CGI servers

Philippe Martin
(philippe.martin@gu.edu.au)

Laboratory for Knowledge Visualisation and Ordering (KVO)
(http://www.int.gu.edu.au/kvo/)

Tutorials
The Common Gateway Interface
The Common Gateway Interface: FORMS
Mosaic for X version 2.0 Fill-Out Form Support
The Basics

FAQs, newsgroup, and problems
CGI Programming OpenFAQ
CGI Security: Writing Secure CGI Programs
The WWW Security FAQ

Script prototypes
C Script storing its GET/POST parameters into an associative list
Shell script using a C program to decode the GET/POST parameters
Interface to test the previous scripts
Decoding function used in the first script
Directory for these scripts
Library to access files on the Net

Script repositories
CGI scripts examples
ScriptSearch (CGIs and other kinds of scripts in many languages)

Miscellaneous stuff related to the CGI
FastCGI
NCSA httpd Installation
An alternative to CGI: incorporating a Web server in the application
A counter

```
#!/bin/sh
echo "Content-Type: text/html\n"

echo "<html>"
echo "<head><title>Result of CGI script</title>"
echo "</head>\n"
echo "<body bgcolor='white'>"

./echoGETorPOSTparam | while read NAME VALUE
do
     echo "<dt>${NAME}<dd>${VALUE}"
done

echo "</dl>\n</body>\n</html>"
```

```
int main ()
{ int c,c1,c2, i;
  if (strEqual(getenv("REQUEST_METHOD"),"POST"))
  { if (!strEqual(getenv("CONTENT_TYPE"),
                  "application/x-www-form-urlencoded"))
    { fprintf(stderr,"No form to decode\n"); return 1; }

    for (i=0; (c=getc(stdin))!=EOF; i++)
    { if (c=='+') putchar(' ');
      else if (c=='%') { c1=getc(stdin);
                         putchar(unescapeChar(c1,getc(stdin))); }
      else if (c=='=') putchar(' ');
      else if (c=='&') { putchar('\n'); i=-1; }
      else putchar(c);
    }
    if (i>0) putchar('\n');
  }
```

```
else if (strEqual(getenv("REQUEST_METHOD"),"GET"))
{ char *query= getenv("QUERY_STRING");
  if (!query || !*query)
  { fprintf(stderr,"No query to process\n"); return 0; }

  for (i=0; ((c = *query)); i++, query++)
  { if (c=='+') putchar(' ');
    else if (c=='%') { c1= *(++query); c2=*(++query);
                       putchar(unescapeURLchar(c1,c2));
                     }
    else if (c=='=') putchar(' ');
    else if (c=='&') { putchar('\n'); i=-1; }
    else putchar(c);
  }
  if (i>0) putchar('\n');
}
```

```
  else
  { fprintf(stderr,"This script should be referenced with "
                   "a method of GET or POST.\nIf you don't "
                   "understand this, see "
                   "<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html\">"
                       "this form overview</A>\n");
    return 1;
  }
  return 0;
}
```

```
char unescapeChar (char c1, char c2)
{ char c = (c1 >= 'A') ? ((c1 & 0xdf) - 'A')+10 : (c1 - '0');
       c*= 16;
       c+= (c2 >= 'A') ? ((c2 & 0xdf) - 'A')+10 : (c2 - '0');
  return c;
}

int strEqual (char *s1, char *s2)
{ if (!s1) return (s2==NULL);
  return !strcmp(s1,s2);
}
```

```
/******************* printParameters.c *************************
  Webserver log files on Hal are in /usr/ns-home/httpd-hal/logs/
 **************************************************************/
#include "handyFcts.h"
#define TRACE_FILE "/var/tmp/myTraces"
extern FILE *FpTrace;

static BOOL AllTheWorkIsDone=NO;
#include <signal.h>
void cleanExit (int sig)
{ if (!AllTheWorkIsDone) pr("Bug!\n</body>\n</html>\n");
  if (FpTrace) fclose(FpTrace);  exit(sig);
}
int FCGI_Accept ()  /* to remove if FastCGI is installed */
{ static int nbCalls=1; nbCalls--; return nbCalls;
}
extern char **TheEnvp;
HTAssocList *Parameters=NULL; /* GET/POST parameters */
```

```
int main (int argc, char *argv[], char *envp[])
          /* argv is empty since this is a CGI script */
{ char *strErr, buffForTests[250];
  int err=0;  errno=0;  TheEnvp=envp;
  /*signal on 2,3,4,10,11,15 (cf. /usr/include/signal.h)*/
    signal(SIGINT,cleanExit);  signal(SIGQUIT,cleanExit);
    signal(SIGILL,cleanExit);  signal(SIGBUS,cleanExit);
    signal(SIGSEGV,cleanExit); signal(SIGTERM,cleanExit);

  while (FCGI_Accept() >=0)
  { pr("CONTENT-TYPE: text/html\n\n");
    pr("<html>\n<head><title>CGI results</title><head>\n");
    pr("\n<body bgcolor='white'>\n");  fflush(stdout);
    ...
  }
  AllTheWorkIsDone=YES; /* to escape bug of Web server */
  return err;
}
```

```
Parameters= HTAssocList_new();
if ((strErr= initListParamWithCGIparam(initP Parameters)))
{ if (!strncmp(strErr,"This script ",11))
  { pr("Enter command(s):\n");
    fgets(buffForTests,245,stdin);     FpTrace=stderr;
  }
  else { pr(strErr); continue; }
}
else { unlink(TRACE_FILE);
       chmod(TRACE_FILE, S_IRWXU | S_IRWXG | S_IRWXO);
       if (!(FpTrace= freopen(TRACE_FILE,"w",stderr)))
       { pr("Cannot redirect stderr into " TRACE_FILE "\n");
          continue;
       }
       printParameters();
     }
pr("\n</body>\n</html>\n"); HTAssocList_delete(Parameters);
if (FpTrace) fclose(FpTrace);
```

```
void printParameters()
{ HTAssocList *paramList= Parameters;  HTAssoc *param;

  pr("\n<DL>\n");
  while ((param= HTAssocList_nextObject(paramList)))
  { pr("<DT>%s:\n",param->name);
    pr("    <DD>%s\n",param->value);
  }
  pr("</DL>\n");
}
```

```
ErrMess initListParamWithCGIparam (initP HTAssocList *paramList)
{ int c,c1,c2, i, allocSize;   char *s, *name, *value;
  if (!paramList) return "Bug! No parameter list to initialize";
  if (strEqual(getenv("REQUEST_METHOD"),"POST"))
  { int l=atoi(getenv("CONTENT_LENGTH"));
    if (l==0) return "Bug! CONTENT_LENGTH==0";
    else allocSize= (l>ALLOC_SIZE) ? ALLOC_SIZE : l;
    ...
  }
  else if (strEqual(getenv("REQUEST_METHOD"),"GET"))
  { char *query= getenv("QUERY_STRING"); allocSize=ALLOC_SIZE;
    if (!query || !*query) return "No query to process.";
    ...
  }
  else return "This script should be referenced with ...''
  MY_FREE(s);  return NULL;
}
```

```
s= (char*) my_calloc(1,sizeof(char)*(allocSize+1));
if (!s) return "Cannot store the parameters";
for (i=0,name=s,value="";(c=getc(stdin))!=EOF && (l>0); i++,l--)
{ if (i==allocSize) ... /* reallocation of s */
  if (c=='+') s[i]=' ';
  else if (c=='%') {c1=getc(stdin);
                    s[i]=unescapeChar(c1,getc(stdin)); l-=2;}
  else if (c=='=') {s[i]='\0'; value= &s[i+1];}
  else if (c=='&')
  { s[i]='\0'; if (!HTAssocList_addObject(paramList,name,value))
                  return "Error in storing the parameters";
    i= -1;  name=s;  value="";
  }else s[i]=c;
}
if (i>0) /* => s[i-1]!='&' */
{ s[i]='\0'; if (!HTAssocList_addObject(paramList,name,value))
              return "Error in storing the parameters";
}
```

```
s= (char*) my_calloc(1,sizeof(char)*(allocSize+1));
if (!s) return "Cannot store the parameters";
for (i=0, name=s, value=""; ((c = *query)); i++, query++)
{ if (i==allocSize) ... /* reallocation of s */
  if (c=='+')  s[i]=' ';
  else if (c=='%') { c1= *(++query); c2= *(++query);
                     s[i]=unescapeChar(c1,c2); }
  else if (c=='=') { s[i]='\0'; value= &s[i+1]; }
  else if (c=='&')
  { s[i]='\0'; if (!HTAssocList_addObject(paramList,name,value))
                 return "Error in storing the parameters";
    i=-1;  name=s;  value="";
  }else s[i]=c;
}
if (i>0) /* && s[i-1]!='&' */
{ s[i]='\0'; if (!HTAssocList_addObject(paramList,name,value))
               return "Error in storing the parameters";
}
```

Some of the functions implemented in handyFcts.c:

```
HTAssoc *HTAssocList_findAssoc (HTAssocList *list, char *name);
char *my_HTAssocList_findObject (HTAssocList *list, char *name);
BOOL my_HTAssocList_replaceObject (HTAssocList *list,char *name,
                                   char *value);
BOOL my_HTAssocList_removeObject (HTAssocList *list, char *name);
BOOL HTAssocList_appendObject (HTAssocList *list, char *name,
                               char *value);
int   my_HTArray_addObject (HTArray *array, void *object);
void *my_HTArray_getObject (HTArray *array, int pos);

char *loadURLtoString(char *url);
BOOL loadURLtoFile(char *url,char *fileName);
int loadURLsToTmpfiles (int nbUrls, char *urlArray[],
                      varP HTAssocList *assocUrlsToTmpFileNames);
```

Javascript project

Due on Wed. 20/10.

1) Complete the syntaxic parsing of FCGs:
path specifications and aliases.

2) For extra points, implement printSpecsOfType()
and specsOfFCG() (this requires the refinement of
the construction of objects during the parsing).

Tests your work on the assertions and queries inside kb.html.
Invent other tests and add them to your kb.html.
(Comment out some of my tests to debug;
don't forget about cache problems).