypertext Help

An important application of the `showme` command is to provide hypertext help for other software packages.

In this situation, the help information would be contained in one or more hypertext documents and the controlling software would invoke the `showme` command to display the required part of it on demand, the information being selected by giving an appropriate cross-reference label (see §3.1). The person reading the displayed information can then explore any hyper-links within it to gain further information. These could point at other documents you have linked the help documents against or, indeed, anything else on the WWW.

When designing a graphical user interface, help information can easily be made "context sensitive" by setting the cross-reference label according to the task being performed (*e.g.* which window is active) and invoking `showme` when a "help" button is pressed. In fact, with a dedicated WWW browser, one might even consider displaying the information automatically as the task in hand changed, without waiting to be asked.

By going via this HTX interface, the controlling software is insulated from changes in the way the help documentation is organised and indexed. It also need not concern itself with how to communicate with the WWW browser. In some cases, however, you may want your software to handle the display of help information yourself. You can do this by using the `-n` switch on the `showme` command, thus:

```
showme -n help_document subject_label
```

This prevents `showme` from displaying the document. Instead, it simply writes the URL for the part of the document you requested to its standard output. Your software can then read this and handle it in whatever way you choose.

### 9.2 Using HTX to Control a WWW Browser

If you have used the `showme` command to obtain a URL for a document (see above), one possible way of using this might be to save it and pass it back to `showme` later on, using the `-u` option for displaying a document by URL:

```
showme -u url
```

Of course, the URL you give need not have come from `showme` in the first place – any URL that your WWW browser can handle would be acceptable.

You can also supply the name of a local file using the `-f` switch, thus:

```
showme -f filename
```

Because you can give relative file names to `showme`, this is often an easier way of viewing a file interactively than typing its full path into the browser.

These examples illustrate how `showme` provides a convenient and uniform interface for "remotely controlling" a WWW browser, permitting you to display any WWW document for which you have a URL or file name.

### 9.3 Performing Document Searches from Other Software

You can use the `findme` command from within other software to implement document searches with a user interface of your own design. This is done by using the `-html` switch, as follows:

```
findme -html -q keyword
```

This switch prevents `findme` from displaying its list of results via a WWW browser, and it instead writes them to its standard output in HTML format.[6] In this mode, a "naked" output list is produced that lacks the surrounding HTML document and this allows it to be embedded in any other HTML context where it may be needed.

An example of this might be a WWW forms interface for performing local document searches. Typically, a *Common Gateway Interface* (CGI) script would be written to be invoked by your WWW server and perform the search, generating a page of HTML as its output. Such a script could invoke the `findme` command in the form above to implement the search and then embed its output in the page it generates.

Alternatively, a script invoked from the command line or a graphical user interface might build an HTML document, including search output from `findme`, in a local file. It could then use the `showme` command with its `-f` flag to display the result.

You can test for the success of a search by examining the return status from `findme` which is set to the number of documents that were matched.

# 10 CATALOGUE FILES

Catalogue files provide a way of introducing documents into the searches performed by the `findme` command which it would not otherwise be able to search. This is of particular benefit if some documents are not stored locally, or are not in hypertext format.

## 10.1 Catalogue File Name and Format

An HTX catalogue file is a text file with the name "htx.catalogue" which resides in a document library. Each line of the file should contain an entry consisting of three fields separated by white space, in the form:

```
docname docfile title_text
```

where:

[6]In this case the `-q` switch has also been used to suppress messages about the progress of the search.

- **docname** is the name by which the document is to be known
- **docfile** is the name of the file containing the document, given relative to the document library directory
- **title_text** is the document's title, which may contain further white space

## 10.2 A Catalogue File Example

Suppose, for example, that you are converting an existing documentation set into hypertext form, but still have some documents available only in *DVI* and *postscript* format (with file extensions ".dvi" and ".ps"). The `findme` command will not be able to search these "old" documents because it doesn't know how to extract (for example) their titles from the files provided. To help overcome this, you would describe these documents in a catalogue file, perhaps along the following lines:

```
review doc1.ps A Review of Documentation Systems
intro doc2.dvi Introduction to Hypertext
writing writing.ps How to Communicate Effectively
...
```

Note that the document name and file name need not match. This file introduces the listed documents to the `findme` command, tells it where to find the corresponding document files and allows it to perform searching by document name and/or title (but not by page heading or lines of textual content, since it cannot know how to decode the document format to obtain these).

## 10.3 How Catalogue Files are Searched

Documents listed in HTX catalogue files are added into the documentation set **after** all hypertext documents have first been found using the `HTX_PATH` search path (see §2.3). If a document is found in hypertext form, it occludes any subsequence occurrence of a document with the same name in a catalogue file. This means that if you convert an "old" document into hypertext form (with a ".htx" file extension), the new version will automatically be found in preference to the old one – there is no need to remove it from the catalogue file.

Catalogue files are also found by following the `HTX_PATH` search path after it has been used to find hypertext documents, and are recognised by the name "htx.catalogue". If more than one catalogue file is found, their contents are simply concatenated in the order in which they are found.

Duplicate entries for a document are permitted in catalogue files (and can also arise when catalogue files are concatenated). They provide a mechanism for a document to have alternative titles. This can sometimes improve the usefulness of document searches if the original title lacks any useful keywords (you might think of this as combining both a title index and a subject index into the same file). If more than one title entry is matched for a particular document, then the one that occurs first in the catalogue file(s) is used.

### 10.4 Providing a Local Catalogue of Remote Documents

The documents listed in HTX catalogue files need not necessarily exist on the local file system. HTX will check to see if they do, and will generate hyper-links to them if they appear to be readable. For files that are not accessible, however, it will generate a reference to the remote document server (see §6). This reference will take the standard form (§6.1). using the document name – not the file name used in the the catalogue file.

Catalogue files can therefore be used as a searchable catalogue of documents that are available remotely. In fact, a document library containing only a catalogue file could be searched by the `findme` command and any matches would then refer to the remote version of the document, in whatever form it happens to be stored.

## A DESCRIPTIONS OF HTX COMMANDS

## findme
## Search for documents by keyword and display a list of those found

**Description:**

This command performs keyword searching of locally available (or locally catalogued) documents and displays a list of those found using a WWW browser. This list includes hyper-links to the parts of each document that were matched.

**Invocation:**

```
findme [switches] [keyword] [doclist]
```

**Parameters :**

**keyword**

The string of characters to be searched for. This is treated as a single string (not as multiple keywords) and should be quoted if it contains special characters or white space. Pattern matching characters, as used in "sed" or "grep" regular expressions, may be included.

If this parameter is omitted, then all documents searched will be matched. This provides a convenient way of listing all the documents available.

**doclist**

An optional space-separated list of the documents to be searched. If this is omitted, then the complete set of hypertext documents will be searched, as found on the `HTX_PATH` search path (see §2.3). Any further documents described in catalogue files (§10) will also be included.

Any ".htx" extension on document names is ignored.

**Switches :**

**-b** Requests that a "brief" list be produced of the documents found. This means that only document names and titles will appear and references to individual pages will be omitted. By default, individual pages will be listed if the search has included them and they match the search criteria.

**-c** Indicates that case is significant when searching for the keyword. By default, differences in case are ignored.

**-f** Indicates that a full search should be performed, involving searching document names, titles, page headings and lines of text. This option is shorthand for the switches `-n`, `-t`, `-h` and `-l` used together.

**-h** Indicates that a search for the keyword is to be performed on all the page headings(see§8.2) within each document.This provides a convenient compromise between speed of execution and full search coverage, and generally produces an acceptable amount of output. By default, a search on page headings is performed if the keyword cannot be found in any document title. Page headings may only be searched in local hypertext (".htx") documents.

**-html**
Indicates that the list of documents found by this command should not be displayed using a WWW browser. Instead, the results are simply written to standard output as a list in HTML format (without a surrounding HTML document). This provides an interface for other software that will display the results itself.

Note that the URLs used for hyper-links in this output list will be suitable only for local use by a WWW browser running on the same machine as the `findme` command. They may not be suitable, for instance, for embedding in an HTML document that will be interpreted by a remote WWW browser.

**-l** Indicates that a search for the keyword is to be performed on all lines of text (see §8.2) within each document (note that this will include all HTML tags, URLs, *etc.*). This provides the fullest possible form of keyword search, but may take some time to complete and could generate a large volume of output. By default, a search of document lines is only performed if the keyword cannot be found in any document title or page heading. Line-oriented searching can only be performed on local hypertext (".htx") documents.

**-m** Indicates that the output list is to contain information on which search criteria were matched and how many matches were found. By default, this information is omitted.

**-n** Indicates that a search for the keyword is to be performed on the name (see §8.2) of each document. Searching of document names is not performed by default.

**-q** Indicates that the search should progress in "quiet" mode without producing messages about its progress. By default, messages about the progress of the search are written to the controlling terminal.

**-s** Indicates that the output list is to be sorted so that the most significant matches appear first and the least significant last. In assessing this, matches to the document name are given the highest significance, then matches to titles, page headings and finally lines of textual content (see §8.2). An alphabetical sort on document title and page heading is used to resolve any remaining ambiguity over output order. By default, a simple alphabetical sort on document title and page heading is used alone.

**-t** Indicates that a search for the keyword is to be performed on the title (see §8.2) of each document. This provides a quick but effective form of search for major topics and gives at most one entry in the output list for each document matched. By default, title matching is performed first, and a search of page headings (and eventually lines of textual content) is performed only if this initial search fails to match any documents (see §8.2). Using `-t` prevents these subsequent searches from happening automatically.

**-w** Indicates that the keyword supplied must match an entire word (*i.e.* a string delimited by characters which are not underscores or alphanumerics, or delimited by the beginning or end of the text, or by a newline). By default, the specified string of characters is matched wherever it occurs, so long as it does not span multiple lines of text.

**-warn**
Indicates that any warning messages issued by the WWW browser (*e.g.* when it is started up) are to be suppressed. By default, these warnings are written to standard error along with any other warning or error messages.

**Exit Status :**

The exit status from this command is set equal to the number of documents matched by the search. Thus a non-zero exit status indicates success, while a zero status indicates failure to find any document. Note that this is the reverse of the convention normally adopted by UNIX commands.

**Notes On Searching :**

- If none of the switches `-n`, `-t`, `-h` or `-l` is used, the keyword given will first be searched for in the title of each document. If this fails to produce a match, it will next be searched for in the page headings of each document. If this also fails to produce a match, a final search of the lines of text within each document will be made.
- If one or more of the switches `-n`, `-t`, `-h` or `-l` is used, the automatic sequence of searches described above will not occur. Instead, only those document components specified by these switches (name, title, page header and lines of text, respectively) will be searched. This will be done in a single pass through all documents.
- To obtain the fullest possible (but slowest) search, use the `-f` option. This is equivalent to using all of the switches `-n`, `-t`, `-h` and `-l` together.

**Notes on Specifying Documents :**

- If no documents are specified, then all directories on the `HTX_PATH` search path will be inspected for hypertext (".htx") documents and all those found will be searched. In addition, if any directory on the search path contains a catalogue file, the documents it describes will also be included in any search of document names or titles (but not of page headings or lines of text).
- If one or more document names are supplied, then the search will be restricted to the specified documents only. If these document names are supplied without directory information, then they will be located by following the `HTX_PATH` search path and then by reading the contents of any catalogue files, if necessary. If document names are supplied with explicit directory information, then they must refer to local hypertext documents and no search will be made to locate them.
- If documents with the same name are found both locally in hypertext form and in one or more catalogue files, then the hypertext version takes precedence. If a document with the same name appears more than once in the list given for the `doclist` parameter, then the first occurrence takes precedence, except that the first occurrence of a name with explicit directory information always takes precedence over the same document specified without directory information.

**Examples:**

```
findme
```

Finds all available documents.

```
findme guide
```

Finds all documents with the string “guide” in them. This is done by first searching their titles, then (if that fails) by searching all their page headings, then (if that also fails) by searching all of their lines of text.

```
findme -t guide
```

Finds all documents with the string “guide” in their title. Only titles are searched.

```
findme -n sun
```

Finds all documents whose names contain the string “sun”.

```
findme -w star
```

Finds documents that contain “star” as a word on its own.

```
findme -c GnS
```

Finds documents that contain the string “GnS” with the correct capitalisation.

```
findme -l -b unix
```

Searches all lines of text in all documents for the string “unix” and displays a brief listing of the results, so that only the relevant document names and titles are shown.

```
findme -h DAT_ sun92
```

Finds a document called “sun92” and searches its page headings (only) for the string “DAT_”. Each page which matches is listed.

```
findme -t -h '?$' docs/*.htx
```

Searches the titles and page headings of all hypertext documents stored in directory `docs` and lists those which end in a question mark (“?$” is a regular expression specifying that there should be a question mark at the end of a line).

```
findme -h -html -q "$keyword" pkg_manual »/tmp/results$$
```

This command might be used to provide a command lookup facility for a software package. It searches all the page headings in the document called “pkg_manual” for the command stored in the `keyword` variable and appends the resulting list (in HTML format) to a scratch file in which an HTML page of results is being constructed. Messages about the progress of the search are suppressed with the `-q` switch.

**Environment Variables Used :**

`HTX_BROWSER`
: The command which will be used to invoke the WWW browser (see §B.1).

`HTX_PATH`
: A colon-separated list of the library directories in which to search for hypertext documents see §2.3).

`HTX_SERVER`
: The URL of the document server to be used for serving remote documents (see §6.1).

`HTX_TMP`
: The name of the directory in which to create temporary communication files. If this variable is not set or is null, then `$HOME/.htxtmp` is used instead (see §B.3).

## hlink
## Perform cross-linking of hypertext documents

**Description:**

This command searches for hypertext documents in a specified list of library directories and link-edits their ".html" files to insert the correct URLs so that cross-reference links between the documents point at the appropriate files.

**Invocation:**

```
hlink [switches] [dirlist]
```

**Parameters :**

**dirlist**

A space-separated list of directories in which to search for the hypertext documents to be linked (as identified by their ".htx" file extension). If this list is not given, the default directory "." is used.

**Switches :**

**-a** Specifies that all documents in the specified directories should be re-linked "from source". This means that all documents will be re-examined for cross-references, regardless of whether they appear to have changed since they were last linked, and all the resulting cross-references will be resolved.

**-d** Specifies that a "deep" dependency test should be performed to see which documents have changed, and therefore need re-linking. This involves examining the modification dates of all the ".html" files in each document. A deep dependency test is more likely to identify changed documents than the default method (which simply examines the modification date of the document directory file) but it may take considerably longer.

**-r** Specifies that documents that make references only to other "remote" documents should be re-linked. This option is present for historical reasons and it should not normally be necessary to use it. It may eventually be removed.

**-v** Specifies "verbose mode", which results in additional information about the linking process being written to standard output.

**Notes:**

Documents which are found on the `HTX_PATH` search path will be linked against (*i.e.* references to them will be resolved), but they will not themselves be modified unless they are also found in the directory list specified on the command line.

**Examples:**

```
hlink
```

Re-links all documents in the current directory against themselves and all other documents found on the `HTX_PATH` search path. New index files are first generated for any documents that have changed.

```
hlink .  mydocs
```

Re-links all documents found in both the current directory and the "mydocs" directory against themselves and all other documents found on the `HTX_PATH` search path. Only those documents residing in the "." and "mydocs" directories will be modified by this command.

```
hlink -d
```

Re-links all documents in the current directory after first checking the modification dates of **all** the ".html" files they contain to determine which documents have changed since the last re-link.

```
hlink -a dir1 dir2 dir3
```

Generates new index files and re-links all documents in the specified directories regardless of whether they have changed since the last re-link.

**Environment Variables Used :**

`HTX_NOLINK`
Setting this environment variable (to any value) will suppress re-linking. A message to this effect will be produced instead(see §5.3).

`HTX_PATH`
A colon-separated list of the library directories in which to search for hypertext documents (see §2.3).

`HTX_SERVER`
The URL of the document server to be used for serving remote documents (see §6.1).

## showme
## Display a specified part of a document using a WWW browser

**Description:**

This command takes the name of a document and (optionally) the name of a cross-reference label within it, and displays the requested part of the document using a WWW browser. It may also be used to display local files or any other WWW page for which a URL is available.

**Invocation:**

```
showme [switches] doc [label]
```

**Parameters :**

**doc**

The name of the document. If no directory information is supplied, the document will be located using the `HTX_PATH` search path (see §2.3). If directory information is given, no search will be performed. In either case, a ".htx" extension to the document name is optional.

If the `-f` option is used, this parameter is interpreted as the name of a local file which is to be displayed. If the `-u` option is used, it is interpreted as a general URL.

**label**

An optional cross-reference label. If given, this specifies which part of the document is required. If omitted, it is assumed that the entire document is required and the appropriate "top" page is displayed(see §3.3).

This parameter is ignored if the `-f` or `-u` options are used.

**Switches :**

**-f** Specifies that the document name supplied is the name of a local file (*e.g.* in HTML format) which is to be displayed. If this option is used, the `label` parameter and the `-l` and `-r` switches are ignored.

**-l** Specifies that a local document should be displayed and that it should not be fetched from a remote document server. In this case, if the document cannot be found locally an error will result.

**-n** Specifies that the document is not to be displayed. In this case, no WWW browser will be used and the URL that would otherwise have been passed to it is simply written to standard output. This provides an interface for use by other software that will handle document display itself.

Note that the URL generated is suitable only for local use by a WWW browser running on the same machine as the `showme` command. It may not be suitable, for instance, for embedding in an HTML document that will be interpreted by a remote WWW browser.

**-r** Specifies that a remote document is required. In this case, a reference to a remote document server will be generated and passed to the WWW browser, even if a local copy of the document exists.

**-u** Specifies that the document name supplied is a URL which is to be passed directly to the WWW browser for interpretation. Any form of URL which the browser can handle may be given. If this option is used, the `label` parameter and the `-l` and `-r` switches are ignored.

**-warn**
Indicates that any warning messages issued by the WWW browser (*e.g.* when it is started up) are to be suppressed. By default, these warnings are written to standard error along with any other warning or error messages.

**Notes:**

- If the document is found locally, the local copy will be displayed. If it is not found locally (subject to any command switches) a reference to a remote document server will be passed to the WWW browser instead.
- If a cross-reference label is specified for a local document but it cannot be found in that document, then an error will result. There is no checking of the existence of remote documents, nor of the cross-reference labels they contain. This checking must be performed by the remote document server.
- The `-l` and `-r` switches are mutually exclusive. If both are given, the latter one predominates. Similarly, the `-f` and `-u` switches are mutually exclusive. If both are given, the latter one predominates.

**Examples:**

```
showme sun188
```

Displays the document called "sun188". The local copy is used, if available, otherwise it is fetched from the remote document server.

```
showme sun188 showme
```

Displays the part of document "sun188" identified by the cross-reference label "showme" (*i.e.* the section you are reading now). As before, the local copy of the document is used, if possible.

```
showme -l quantum-theory
```

Searches for a local copy of the document "quantum-theory" and displays it. If the document cannot be found locally, an error results.

```
showme -r quantum-theory speculation
```

Displays the section identified by the "speculation" cross-reference label in a remote copy of the document called "quantum-theory". Any local copy is ignored.

```
showme mydocs/help available_commands
```

Displays the section identified by the “available_commands” label in a local document stored in the directory `mydocs/help.htx`. Because explicit directory information is given, the document is not searched for using the `HTX_PATH` search path. If the document doesn’t exist, a copy will be fetched from the remote document server.

```
showme -n -l mydocs/help available_commands
```

Performs the same function as the previous example, except that the document is not actually displayed. Instead, its URL is simply written to standard output. The `-l` switch specifies that a local document is required, so an error will result if it cannot be found.

**Environment Variables Used :**

`HTX_BROWSER`
: The command which will be used to invoke the WWW browser (see §B.1).

`HTX_PATH`
: A colon-separated list of the library directories in which to search for hypertext documents (see §2.3).

`HTX_SERVER`
: The URL of the document server to be used for serving remote documents (see §6.1).

`HTX_TMP`
: The name of the directory in which to create temporary communication files. If this variable is not set or is null, then `$HOME/.htxtmp` is used instead (see §B.3).

# B USING WWW BROWSERS WITH HTX

## B.1 Specifying a WWW Browser

HTX commands that need to display documents do so using a WWW browser and will try to make use of one that you already have running, if possible. Otherwise a new one will be started. By default, one of the browsers in the list *firefox*, *mozilla*, *netscape*, and *mosaic* will be used (in that priority order). On Mac OS X the default browser will be used.

You can change the browser that is used by means of the HTX_BROWSER environment variable. Thus, if you normally use the command "mozilla" to invoke your browser (and also have "firefox" installed), you could put the following into your `.login` file:

```
setenv HTX_BROWSER mozilla
```

Note that this command should not be an alias (*e.g.* if you normally use the C shell).

Currently, HTX can work only with the browsers listed above.

## B.2 WWW Browser Communication

HTX communicates with your WWW browser using a *remote control* mechanism which is different for each browser.

In general, if you use only one machine at a time and never start up more than one copy of your browser, you are unlikely to have problems. However, if you use multiple machines or multiple browsers, the following describes some of the browser-specific behaviour you may need to be aware of.

### B.2.1 Netscape, Mozilla and Firefox Behaviour

HTX communicates with these browsers using the remote control commands, which work by means of *X resources*, in essence looking for an existing window on your X display. If there is more than one of these, the first one will generally be used. If it cannot find such a window, HTX will start a new instance.

One problem with this arrangement is that HTX may detect an instance running on a different machine but displaying on the same X display. In this case, attempts to communicate with it will succeed, but the browser may not necessarily be able to access the local files that HTX asks it to view (but see §B.3). The best way around this problem is to ensure that the browser to be used by HTX is started first.

## B.3 Inter-Machine Browser Communication

Often, your file system will be shared by more than one machine so that communication between different machines is possible via files. Such an arrangement provides an opportunity for HTX to pass information to WWW browsers which are already running on other machines. With the *Netscape* family of browsers, this can make working on multiple machines considerably easier by reducing the number of browser invocations you require. For it to work successfully, however, the files you are accessing must be known by identical names on all the machines involved.

If your file system is suitably set up, then HTX will normally be able to take advantage of this without any further action on your part. To achieve this, the temporary files which HTX creates to communicate with the browser will be placed in the directory `$HOME/.htxtmp`[7]. Therefore, if you use the same login directory (and `$HOME` translates to the same name) on each machine, these temporary files will be accessible to the browser wherever it is running.

If your login directory is not the same on each machine, then you can set the environment variable `HTX_TMP` to give the name of a suitable alternative directory to hold these temporary communication files. The directory you use should be accessible (for both reading and writing) and be known by identical names on all the machines which will use it.

[7]This directory is created automatically when needed.