“Very rough” source code for the HTML extension

Drop these bits into your existing server. They’re C89/Windows-friendly and reuse your `send_all(...)`. The idea:

- `GET /` or `/index.html` → serve `./index.html`
- `GET /static/...` → serve files from `./static/...`
- (optional) `GET /harbour-writer` → call a Harbour function that returns HTML

1) Small helpers (top of your C file, after includes)

```
/* --- helpers for routing and mime --- */
static int starts_with(const char *s, const char *p) {
    while (*p) { if (((*s|32) != (*p|32))) return 0; s++; p++; } return 1;
}

static void strip_query(char *path) {        /* in-place: "/x?a=b" -> "/x" */
    char *q = path ? strchr(path, '?') : NULL;
    if (q) *q = '\0';
}

static const char *guess_mime(const char *path) {
    const char *ext = path ? strrchr(path, '.') : NULL;
    if (!ext) return "application/octet-stream";
    if (_stricmp(ext, ".html")==0 || _stricmp(ext, ".htm")==0) return "text/html; charset=utf-8";
    if (_stricmp(ext, ".css")==0)   return "text/css";
    if (_stricmp(ext, ".js")==0)    return "application/javascript";
    if (_stricmp(ext, ".json")==0)  return "application/json";
    if (_stricmp(ext, ".png")==0)   return "image/png";
    if (_stricmp(ext, ".jpg")==0 || _stricmp(ext, ".jpeg")==0) return "image/jpeg";
    if (_stricmp(ext, ".svg")==0)   return "image/svg+xml";
    return "application/octet-stream";
}

static void BuildAndSendHtmlResponse(SOCKET s, const char *htmlBody, const char *connection) {
    char header[256];
    int bodyLen = (int)(htmlBody ? strlen(htmlBody) : 0);
    int headerLen = _snprintf(header, sizeof(header),
        "HTTP/1.1 200 OK\r\n"
        "Content-Type: text/html; charset=utf-8\r\n"
        "Content-Length: %d\r\n"
        "Connection: %s\r\n"
        "\r\n",
        bodyLen, connection ? connection : "close");
    send_all(s, header, headerLen);
    if (bodyLen > 0) send_all(s, htmlBody, bodyLen);
}

/* simple static file sender using stdio (portable enough) */
static void serve_static_file(SOCKET s, const char *localPath, const char *connection) {
    FILE *f = fopen(localPath, "rb");
    if (!f) {
        const char *nf = "HTTP/1.1 404 Not Found\r\nContent-Length:0\r\nConnection: close\r\n\r\n";
        send_all(s, nf, (int)strlen(nf));
        return;
    }
    if (fseek(f, 0, SEEK_END) != 0) { fclose(f); return; }
    long sz = ftell(f);
    if (sz < 0) { fclose(f); return; }
    rewind(f);
```

```
    /* header */
    {
        char hdr[256];
        const char *mime = guess_mime(localPath);
        int hlen = _snprintf(hdr, sizeof(hdr),
            "HTTP/1.1 200 OK\r\n"
            "Content-Type: %s\r\n"
            "Content-Length: %ld\r\n"
            "Connection: %s\r\n"
            "\r\n", mime, sz, connection ? connection : "close");
        send_all(s, hdr, hlen);
    }
    /* body */
    {
        char buf[4096];
        size_t n;
        while ((n = fread(buf, 1, sizeof(buf), f)) > 0) {
            send_all(s, buf, (int)n);
        }
    }
    fclose(f);
}
```

2) (Optional) Harbour page that returns HTML

```
FUNCTION PAGE_HARBOUR_WRITER()
   LOCAL cHtml := ""
   TEXT TO cHtml
<!doctype html>
<html><head><meta charset="utf-8"><title>harbour writer</title></head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body></html>
   ENDTEXT
RETURN cHtml
```

3) Router additions in your GET branch

Inside your existing `if (stricmp(httpMethod,"GET")==0)` section:

```
/* normalize path before routing */
strip_query(httpPath);

if (_stricmp(httpPath, "/")==0 || _stricmp(httpPath, "/index.html")==0) {
    /* serve the main page */
    serve_static_file(ClientSocket, "./index.html", connection);
    alreadySent = 1;

} else if (starts_with(httpPath, "/static/")) {
    /* map /static/... -> ./static/... */
    char local[512]; _snprintf(local, sizeof(local), ".%s", httpPath);
    serve_static_file(ClientSocket, local, connection);
    alreadySent = 1;

} else if (_stricmp(httpPath, "/harbour-writer")==0) {
    /* dynamic HTML from Harbour */
    const char *html = NULL;
    hb_vmPushSymbol( hb_dynsymSymbol( hb_dynsymFindName("PAGE_HARBOUR_WRITER") ) );
    hb_vmPushNil();
    hb_vmDo(0);
    html = hb_parc(-1);
    if (!html) html = "<!doctype html><meta charset=\"utf-8\"><h1>empty</h1>";
    BuildAndSendHtmlResponse(ClientSocket, html, connection);
    alreadySent = 1;
```

```
/* ...else: your other GET API endpoints or 404 ... */
}
```

Place your UI assets like this:

```
your.exe
index.html
static/
  css/style.css
  js/app.js
  img/logo.png
```

4) Cloudflare Tunnel (very short dummy)

`config.yml`:

```
tunnel: zplan
credentials-file: C:\Users\<YOU>\.cloudflared\zplan.json
ingress:
  - hostname: app.mytest.com
    service: http://localhost:9090
  - service: http_status:404
```

Run:

```
cloudflared tunnel create zplan
cloudflared tunnel route dns zplan app.mytest.com
cloudflared tunnel run zplan
```

That's it: `https://app.mytest.com/` hits your EXE, which serves `index.html` (and `/static/...`) and still handles your JSON API routes.