This server will only run on JK microsystems' Flashlite computers.

Setup/Configuration:

WebTCP uses the WATTCP TCP/IP stack and libraries. The `WATTCP.CFG` file must be properly configured and in the same directory as the web server. As a bare minimum, you must change the `my_ip=xxx.xxx.xxx.xxx` line in the CFG file. Enter the IP address of the Flashlite here. If your network requires a sub net mask, enter it on a line `netmask=xxx.xxx.xxx.xxx.` Gateway and name server addresses are entered in a similar manner. If the `my_ip` line is omitted, WATTCP will attempt to configure itself using BOOTP.

A typical CFG file might look like this:

```
my_ip=168.150.253.18
subnet=255.255.254.0
gateway=168.150.253.254
nameserver=209.21.20.253
nameserver=209.94.127.2
```

See the WATTCP documentation for more information about the configuration file.

WATTCP communicates TCP/IP packets via a packet driver. This driver interfaces with the hardware to send the packets over the available network. The `FLASHPKT.COM` packet driver is required for communication with the Flashlite Ethernet controller. A packet driver must be installed before running WebTCP.

The time zone defaults to pacific daylight time with an 8 hour adjustment to GMT. To properly configure WebTCP for your time zone, include the following environment variable:

`TZ="`*`xxxnyyy`*`"`

`xxx` - primary time zone name (PST, MST, EST, etc)
`n` - difference to GMT
`yyy` - daylight savings time zone name (PDT, etc)

For example, include `SET TZ=MST7` in the STARTUP.BAT file to set the time zone to mountain time.

The server does not provide much console output during default operation. Console I/O is time intensive and not generally required. If the server is started with the `-V` command line argument, it will be verbose and all hits as well as some other information will be logged to the console.

A typical `STARTUP.BAT` file might look like this:

```
FLASHPKT 0X60
SET TZ="PST8PDT"
SET HDWR=YES
DFS
TSR
WEBTCP -V
```

The server is HTTP/1.0 compliant but only supports "GET" and "HEAD" commands. Scripting (CGI) is not supported at this time.

Recognized file extensions are:

HTM	HTML document	text/html
SHT	server parsed HTML	text/html
PDF	Adobe Acrobat	application/pdf
GIF	GIF graphics	image/gif
JPG	JPEG graphics	image/jpeg
ZIP	zip compressed archive	application/zip
TXT	ASCII text	text/plain
	default	*/*

Defaults:

HTML page:	homepage.htm
HTML directory:	\www (on current drive)

Files are searched for in the following order:
1 - Internally generated (reserved) files
2 - Files in the dynamic files system (if present)
3 - DOS files

The first instance of a matching file name will be returned to the requesting client.

Special functions:

Dynamic Files from the DFS TSR:
The DFS TSR provides a method to serve complete HTML pages received from a serial device. DFS can handle up to four files, each up to 4k in length. See the DFS manual for exact configuration, interface and protocol information.

When a file is requested, a table is checked in memory (starting at INT 30h CS:108) for that file. If the file exists, a pointer is returned to the server along with the file length. The server then sends the file as if it were a disk file. If the file is not present, the disk file system is checked. If the file is not on the disk, the 404 (not found) message is returned.

Files handled by DFS must be complete HTML or graphic files. The server does not parse or modify the file before sending it to the requesting client.

HDWR.SCR:
The internal HDWR.SCR page allows output to hardware ports A, B and the AB direction register using the GET method of form input. See the sample page for an example. A request is compiled from the form input and then written to the hardware. Read-back is not currently available.

Requests should take the following form:

```
http://129.99.19.29/hdwr.scr?port=xx&v=yy&v=yy&act=zz
```

xx	- A or B or DIR_AB
yy	- hex value of bit mask (multiple masks will be combined with OR)
zz	- SET or CLEAR

This feature can be a bit dangerous and is only enabled if the HDWR environment variable is set to YES.
ie: `SET HDWR=YES` in the STARTUP.BAT file. User/password features may be added in the future.

Please be aware that internally generated pages and take precedence over external files.

Server Parsed Files (.SHT files):
Server parsed files allow the insertion of hardware parameters into static web pages. At present, the interface only supports data output (from the Flashlite to the client) not input (from the client to the Flashlite).

A user supplied TSR servicing INT31h returns information to be inserted into the web page. Data returned from the TSR is not interpreted by the server, just passed to the client. This data may consist of simple text or HTML type tags for graphics or links to other pages.

Static page must be less than 12k and the total page size after parsing must be less than 16k.

The include tag takes the form: *`<!-- #include file="XX" -->`*

For example, the static page

```
<html>
<body>
The temperature is <!-- #include file="1" --> degrees Celsius.<br>
</body>
</html>
```

might return: `The temperature is 28.2 degrees Celsius.`

The include tag is case and space sensitive. The value `XX` refers to the location in the lookup table of the desired data. `XX` is a decimal value from 0 up to the maximum number of entries in the table. Indices outside the table will be ignored.

Parsing is only supported for static (DOS) pages, dynamic pages (from DFS) will not be parsed (even if the .SHT extension is present).

WebTCP looks for the parsing TSR signature (0x55AA) at offset 18 and version number (1 byte) at offset 20 from the location given in the interrupt table for the INT31h service routine. After the server verifies that the TSR is present, all other communication is done via an INT31h. The ISR receives its function request in AH and currently requires two functions:

0 - Return the number of table entries as an integer value in AX.

1 - Return data requested in AL
AX= Length of return data
BX= Pointer to first byte of return data
CX= TSR Data Segment
A returned length of 0 indicates an error.
The TSR should verify that the request is within the valid table range and ignore out of range requests.

The TSR may be written in the language of your choice as long as the previous requirements are met. Every effort should be made to keep the time spent in the TSR and its resident size to a minimum. If extensive processing is necessary, the TSR should be written in assembly rather than a high level language. Please see the file TSR.C for an example.

Notes:
Example pages were developed using Netscape 4. Although some verification was done using Microsoft IE3, there may be cases where pages do not display (or update) properly. Some of the examples use JAVASCRIPT which does not have a consistent implementation on all browsers.

Problems with the example pages are most likely the result of the page content, the scripts, or subtle differences between Netscape and Microsoft IE, not the WebTCP server.

Please check our website for updates to the WebTCP server.