

Hype — A Hypertext Browser

*Gerard J. Holzmann
Bell Laboratories
Murray Hill, New Jersey 07974*

ABSTRACT

Hype is an X-based previewer for troff documents. It offers support for hyper-text commands that can either be embedded into the troff sources or issued by independent UNIX® commands, while the previewer is running.

The embedded commands leave tags in the troff output that are interpreted by `hype`, but are invisible to other previewers and to devices such as line-printers and typesetters.

Issuing the commands at run-time make it possible to add hyper-text support to any troff document, without modification of the sources. The information required to built hyper-text support of this type can be extracted from the raw troff-ouput files with a few simple post-processing tools that will be discussed.

September 9, 1993

Hype — A Hypertext Browser

*Gerard J. Holzmann
Bell Laboratories
Murray Hill, New Jersey 07974*

1. INTRODUCTION

Hype is a general previewing tool for troff output on X-terminals. When given a single filename as an argument, it reads the file and displays the first page on the screen. Using a menu, or the buttons, the user can browse through the pages of the document, in arbitrary order. If no filename argument is given, hype reads the standard input up to an end-of-file, places the resulting data into a temporary file, and proceeds as before.

These two modes of use are illustrated by the following examples. The dollar-sign is the shell-prompt.

```
$ pic memo | eqn | troff -ms > out
$ hype out &
```

or

```
$ pic memo | eqn | troff -ms | hype
```

Diagnostics are always sent to the standard output, i.e., to the X-window that issued the hype command.

Section 2 explains the general working of the tool. Section 3 introduces the hypertext mechanisms that are supported. Section 4 reviews a small number of pre- and post-processing tools that can be used in combination with hype to create bitmaps, indexes, or hyper-links. Section 5 mentions the temporary files created by hype, and Section 6 summarizes the main features and the control language.

CONTENTS					
Section	Title	Page	Section	Title	Page
1	Introduction	1	3.4	Userdefined Menus	5
2	Basic Operation	2	3.5	File Access	6
2.1	Predefined Buttons	2	3.6	The System Command	6
2.2	Predefined Menus	2	4	Pre- and Post-processing	8
2.3	Defining a Page Grid	2	4.1	Creating X11 Bitmap Files	8
2.4	Bitmaps	3	4.2	Page Balancing	8
3	Hyper-text Support	3	4.3	Finding Keywords – Preparing an Index	8
3.1	Control Channel	4	5	Temporary Files	10
3.2	Control Language	4	6	Summary	11
3.3	Userdefined Buttons	4			

2. BASIC OPERATION

2.1. PREDEFINED BUTTONS

Hype places six buttons at the top of the screen, as illustrated in Figure 1. The buttons are labeled `redraw`, `first`, `next`, `previous`, `quit`, and `open`.

Pushing the `open` button causes `hype` to restart the session for the filename specified in the immediately following text-entry box. The text-entry box can be scrolled left or right on long file-names, using mouse button 1. Typing a newline in the text-entry box has the same effect as pushing `open`.

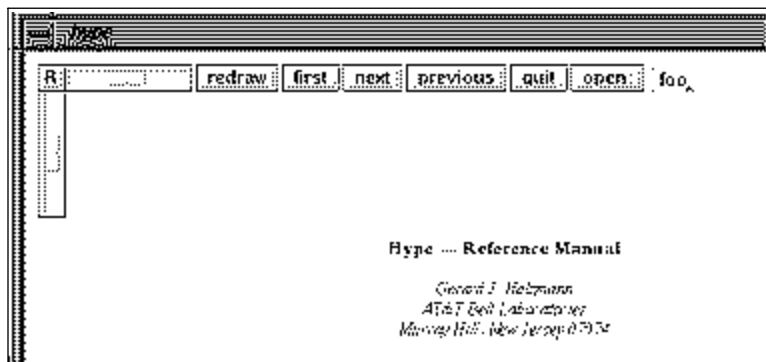


Figure 1 – Predefined Controls

The buttons `first`, `next`, and `previous` cause jumps to, respectively, the first page of the document, the page immediately following, or the page immediately preceding the currently displayed one.

The `redraw` button redraws the currently displayed page. This can be used when X does not correctly redraw a cluttered display, or, more importantly, to read in the new contents of a page, after the input file has been updated by user-commands in other windows. A useful mode of operation is to run `troff` commands repeatedly in one window, and to hit `redraw` after each such run to study the results. `Hype` will not get confused by changing page sizes, but it will not update its button-1 menu (with the page count) until the `open` button is pushed.

In the upper left hand corner of the display two scales are placed. The horizontal slider pans the display left or right; the vertical slider pans it up or down. A small square button, labeled `R` can be used to reset the sliders to their initial values.

2.2. PREDEFINED MENUS

When `hype` starts up, and when `open` is pushed the complete input file is scanned to determine how many pages exist. The page numbers are inserted into a button-1 menu, to facilitate browsing.

The button-2 menu is user-defined, with `index` hyper-text commands, as will be explained shortly.

The button-3 menu allows the user to increase or decrease the size of the displayed text, and it allows the user to toggle two-page display mode, with an adjustable gap between the pages. (Resize the X-window manually to allow for the two pages to be visible simultaneously.)

2.3. DEFINING A PAGE GRID

`Hype` accepts a command-line option to set a page-grid. Starting the program as:

```
$ hype -g ury lly urx llx
```

where `ury`, `lly`, `urx`, and `llx` are integer numbers, `hype` will draw a grid on each page that can be used to judge page balancing, e.g., for book-production. The integer coordinates are specified in `troff` units. For a device resolution of 720 dots per inch, for instance, the command

```
$ hype -g 720 6264 1080 4536
```

draws a box of 4.8 inches wide and 7.2 inches high, with its upper right hand corner 1.5 inch from the left, and 1 inch from the top of each page.

2.4. BITMAPS

Half-toned images can be displayed, using the macros from `mpictures(6)`. But, where the macros from `mpictures` expect images in postscript format (for output to printers) `hype` expects also the presence of a file with the same image encoded in standard X11 bitmap format (see for instance, the samples in `/usr/include/X11/bitmaps`). To make both screen display with `hype` and printed output with `lp` work from the same source files, prepare two image files: one ending in the suffix `.ps` with the postscript encoded image, and one with the same name, but ending in the suffix `.bm` holding the X11 bitmap format of the image. Provide the postscript version of the name to the `.PI` macro from `mpictures(6)`. To display the image, `hype` will replace the suffix `.ps` with `.bm` and open the corresponding bitmap image for display. The bitmap image must be scaled correctly, given the screen resolution of the target display (approximately 95 dots per inch on the NCD X-terminals). That is: if the size of the image is specified as `w` by `h` inches in the `.PI` macro, the bitmap image should be `w×95` by `h×95` pixels, to produce the exact effect of a printed copy of the same page.

EXAMPLE

Figures 1 and 2 from this document were produced in this way. Figure 1, for instance, with the black border provided by `pic(1)`, is produced as follows:

```
.KF
.mk
.PI figure1.ps 1.87i,4i,170u,720u w
.rt
.PS
scale=1
box wid 4i ht 1.87i
.PE
.I
.tl "Figure 1 \- Predefined Controls"
.R
.KE
```

The postscript version of the image is stored in file `figure1.ps`. The image is specified to be 4 inches wide, 1.87 inches high, and is placed at an offset of one inch (`720/720`) from the left margin, and 0.24 inches (`170/720`) down the page, from the point where the mark `.mk` was placed.

When parsing the `.PI` line, `hype` will open a file named `figure1.bm` which holds an image of the correct width and height for the display used, in X11-bitmap format. The resolution of the NCD terminals is roughly 95 pixels per inch, giving an image resolution of `380×178` pixels for the bitmap file.

Note that `lp` will automatically scale the postscript version of the image if it happens not to be the correct size. `Hype` performs no similar service for the bitmap versions.

Details on the generation of the postscript and bitmap files can be found in a later section of this memo, on Pre- and Post-Processing.

3. HYPER-TEXT SUPPORT

`Hype` recognizes 23 different commands. Each command can be embedded in the source text, which is translated by `troff` into a device control line, only interpreted by `hype`. The commands can also be used at runtime, by sending them to `hype`'s control channel: a named pipe in the file system.

3.1. THE CONTROL CHANNEL

The control channel is by default named `/tmp/hype_user`, where `user` is your login name. If a file with this name already exists when `hype` is started, a warning message is printed on the standard output and `hype` replaces the file with its own copy.

The warning message typically means that the last copy of `hype` died a sudden death and failed to remove its control channel. Another possible reason can be that the same user wants to run multiple copies of `hype` on the same machine. In this case, by default the last started copy will own the control channel and remove it when it quits. `Hype` also allows the user to specify alternative control channel names to override the default, e.g., when all copies of `hype` must be accessible at runtime through distinct command channels. For instance,

```
$ hype -c /tmp/hype_user2
```

forces the new copy of `hype` to read commands from the differently named channel.

3.2. CONTROL LANGUAGE

The 23 available `hype` commands can be grouped into roughly two broad classes: those that duplicate the predefined controls already discussed, and those that support more specific hypertext-like functions.

In the first class we find 16 commands. First, there are the five commands that perform the function of the push buttons at the top of the display, carrying the same names: They take no arguments. They are:

```
redraw first next previous quit
```

The next four commands manipulate the position of the horizontal and vertical scales, in the upper left corner of the display:

```
up down left right
```

Each of these accepts a single numeric argument, between 0 and 500. That is, the displayed image can be shifted in either direction by up to 500 `troff` units. The moves are absolute. So to reset the scales to their initial values it suffices to issue the commands `up 0` and `right 0`.

The next group of seven commands performs all the functions from the predefined button-3 menu. None take arguments. They are:

```
larger smaller original 2page 1page offmore offless
```

The first three of these commands affect the print size of the display. The next two set the display in either two-page side-by-side mode, or in (the default) single page mode. The last two commands can be used to increase or decrease the gap between the two pages in two-page mode. (Note, there is no argument to these last two commands; the increments and decrements are by a predefined amount.)

The next class of commands provides more specific hypertext-like support, this class includes commands for userdefined push-buttons, arbitrary file access and page location, and arbitrary UNIX® system calls. Each of these command will be discussed in the sections that follow.

3.3. USERDEFINED BUTTONS

SYNTAX

```
button name; command[; y]  
delbutton
```

DESCRIPTION

Each button placed on the screen is given a name and it is linked to a command that will be executed when the button is pushed. The command is itself can be any valid hyper-text command. The command `delbutton` removes all buttons that were placed in this manner. By default, the command `delbutton` is executed at each page change. This user can change the default behavior with the help of a special system command (see below).

EXAMPLE

Commands embedded in troff source:

```
\X'button goodbye cruel world; quit'  
\X'button remove me please!; delbutton'  
\X'button Bibliography; open Bibl.out'
```

The same commands executed at runtime, assuming that user is your login name:

```
$ troff Doc > Doc.out  
$ hype Doc.out &  
$ echo "button goodbye cruel world; quit" > /tmp/hype_user  
$ echo "button remove me please!; delbutton" > /tmp/hype_user  
$ echo "button Bibliography; open Bibl.out" > /tmp/hype_user
```

3.4. USERDEFINED MENUS

SYNTAX

```
index label; command  
delindex
```

DESCRIPTION

The working is almost identical to that of the button commands, but this time the commands are made accessible via the button-2 menu, which is empty by default. By default the command `delindex` is called each time the file that is being accessed is re-opened. The user can override this default with the help of a special system command (see below).

EXAMPLE

The `index` command is most useful when embedded in the `troff` sources, for instance by 'overloading' the `.NH` macro from the `ms` package, as follows, as was done for this document:

```
.de HI  
.NH \\$1  
\\$2  
&\X'index page \\n% - \\*(SN \\$2; page \\n%'  
..
```

The new macro `.HI` can now be used instead of `.NH`.

This section, for instance, begins in the `troff` source:

```
.HI 2 "USERDEFINED MENUS
```

This call produces the embedded command in the raw `troff` output:

```
x X index page 5 - 3.4. USERDEFINED MENUS; page 5
```

which is interpreted by `hype` when this text is read, and placed in the button-2 menu.

The same effect can also be achieved by searching the `troff` output for section headers with a separate post-processing tool, to determine the page numbers. The result of that search can be formatted again as `index` commands, and send to the control file at run-time, for instance as:

```
echo "index Section 3 - 3.4. MENUS; page 5" > /tmp/hype_user
```

The best way to do this is to include menu-entries for each `troff` output file in a script, and to use one of `hype`'s system commands to switch to the appropriate entries each time a new file is opened.

Inside menu `index` terms the slash character `'/'` has a special meaning. The last such character in an `index` term defines the cut-point for a hierarchical menu: everything after the slash goes in a submenu, everything before it in the default toplevel menu. For instance, use entries such as

```
echo "index Section 3 / 3.3. BUTTONS; page 4" > /tmp/hype_user  
echo "index Section 3 / 3.4. MENUS; page 5" > /tmp/hype_user
```

to split the menu entry into two levels. Only a one-level hierarchy is supported.

3.5. FILE ACCESS

SYNTAX

```
open filename [pagenumber [y-coordinate]]  
page pagenumber
```

DESCRIPTION

The `page` command is a shorthand for an `open` command, that can be useful in cases when the filename for the `troff` output is not (yet) known, for instance as illustrated in the previous section for embedded commands. It causes a page jump within the currently displayed file to the pagenumber specified, if it exists, or else to the first page of the file.

The more general `open` command allows the user to change the input file during a previewing session. With just one argument, it switches to the first page of the new document. A second argument can be used to override the default pagenumber. A third argument can specify a position on the page to be marked with a dotted line after the jump. The y-coordinate is specified in `troff` units (e.g., 720 units per inch). Coordinates matching words or phrases on a given page can be extracted with the post-processing tools discussed elsewhere in this report.

When `page` commands are first encountered (either at runtime, or embedded) `hype` translates them into unambiguous `open` commands, by including the current filename, before they are executed or stored for later execution.

3.6. THE SYSTEM COMMAND

SYNTAX

```
[fFpP] system unix_command
```

DESCRIPTION

An unprefixd `system` command causes `hype` to execute the `unix_command` that was specified with a regular `system(3)` call, appending the current filename and pagenumber as arguments. If arguments are present, the two extra ones will appear at the end of the list.

The most useful application is when the `unix_command` is a script that, based on the filename and the pagenumber, computes a set of appropriate hyper-text commands, and sends them to the running program. An example of such a script could be to scan a pre-computed data base of hyper-text links for all links that match the current page, and place buttons near the position of each such link on the current page. The buttons can be tied to jump to the desired new location in the document, within the same file or in another. Return jumps can be computed and inserted in a similar fashion.

There are four special flavors of the `system` command. Prefixing the command with a single letter `F` causes `hype` to both execute the command when it is first issued and to store it for a new execution each time a new file is opened, (or the old file re-opened). Note, however, that since `hype` always executes embedded `index`, `button`, and `system` commands found in the document in the first scan of the contents after a file open request, the upper-case prefix `F` is only useful as a runtime command – its working is indistinguishable from an unprefixd `system` command when it is embedded in the text.

Prefixing the `system` command with the lower-case letter `f` causes `hype` to execute the command and store it for execution each time the current file is closed. If both an `f` and an `F` script has been defined, the order of execution on a file change is:

- Close the current file
- Execute the `f` script
- Open the new file, prescan the text for embedded commands
- Execute the `F` script.

These definitions will override the default behavior of `hype`, which is to clear the `button-2` menu just after the closing of the current file.

Prefixing the command with a `P` causes `hype` to execute the command and to store it for another execution each time after a new page has been displayed.

Prefixing the command with a lower case `p` does the same just after the current page is cleared from the display. These definitions override the default behavior provided, which is to delete all buttons that the user placed just after the current page displayed is cleared.

EXAMPLE

Consider the command

```
p system clr_links
P system new_links
```

where `clr_links` and `new_links` are unix-commands.

A shell-script `clr_links` can contain the command:

```
echo "delbutton" > /tmp/hype_user
```

causing the current userdefined (hyper-link) buttons to be removed (which is strictly speaking redundant, since this is also the default behavior of `hype`), and `new_links` can be an awk-script used to replace them, defined as follows:

```
$ awk -f links.awk $1 $2 < Index.file > /tmp/hype_gerard
```

The two arguments are the filename and the pagenummer, which are by default always provided by `hype` upon execution of the `system` command.

The data-base file referred to here, `Index.file`, contains lines of the following format:

```
$ cat Index.file
0.1      out0 7 2915      outNotes 269 3212
0.2      out0 7 4801      outNotes 269 3334
0.3      out0 8 2207      outNotes 269 3456
0.4      out0 9 1723      outNotes 269 3578
1.1      out1 1 3850      outNotes 269 3940
```

The information for this file is extracted from the raw `troff` output files `out0`, `out1`, `outNotes`, etc. The file specifies a label-name (a combination of a chapter number and a footnote number), the file, pagenummer, and page coordinate where a reference to the footnote is made, and a file, pagenummer and page coordinate where the full text of the same footnote appears.

The complete awk-file named `lines.awk` that interprets this information looks like this:

```
$ cat lines.awk
BEGIN    {      if (ARGC != 3) exit
              file= ARGV[1]; pno=ARGV[2];
              ARGC -= 2; ARGV[1] = ARGV[2] = 0
            }
$2==file && $3==pno    {
                      printf("button %s; open %s %s; %s\n", $1,$5,$6,$4)
                    }
$5==file && $6==pno    {
                      printf("button %s; open %s %s; %s\n", $1,$2,$3,$7)
                    }
}
```

This script extracts lines that have a matching filename and pagenummer, and reformats those lines as `button` commands. Each button is placed at the line where the footnote reference, or footnote text appears, and is tied to a `open` command that brings the user from one to the other. The two scripts are automatically repeated once for every new page.

4. PRE- AND POST-PROCESSING

4.1. CREATING X11 BITMAP FILES

Apart from the X11 tools that produce and edit bitmaps, two local tools are available that can be used to make `hype` and `lp` handle images correctly.

The first step in preparing an image for reproduction is to scale an arbitrary 1127-style grayscale or color image to the required dimensions. There are several local tools available to do this. One such local tool is `resize`, which takes three arguments:

```
$ resize 800 900 pigeons > scaled_pigeons
```

would scale the input file `pigeons` to be 800 pixels wide and 900 pixels high, and writes the result into the file `scaled_pigeons`.

The next command is `mkbmap`, which can be used to convert an arbitrary image into an X11-standard bitmap. It takes the file-name as an argument and has no other options.

```
$ mkbmap scaled_pigeons > pigeons.bm
```

The last command is `postdmd`, which takes the bitmap file and converts it into a postscript file.

```
$ postdmd pigeons.bm > pigeons.ps
```

The last two commands take the filename as an argument, and support no other options.

4.2. PAGE BALANCING

A program, called `stretch`, can be run over arbitrary `troff` output files to improve the page-balancing. The program takes the following optional arguments:

```
$ stretch -?
usage: stretch [options] filename > output
       [-v]           Verbose
       [-r Ntop Nbot] default: 500 6100
       [-t Targetbot] default: 6034
       [-w Minv Maxv] default: 100 200
       [-d Maxdrift]  default: 10
```

All numeric arguments are specified in `troff` units. For instance, if the device resolution is 720 dots per inch, each number divided by 720 gives the distance in inches. The meaning of the last four options is:

```
-r  the range on the page in which adjustments will be made
-t  the required location of the last line within that range
-w  the minimum and maximum existing vertical spacing that
    can be modified
-d  the maximum modification that may be made
```

By default, `stretch` attempts to look for paddable vertical spacings within an area from 500 to 6100 `troff` units from the top of each page; it attempts to stretch each page to precisely 6034 `troff` units long; it will adjust only vertical spacing that is currently between 100 and 200 units height (i.e., 0.14 to 0.28 inches, or approximately a linewidth), and it will never make a change larger than 10 `troff` units (1/72 inch) for each paddable space found.

4.3. FINDING KEYWORDS – PREPARING AN INDEX

We discuss two methods for index preparation in this section: the first requiring changes to the `troff` source text, the other parsing only the `troff` output.

The first method is to explicitly mark each term or phrase for the index in the `troff` source with an inline escape command, defined as a macro as follows:

```
.de MT
\X'keyword \\\$1'
..
```

A macro call such as:

```
.MT "a phrase for the index"
```

causes a tag to be inserted into the troff output that looks as follows:

```
x X keyword a phrase for the index
```

A post-processing tool called `key` can now be run over the troff output file to locate all these lines (invisible to other post processing tools such as `lp`) and print out the page number for the exact location where the phrase starts in the output file, followed by the phrase itself.

With an optional flag `-k`, the program appends the y-coordinate of each phrase, which can be used for establishing hyper-links. The y-coordinate is reported in troff units.

EXAMPLE

Near the start of this document we made a cross reference to the current main section. It appears as follows in the troff source:

```
.MT "r_link0"
.LP
Details on the generation of the postscript and bitmap files
can be found in a later section of this memo, on Pre- and
Post-Processing.
.HI 1 "HYPER-TEXT SUPPORT
.LP
```

At the start of this main section, the target reference occurs:

```
.HI 2 "FINDING KEYWORDS - PREPARING AN INDEX
.MT "t_link0"
.LP
We discuss two methods for index preparation in this section:
the first requiring changes to the
.CW troff
source text, the other parsing only the
.CW troff
output.
```

We have chosen the name `link0` for this particular hyper-text link, and labeled the place of reference with the prefix `r_`, and the place pointed to with the prefix `t_`. The precise names and prefixes are of course arbitrary.

Running the keyword post-processor now produces, for these two entries:

```
$ key -k hype.out
3      3 5964
8      8 6504
```

Meaning that the reference appeared on page 3, at 5964/720 inches from the top of the page, and the target appears on page 8, at 6504/720 inches from the top of that page. With a shell script we can separate the references from the targets (assuming there will be many more than shown here), and with the help of an awk script we can merge the sorted files into entries such as these:

```
$ cat Index.file
link0      hype.out 3 5964      hype.out 8 6504
```

The post-processing script, looks as follows:

```

$ cat compute_links
key -k hype.out > Rin
grep r_ Rin | \
  sed "s/r_link/hype.out link/" | \
  awk ' { printf("%s %s %s %s\n", $3, $2, $1, $4) }' > Rout
grep t_ Rin | sed "s/t_link/hype.out link/" | \
  awk ' { printf("%s %s %s %s\n", $3, $2, $1, $4) }' >> Rout
awk '
{
  Link[$1] = Link[$1] " " $2 " " $3 " " $4
}
END {
  for (name in Link)
    printf("%s %s\n", name, Link[name])
}' < Rout | sort -n +1 > Index.file
rm -f Rin Rout

```

The command `new_links`, discussed earlier, can now be used to scan the `Index.file` once for every new page, to place the hyper-text buttons, as illustrated for one such button in Figure 2. This command to place the links can be either embedded or issued at runtime. In the current document it is an embedded command.

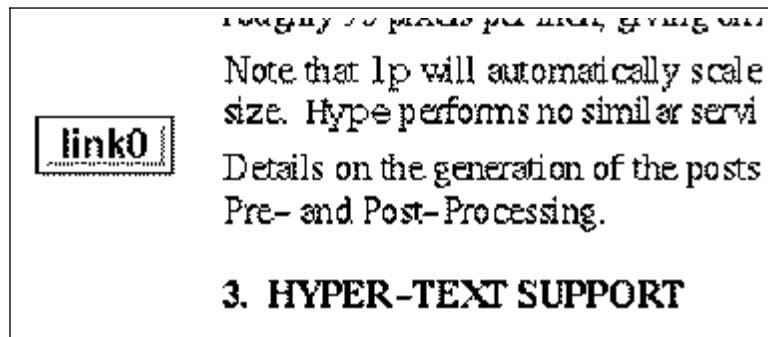


Figure 2 – User-Defined Hypertext Link Between Pages 3 and 8

An alternative way of generating an index is to generate all sequences of, for instance, three words in the `troff` output files, with page number and y-coordinates appended. The list can be sorted alphabetically, and then matched against a separately compiled list of index-terms. In this manner an accurate index can be prepared without any changes to the `troff` sources. Because of the sorting that is now possible, this method of index generation is quite fast (ideal for book preparation).

5. TEMPORARY FILES

Hype creates two temporary files:

```

/tmp/hype.XXXXXXX
/tmp/hype_user

```

The first is only used if the program is run as part of a pipe, as in:

```

$ tbl hype.tm | troff -ms | hype

```

The text read from the standard input is saved in the temporary file, which is then used as if it were a regular input file provided on the command line. (Note that without this feature arbitrary browsing between the pages of a document would become impossible.)

The second temporary file is a named pipe, that can be used to send runtime commands to `hype`.

6. SUMMARY

Hype is a hyper-text browser with a simple general control language that allows the user to customize its behavior. Older documents, for which perhaps the troff source was lost, can still be adapted for the browser with the post-processing tools discussed. The hyper-text commands can either be embedded into the troff source, into the troff output, or kept completely separate in database files that are scanned by user-definable command scripts.

The two tables below briefly summarize the available commands.

Commands Supporting Hypertext Controls					
Command	Nr. Args	1st Arg	2nd Arg	3rd Arg	Page
button	2-3	label string	hype command	y-coordinate	4
delbutton	-				4
index	2	label string	hype command		5
delindex	-				5
open	1-3	file name	page number	y-coordinate	6
page	1	page number			6
system	1	unix command			6

Arguments to the above commands are separated by semi-colons, and may contain spaces and tabs, but no newlines (and no semi-colons). All y-coordinates are specified in integer troff units. Typically, the resolution are 720 troff units per inch.

The remaining commands provide runtime control mimicking the predefined user-interface controls: the top row of push-buttons, the scales, and the button-3 menu.

Commands Duplicating Predefined Controls						
Control	Command Names					Page
Buttons	redraw	first	next	previous	quit	4
Scales	up	down	left	right		4
Menu-3	larger	smaller	original			4
	2page	1page	offmore	offless		4

All commands can either be embedded in the troff source, with troff's escape commands, as follows:

```
\X'hype command'
```

or they can be send at runtime to the control channel of a running hype, as follows:

```
echo "hype command" > /tmp/hype_user
```

The runtime commands can, of course, be issued by preprocessors that undertake to control hype's user interface, and take over the role of the predefined controls.