September 18 – 22
Sheraton Hotel – Needham, MA

Web Server Scripts in ”C”

Robert Gezelter, CSA, CSE

September 2011

ROBERT GEZELTER
SOFTWARE CONSULTANT

As a courtesy to your fellow attendees, kindly set all devices (electronic and otherwise) to the silent or vibrate mode. If you receive a call, please leave the room to answer it.

Web Performance – More with less

- Maximize number of pages delivered
- Minimize resources used
- The essence of “green computing” –
  - Minimize power – microwatts/page
  - Minimize capital – how few servers can be used
  - Maximize agility – how

Why scripting languages?

- Easy for small projects
- “nuisance” of compilation/linking
- Popularity of scripting by “non-programmers”
- Popularity of shells, PERL on *IX

None of these reasons hold for production www sites

- Content management systems (CMS)
- Active www site: millions (or more) executions per change
- “Payload is load that pays”
- Inefficiency is the opposite of a gift that keeps giving; an invoice requiring ongoing payment

"The cloud" does not resolve anything – ask Google

- Google uses enough electricity to power 200,000 homes (NY Times, 9/9/2011)
- Power "burn" directly related to number of servers
- Resource "burn" is less using more efficient (compiled) languages (e.g., "C")

```
$! TEST.COM
$!
$! Author:  Robert Gezelter 20-November-2009
$!
$! Modified by:
$!
$!
$!
$!
$       DEBUG = 0
$       WRITE SYS$OUTPUT "Content-type: text/plain"
$       WRITE SYS$OUTPUT ""
$       show symbol www*
$       WRITE SYS$OUTPUT "This is a test"
$       WRITE SYS$OUTPUT ""
$!
$ CleanUp:
$       IF  DEBUG THEN EXIT 1
$       IF F$TRNLNM("FILEHANDLE","LNM$PROCESS") .NES. "" THEN CLOSE FILEHANDLE
$!      IF F$SEARCH(FILENAME) .NES. "" THEN DELETE /NOLOG /NOCONFIRM 'FILENAME';
*
```

```
/* TESTCGI.C-Program to test CGI Programming techniques for C

   Author: Robert Gezelter      1-December-2009

   Copyright 2009, Robert Gezelter, All Rights Reserved. Not generally published.

   Distribution of this software is prohibited without permission of the  author.

   Modified by:

   */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <time.h>

#include <iodef.h>
#include <ssdef.h>
#include <descrip.h>
#include <starlet.h>
#include <lib$routines.h>
#include "cgilib.h"
```

```
/* Define common constants (for code readability */
#define YES 1
#define NO 0
typedef struct dsc$descriptor (*PageElementProcessor)
    (struct dsc$descriptor *OutputString, struct dsc$descriptor *BaseString,
        void *Parameter);

typedef struct PageElementTable_struct {
    struct dsc$descriptor descriptor;
    struct dsc$descriptor *(*ProcessorRoutine)(struct dsc$descriptor *,
        struct dsc$descriptor *, void *);
    void *ProcessingParameter;
    } PageElementTable;
char *CGIPlusEOFPtr;
```

```
    struct HTTPOutput_struct {
        unsigned long EventFlag;
        unsigned long Channel;
        unsigned long IOStatusBlock[2];};

    struct HTTPOutput_struct HTTPOutput;

/* Output formatting procedure. Note that the string produced by this
function
   is not separately allocated. If the caller wishes to preserve the output,
   external arrangements must be made, to wit, copy the resulting string to
   a separately allocated data structure. */
struct dsc$descriptor *StringOutput(struct dsc$descriptor
*OutputStringDescriptor,
    struct dsc$descriptor *BaseString, void *StringDescriptor) {
        long Result;

        Result = sys$fao(BaseString, &OutputStringDescriptor->dsc$w_length,
            OutputStringDescriptor, StringDescriptor);
        return OutputStringDescriptor;
        }
```

```
void PrintString(struct dsc$descriptor *String) {
        long Result;
#define QIO 1

#ifndef QIO
        printf("%.*s\n", (String)->dsc$w_length, (String)->dsc$a_pointer);
*/
#else
        Result = sys$qiow(HTTPOutput.EventFlag, HTTPOutput.Channel,     \
            IO$_WRITEVBLK, &HTTPOutput.IOStatusBlock[0], NULL, NULL,    \
            (String)->dsc$a_pointer, ((String)->dsc$w_length)-1,        \
            0, ' ', 0, 0);
#endif
        return;
        }
```

```
int main (){
        long Result;
        int IsCGIPlus=NO;
        int DebugFlag;

        char *RemoteAddressString;
        char *datavalue;
        char RemoteAddressValue[255];
        struct dsc$descriptor RemoteAddress;

#define PageElementInit(String, Processor, Parameter) \
        {{sizeof(String), 0, 0, (char *)(&String)}, (Processor), Parameter}

        static PageElementTable Page[] = {
         PageElementInit("Content-Type: text/plain", NULL, NULL),
         PageElementInit("Expires: Thu, 01 Jan 1970 00:00:01 GMT", NULL, NULL),
         PageElementInit("", NULL, NULL),
         PageElementInit("Request originated from: !AS", &StringOutput, NULL),
            {{0, 0, 0, NULL}, NULL, NULL}};
```

```
#define Page_OutputRequestPtr 3

        PageElementTable *LinePtr;

    char OutputStringArea[255];
    struct dsc$descriptor OutputString;

        $DESCRIPTOR(DeviceName, "SYS$OUTPUT");

#if Direct
        $DESCRIPTOR(MIME_Type, "Content-Type: text/plain;");
        $DESCRIPTOR(Blankstring, "");
        $DESCRIPTOR(MessageText, "This is a test message from a QIO");
#endif
```

```
#define QIODirect 1
#ifndef QIODirect
#define WriteString(StringDescriptor) PrintString(StringDescriptor)
#else
#define WriteString(StringDescriptor)   \
        Result = sys$qiow(HTTPOutput.EventFlag, HTTPOutput.Channel,     \
            IO$_WRITEVBLK, &HTTPOutput.IOStatusBlock[0], NULL, NULL,    \
            (StringDescriptor)->dsc$a_pointer,                          \
               ((StringDescriptor)->dsc$w_length)-1,                    \
            0, ' ', 0, 0);
#endif

        DebugFlag = NO;
        Result = lib$get_ef(&HTTPOutput.EventFlag);
        if  (getenv("TEST$DEBUG")) DebugFlag = YES;

        CgiLibEnvironmentSetDebug(DebugFlag);

        IsCGIPlus = ((CGIPlusEOFPtr = getenv("CGIPLUSEOF"))!=NULL);
```

```
Result = sys$assign(&DeviceName, &HTTPOutput.Channel, 0, 0, 0);
   do {
        OutputString.dsc$w_length = sizeof(OutputStringArea);
        OutputString.dsc$a_pointer = &OutputStringArea;
        OutputString.dsc$b_dtype = 0;
        OutputString.dsc$b_class = DSC$K_CLASS_S;
        memset(OutputStringArea, 0x00, sizeof(OutputStringArea));

        /* For debugging, get the HTTP address and URL requested */
        datavalue = NULL;
        if  (IsCGIPlus)
           {
              datavalue = getenv("WWW_REMOTE_ADDR");
              }
        else
           {
             CgiLibVar("");
             datavalue = CgiLibVar("WWW_REMOTE_ADDR");
             }
```

```
 strcpy(RemoteAddressValue, datavalue);
        RemoteAddress.dsc$w_length = strlen(RemoteAddressValue);
        RemoteAddress.dsc$a_pointer = &RemoteAddressValue[0];
        Page[Page_OutputRequestPtr].ProcessingParameter =
           (void *)(&RemoteAddress);

for (LinePtr = &Page[0]; (LinePtr->descriptor.dsc$a_pointer)!=NULL;
                    LinePtr++)
                    {
                        WriteString((LinePtr->ProcessorRoutine!=NULL) ?
                            ((LinePtr->ProcessorRoutine)(&OutputString,
                              &(LinePtr->descriptor),
                  LinePtr->ProcessingParameter)) : (&LinePtr->descriptor));
                    }
                } while (IsCGIPlus);
```

```
#if Direct
        WriteString(MIME_Type);
        WriteString(Blankstring);
        WriteString(MessageText);
#endif
        Result = sys$dassgn(HTTPOutput.Channel);
        exit(SS$_NORMAL);
        }
```

In summary --

- Web application processing on OpenVMS can be implemented using all 3rd generation languages
- Extremely high efficiency can be achieved
- Compiled languages, shareable libraries, and installed images yield extreme benefits.

Questions

Supplemental Materials, Slides:

http://www.rlgsc.com/openvms-bootcamp/2011/web-server-scripting-in-c.html