The THINC Server

Ricardo Baratto
ricardo@cs.columbia.edu
Network Computing Lab - Columbia University

February 10, 2004

1 Introduction

Write me

2 Code Organization

An attempt has been made to give each source file a meaningful name reflecting the file contents. In addition, a comment at the top of the file should give a brief description of the code in it. Some files will have longer comments, explaining data structures, organization, flow of control, etc. Read them, they provide valuable information.

The code is organized in a flat structure, with all source files in one directory. It uses X's Imake system to produce its Makefile. Providing instructions on how to compile THINC is beyond the scope of this document.

The directory *doc* contains random bits of useful information about THINC and its surroundings. The following is a list of all the source files along with their (unedited) top-of-file comments:

- *thincBitmap.c*: Functions for Glyph and Bilevel messages
- *thincBits.c*: Bitmap and bilevel handling functions
- *thincBuffer.c*: Functions for Buffering of Messages
- *thincBuffer.h*: Buffering of Messages
- *thinc.c*: Module for remote display. Should work with any display driver.
- *thincClient.c*: Client handling functions
- *thincCommandBitmap.c*: Functions for offscreen manipulation of BITMAP (glyph/bilevel) commands
- *thincCommandFill.c*: Functions for offscreen manipulation of FILL (solid/pixmap) commands
- *thincCommand.h*: Command structure definitions
- *thincCommandRaw.c*: Functions for offscreen manipulation of RAW commands
- *thincCompress.c*: Functions for data compression
- *thincCompress.h*: Header for data compression
- *thincCopy.c*: Functions for COPY message
- *thincCursor.c*: Functions for hw cursor handling.

- *thincFill.c*: Functions for Solid and Pixmap Fill messages
- *thincGC.c*: Wrapper functions for drawing (wrap GC ops)
- *thincGC.h*: Helper macros for GC functions
- *thinc.h*: Header for thinc module (Remote display ala VNC - SunRay)
- *thincKeyb.c*: ThincKeyboard functions
- *thincKeyb.h*: Keyboard mappings. From xf4vnc (xf4vnc.sourceforge.net)
- *thincList.c*: linked list functions
- *thincList.h*: Header for doubly linked list
- *thincMouse.c*: ThincMouse functions
- *thincMsg.h*: Header for Thinc messages
- *thincNetwork.c*: Network utility functions for thinc
- *thincPixmap.c*: Pixmap functions
- *thincPixmap.h*: Header for offscreen pixmap operations
- *thincRaw.c*: Functions for RAW message
- *thincResize.c*: Resize handling functions
- *thincSchedule.c*: Command scheduling functions
- *thincStats.c*: Functions for statistics
- *thincText.c*: Text drawing functions
- *thincUtil.c: THINC util functions*:
- *thincUtil.h*: Header with utility functions for thinc
- *thincVideo.c*: XVideo support
- *thincWindow.c*: Window functions
- *thincWindow.h*: Header for window related functions

Naming: Everything THINC related in the code, in particular functions and data structures, contains the prefix *thinc*. All data structures are named following XFree86's naming conventions: They are type-def'ed to a single-name data type, to which Rec and Ptr are appended. The Rec type represents the actual structure, while the Ptr type represents a pointer to the structure.

Return Values: For portability, no function should use XFree86's Bool type (there are some in the current code which do but will be eventually fixed). Use int instead. Standard return values have been defined for the server code, which are defined in thincUtil.h. Generic ones are T_OK and T_ERROR which are self-explanatory. Additional return values defined for special cases are: T_TRUE, T_FALSE, T_YES, T_NO, T_CLOSED, T_WOULDBLOCK, T_TIMEOUT. Feel free to define new ones to fit your needs.

Hacking THINC: There is a document detailing THINC's hacking guidelines, which can be found at *doc/hacking*. Please follow them as closely as possible.

3 Data Structures

The two major structures in the code are:

- thincScreen write me
- thincClient wirte me

In addition, many smaller structures are used throught the code to represent different pieces of information needed for a particular section of the server. A selection of these follows.

- *thincVideo, thincClVideo*: Contain all video related information, for the server and the client, respectively. Video formats supported and active video streams are stored in here.
- *thincSched*: Represents the scheduler which manages the traffic sent from the server to a particular client. Each client has a scheduler associated with it.
- *thincCmd*: Representation of a THINC command.

4 Execution

4.1 Initialization

Write me

4.2 Running

There are 3 points at which the X server hands control to thinc:

1. thincWakeupHandler
2. thincBlockHandler
3. Draw Functions

4.2.1 WakeupHandler

Called when something happened which made the server wakeup. In normal X this happens when an X client sends a request to the server. We do not deal with those events here. However, we do use this mechanism to control our own clients.

We check if there is either a new connection, one of our connected clients sent a message, or a connection was closed. If a new connection is received, thincNewConnection() is called. If an active client talks to us, thincHandleClient() is called.

- thincNewConnection write me
- thincHandleClient write me

4.2.2 BlockHandler

Write me

4.2.3 Draw Functions

Write me

4.3 Shutdown

Write me