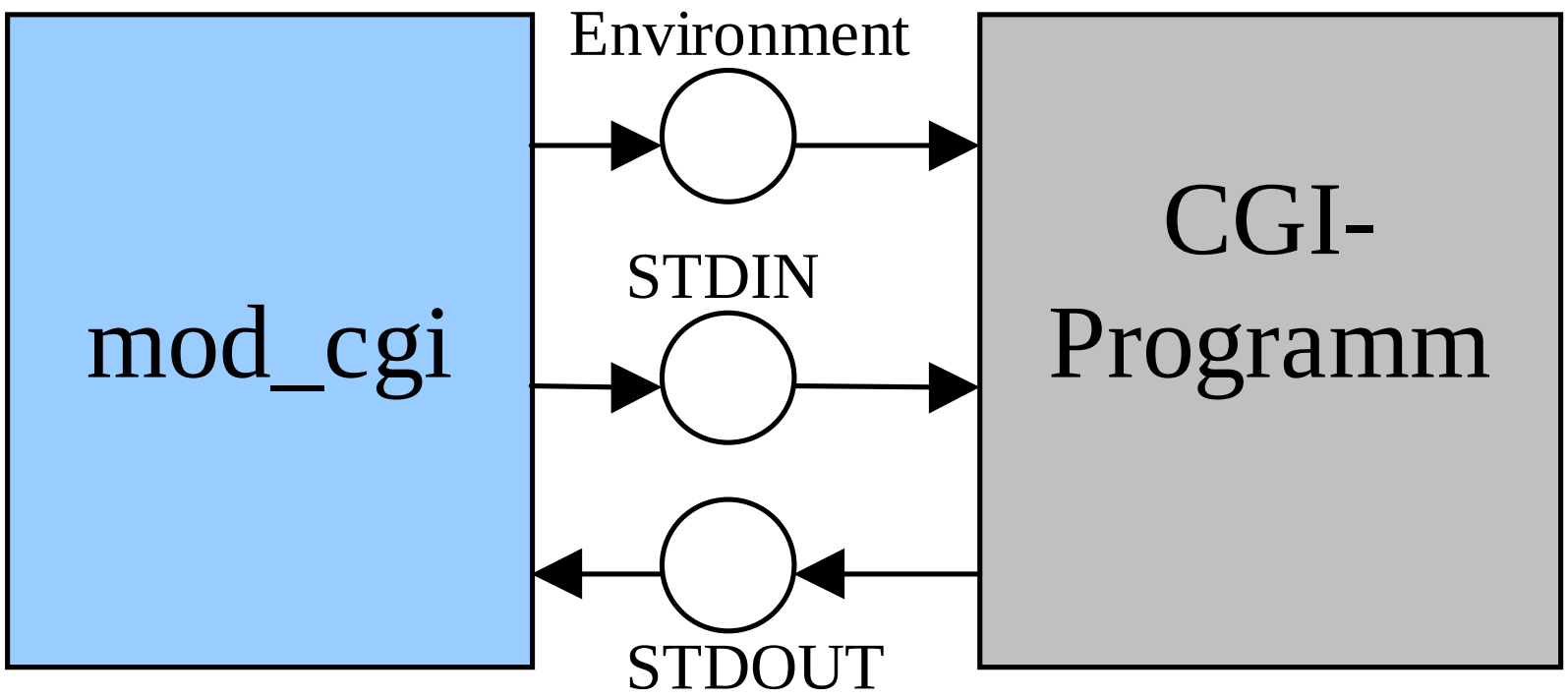
`mod_cgi` und Pipes

- Was sind Pipes und wie verwendet man sie?
- Wie kann man Umgebungsvariablen aus einem C-Programm heraus setzen und abfragen?
- Welche Prozeduren stellt Apache für die Erzeugung von Prozessen, Setzen und Lesen von Umgebungsvariablen und Schreiben und Lesen auf Pipes wie z.B. `StdIO` zur Verfügung?
- Stellen Sie vor diesem Hintergrund das Apache Modul `mod_cgi` vor.

Pipes

- eine Pipe (-line) ist ein Kommunikationskanal zwischen zwei Prozessen, der wie eine Datei behandelt wird
- `STDIN` und `STDOUT` sind vordefinierte Pipes und stellen i.d.R. eine Verbindung zur Tastatur bzw. zum Monitor her
- durch Umleitung der Pipes liefert Apache die Daten per `STDIN` (bei POST-Anfragen) und erwartet die Ausgabe in `STDOUT`
- in C++ kann man bequem mit `cin` bzw. `cout` arbeiten


Environment
mod_cgi
STDIN
CGI-
Programm
STDOUT

Erzeugung von Prozessen und Zugriff auf Pipes mit Apache

- Windows:
  - mittels `ap_bspawn_child` kann ein neuer Prozess gestartet werden:
    ```
    API_EXPORT(int) ap_bspawn_child(pool *p, int (*func) (void *, child_info *),
    void *data, enum kill_conditions kill_how,
    BUFF **pipe_in, BUFF **pipe_out, BUFF **pipe_err)
    ```
  - die Pipes müssen extra mit `CreatePipe` erzeugt werden, dabei wird auch ihr Typ festgelegt (Zerstörung mit `CloseHandle`)

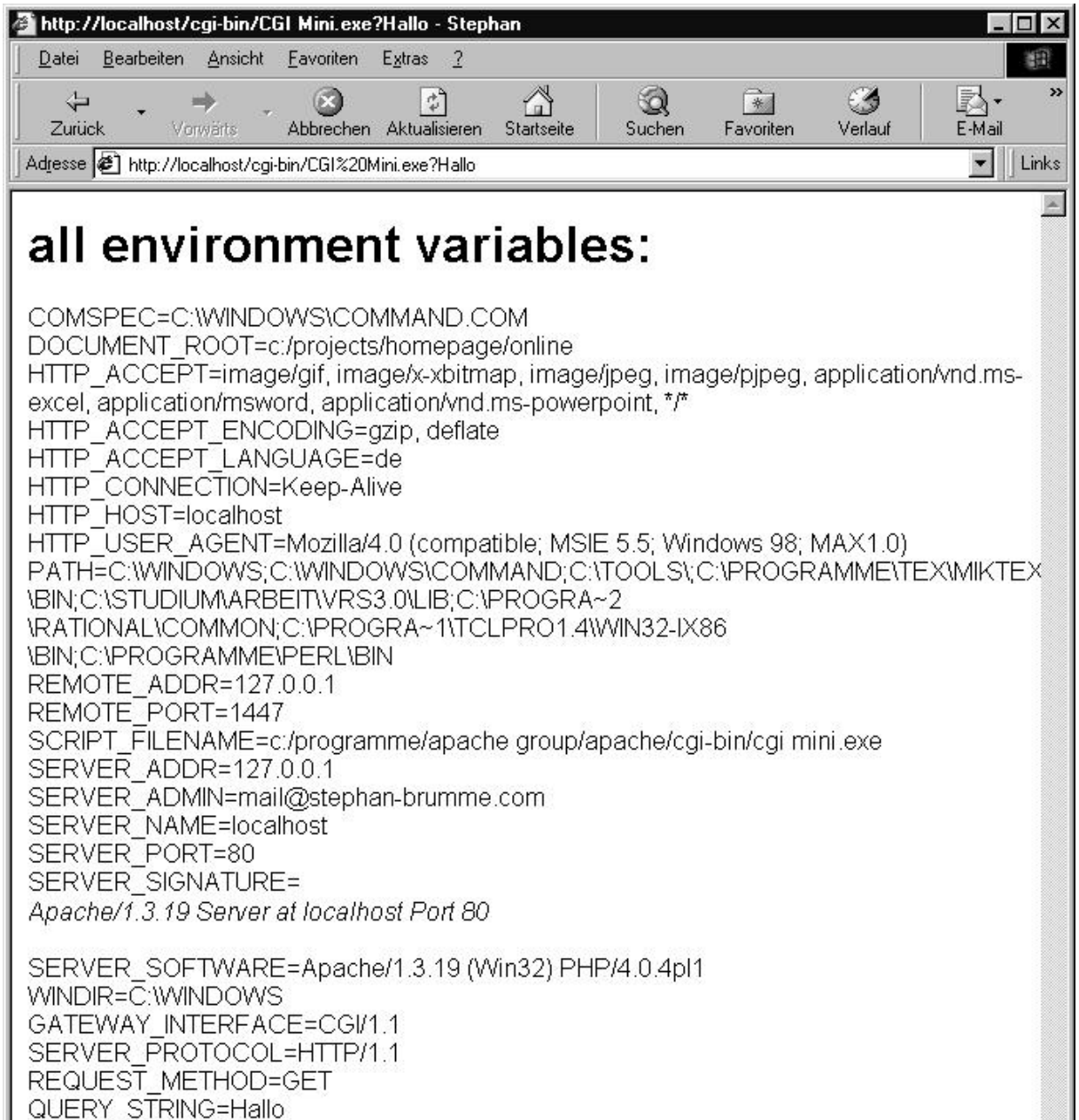
- Unix:
  - über `spawn_child_core` werden neue Prozesse angelegt (wird indirekt von `ap_bspawn_child` aufgerufen):
    ```
    static pid_t spawn_child_core(pool *p, int (*func) (void *, child_info *),
    void *data,enum kill_conditions kill_how, int *pipe_in, int *pipe_out, int
    *pipe_err)
    ```

- Pipes werden als Datentyp `BUFF` behandelt
- in `buff.c` werden Funktionen zum Umgang mit gepufferten I/O-Zugriffen definiert
- Schreiben:
  `int ap_read(BUFF *fb, void *buf, int nbyte)` oder
  `int buff_read(BUFF *fb, void *buf, int nbyte)`
- Lesen:
  `int ap_write(BUFF *fb, const void *buf, int nbyte)` oder
  `int buff_write(BUFF *fb, const void *buf, int nbyte)`

Zugriff auf Environment-Variablen mit Standard-C-Funktionen

- Einbindung von `stdlib.h`
- die komplette Umgebung ist verfügbar mittels:
  `char** environ; // Microsoft: _environ`
- Auslesen einer Variablen:
  `char* getenv( const char *varname );`
  liefert Zeiger auf Wert der Variablen zurück, wenn gefunden
  sonst `NULL`
  Beispiel: `cout << getenv(„REQUEST_METHOD“);`
- Schreiben:
  `int putenv( const char *envstring ); // MS: _putenv`
  erzeugt oder modifiziert eine Umgebungsvariable, liefert 0 bei Erfolg zurück
  Beispiel: `putenv(„REQUEST_METHOD=GET“);`

```
// Print all environment variables available to a CGI
// output is HTML for all these browsers out there
//
// author:       Stephan Brumme
// last changes: 06/13/2001 23:58

#include <iostream>

// partially open namespace "std"
using std::cout;
using std::endl;


void main(void)
{
    // common HTTP header followed by HTML header
    cout << "Content-type: text/html" << endl << endl
         << "<html><body>" << endl;

    // print all variables by going through the _environ array
    cout << "<h1>all environment variables:</h1>" << endl;

    int nIndex = 0;
    while (_environ[nIndex])
        cout << _environ[nIndex++] << "<br>" << endl;

    // finish up HTML
    cout << "</body></html>" << endl;
}
```


http://localhost/cgi-bin/CGI Mini.exe?Hallo - Stephan
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Vorwärts Abbrechen Aktualisieren Startseite Suchen Favoriten Verlauf E-Mail
Adresse http://localhost/cgi-bin/CGI%20Mini.exe?Hallo
Links
all environment variables:
COMSPEC=C:\WINDOWS\COMMAND.COM
DOCUMENT_ROOT=c:/projects/homepage/online
HTTP_ACCEPT=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_ACCEPT_LANGUAGE=de
HTTP_CONNECTION=Keep-Alive
HTTP_HOST=localhost
HTTP_USER_AGENT=Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; MAX1.0)
PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;C:\TOOLS\;C:\PROGRAMME\TEX\MIKTEX
\BIN;C:\STUDIUM\ARBEIT\VRS3.0\LIB;C:\PROGRA~2
\RATIONAL\COMMON;C:\PROGRA~1\TCLPRO1.4\WIN32-IX86
\BIN;C:\PROGRAMME\PERL\BIN
REMOTE_ADDR=127.0.0.1
REMOTE_PORT=1447
SCRIPT_FILENAME=c:/programme/apache group/apache/cgi-bin/cgi mini.exe
SERVER_ADDR=127.0.0.1
SERVER_ADMIN=mail@stephan-brumme.com
SERVER_NAME=localhost
SERVER_PORT=80
SERVER_SIGNATURE=
Apache/1.3.19 Server at localhost Port 80
SERVER_SOFTWARE=Apache/1.3.19 (Win32) PHP/4.0.4pl1
WINDIR=C:\WINDOWS
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=Hallo

Zugriff auf die Umgebungsvariablen mit Apache

.htaccess mit `mod_env`:

- in .htaccess Umgebungsvariablen für CGI-Skripte und SSI modifizieren, Erweiterungen sind im Modul `mod_setenvif` verfügbar
- Setzen:
  `SetEnv Variable Wert`
- Löschen:
  `UnsetEnv Variable`
- explizites Durchreichen einer Variablen von der Shell-Umgebung an ein CGI:
  `PassEnv Variable`

In einem CGI-Skript über `mod_cgi`:

- Apache erzeugt eine eigene Umgebung, die nur die Variablen enthält, die für den Webserver-Betrieb notwendig sind (Prinzip der Daten-Kapselung)
- in den `request_rec` des neuen CGI-Child-Prozesses werden die Apache-spezifischen Variablen gespeichert:
```
ap_add_common_vars(request_rec *r);
ap_add_cgi_vars(*request_rec);
```
- daraus wird die gesamte Umgebung konstruiert:
```
char **env = ap_create_environment
(request_rec->pool, request_rec->subprocess_env);
```
- Definition der Funktionen zum Zugriff auf Umgebungsvariablen in util_script:
```
API_EXPORT(char **) ap_create_environment(pool *p, table *t);
API_EXPORT(void) ap_add_cgi_vars(request_rec *r);
API_EXPORT(void) ap_add_common_vars(request_rec *r);
```

- allgemeine an das CGI-Skript übergebene Variablen (`ap_add_common_vars`):
  - im Request mitgelieferte Informationen, z.B. `CONTENT_TYPE`
  - Server-spezifische Informationen, z.B. `DOCUMENT_ROOT`
  - eventuell User-Informationen, z.B. `REMOTE_USER`
  - Betriebssystem-abhängige Variablen, z.B. `WINDIR`

- CGI-bezogene Variablen (`ap_add_cgi_vars`):
  - Datenübergabe, z.B. `QUERY_STRING`
  - Skript-Eigenschaften, z.B. `SCRIPT_NAME`

mod_cgi

- Modul zum Ausführen eines CGI-Prozesses
- Konfiguration in .htaccess oder httpd.conf möglich
- Übergabe der Anfragedaten per `STDIN` oder Umgebung, Rückgabewerte in `STDOUT` erwartet
- Ausführung von CGI-Prozessen außerhalb des HTML-Verzeichnisbaumes möglich (z.B. `ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/`)
- explizite Freigabe bzw. Sperrung von Verzeichnissen erlaubt:

```
<Directory />
    Options +ExecCGI
</Directory>
```

- `mod_cgi` erzeugt eine Logdatei, die alle fehlgeschlagenen CGI-Aufrufe protokolliert
- auf jeden Fall werden festgehalten:
```
%% [time] request-line
%% HTTP-status CGI-script-filename
```
- scheiterte bereits der Aufruf des Skriptes, so findet man:
```
%% error
   error-message
```
- ansonsten:
```
% request
% response
% stdout
% stdin
```

Start eines Skriptes in `mod_cgi.c`:

```
static int cgi_child(void *child_stuff, child_info *pinfo)
{
    struct cgi_child_stuff *cld = (struct cgi_child_stuff *) child_stuff;
    request_rec *r = cld->r;
    char *argv0 = cld->argv0;
    int child_pid;

#ifdef DEBUG_CGI
#ifdef OS2
    /* Under OS/2 need to use device con. */
    FILE *dbg = fopen("con", "w");
#else
    FILE *dbg = fopen("/dev/tty", "w");
#endif
    int i;
#endif

    char **env;

    RAISE_SIGSTOP(CGI_CHILD);
#ifdef DEBUG_CGI
    fprintf(dbg, "Attempting to exec %s as %sCGI child (argv0 = %s)\n",
        r->filename, cld->nph ? "NPH " : "", argv0);
#endif

    ap_add_cgi_vars(r);
    env = ap_create_environment(r->pool, r->subprocess_env);

#ifdef DEBUG_CGI
    fprintf(dbg, "Environment: \n");
    for (i = 0; env[i]; ++i)
    fprintf(dbg, "'%s'\n", env[i]);
#endif

#ifndef WIN32
    ap_chdir_file(r->filename);
#endif
    if (!cld->debug)
    ap_error_log2stderr(r->server);

    /* Transumute outselves into the script.
     * NB only ISINDEX scripts get decoded arguments.
     */

#ifdef TPF
    return (0);
#else
    ap_cleanup_for_exec();

    child_pid = ap_call_exec(r, pinfo, argv0, env, 0);
#if defined(WIN32) || defined(OS2)
    return (child_pid);
#else

    /* Uh oh.  Still here.  Where's the kaboom?  There was supposed to be an
     * EARTH-shattering kaboom!
     *
     * Oh, well.  Muddle through as best we can...
     *
     * Note that only stderr is available at this point, so don't pass in
     * a server to aplog_error.
     */

    ap_log_error(APLOG_MARK, APLOG_ERR, NULL, "exec of %s failed", r->filename);
    exit(0);
    /* NOT REACHED */
    return (0);
#endif
#endif  /* TPF */
}
```

Quellenverzeichnis

- http://httpd.apache.org
- Rechenberg et al: *Informatik-Handbuch*, Carl Hanser Verlag München, 1999
- Bernhard Gröne: *Apache Modellierung und Code-Analyse*, Vortrag vom 14.6.2001