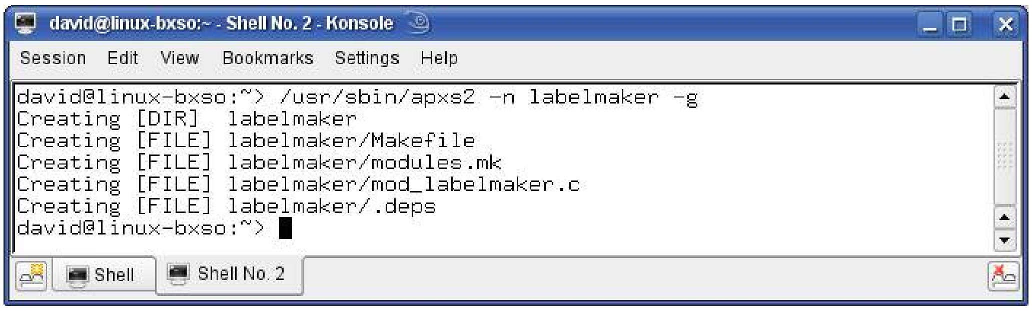
7 Apache C modules

7.1 Safer C web applications

In real life few web administrators would dream of letting anyone run C programs as CGI content generators because of the risk of crashes and core dumps. However the Apache server is itself written in C and there are simple utilities that come with its development tools that permit you to create code stubs into which you can place your C programs and run them as Apache modules when they are loaded as part of the server and managed safely in a kind of "sand-box". Here we will take an earlier example and turn it into an Apache module.

A utility called apxs2 is included in the Apache2 development libraries which can be invoked to generate a code stub for a program which can be compiled into a module that is loaded and managed by the Apache web server. These modules can be used to perform a huge variety of tasks but in our case we will do something which is akin the an ISAPI DLL found in the IIS server. The exact location of the apxs2 utility will change according to the Linux distribution you are using but with OpenSuse it runs like this.

In a terminal type: **apxs2 -n labelmaker -g**


david@linux-bxso:~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2

```
david@linux-bxso:~> /usr/sbin/apxs2 -n labelmaker -g
Creating [DIR]  labelmaker
Creating [FILE] labelmaker/Makefile
Creating [FILE] labelmaker/modules.mk
Creating [FILE] labelmaker/mod_labelmaker.c
Creating [FILE] labelmaker/.deps
david@linux-bxso:~>
```

This creates a folder of the name you give it (labelmaker) and a **Makefile**, a **modules.mk** file which can be used by the Make utility, and a file called **mod_labelmaker.c**.

The C file generated is kind of like a Hello World for Apache. It may look like a complex thing but it does supply a long explanatory comment header which is worth reading. The idea is that when Apache starts any modules in a specified location which are configured as needing to be loaded in the server configuration files, will be loaded. The *_register_hooks function lists the names and signatures of functions that can be called at specific stages in the Apache server process. In this case if the name http://localhost/labelmaker is called this module will be asked to handle whatever happens in the *_handler function.

The configuration of the server can be a bit fiddly but in OpenSuse we have to add this to the file

/etc/apache2/sites-available/default

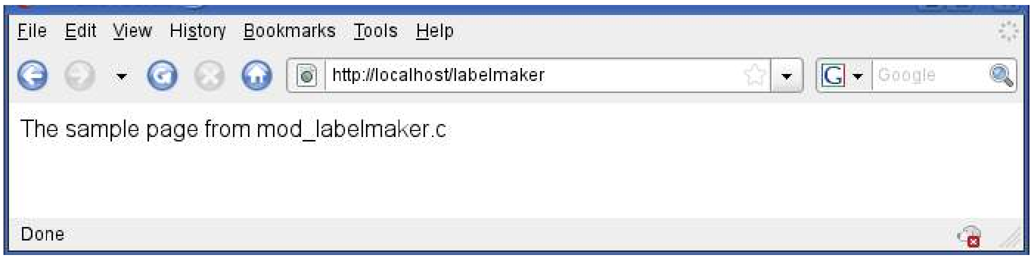
```
<Location /labelmaker>
        SetHandler labelmaker
</Location>
```

and in **/etc/config.sys/apache2** we **add** the name of our module labelmaker to long comma-separated list in the line starting

APACHE_MODULES=".....,labelmaker"
Now go to the folder labelmaker and type:

```
sudo apxs2 -c -i mod_labelmaker.c
sudo /etc/init.d/apache2 restart
```

Call this in a browser like this:


File Edit View History Bookmarks Tools Help
http://localhost/labelmaker
Google
Done

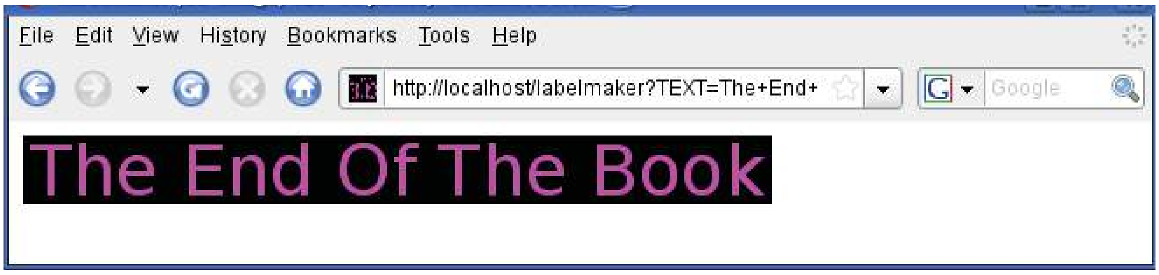
The sample page from mod_labelmaker.c

```
#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"

/* The sample content handler */
static int labelmaker_handler(request_rec *r)
{
   if (strcmp(r->handler, "labelmaker")) {
      return DECLINED;
   }
   r->content_type = "text/html";

   if (!r->header_only)
      ap_rputs("The sample page from mod_labelmaker.c\n", r);
   return OK;
}

static void labelmaker_register_hooks(apr_pool_t *p)
{
   ap_hook_handler(labelmaker_handler, NULL, NULL, APR_HOOK_MIDDLE);
}

/* Dispatch list for API hooks */
module AP_MODULE_DECLARE_DATA labelmaker_module = {
   STANDARD20_MODULE_STUFF,
   NULL,                  /* create per-dir    config structures */
   NULL,                  /* merge  per-dir    config structures */
   NULL,                  /* create per-server config structures */
   NULL,                  /* merge  per-server config structures */
   NULL,                  /* table of config file commands       */
   labelmaker_register_hooks  /* register hooks                  */
};
```

7.2 Adding some functionality

Now we can plug in the work we did for the graphics library in Chapter 6 as a replacement handler function (in the code Chapter7_1.c there are BOTH handlers, one commented out). Note the (highlighted) call to a modified decode_value function that uses the r->args pointer to get the QUERY_STRING rather than getenv(). Also Apache handles the output a bit differently too – get get a pointer to the array of bytes in the image by calling gdImageGifPtr then the ap_rwrite function outputs the data. We have to free the pointer with gdFree after the output call.

```
static int labelmaker_handler(request_rec *r)
{
        void    *iptr;
        int sz = 0;

    if (strcmp(r->handler, "labelmaker")) {
        return DECLINED;
    }
                r->content_type = "Content-type: image/gif";
    if (!r->header_only){
        int text=0,background=0, x=0,y=0,size=30,string_rectangle[8];
        double angle=0.0;
        char value[255] = "Hello";
        char font[256] = "/usr/share/fonts/truetype/DejaVuSans.ttf";
        char *err = NULL;
        gdImagePtr im_out = NULL;
        decode_value(r,"TEXT=", (char *) &value, 255);
        err=gdImageStringFT(NULL,&string_rectangle[0],0,
                font,size,angle,0,0,value);
        x = string_rectangle[2]-string_rectangle[6] + 6;
        y = string_rectangle[3]-string_rectangle[7] + 6;
        im_out = gdImageCreate(x,y);
        background = gdImageColorAllocate(im_out, 0,0,0);
        text  = gdImageColorAllocate(im_out,255,0,255);
        x = 3 - string_rectangle[6];
        y = 3 - string_rectangle[7];
        err = gdImageStringFT(im_out,&string_rectangle[0],text,
                font,size,angle,x,y,value);
        iptr = gdImageGifPtr(im_out,&sz);
        ap_rwrite(iptr, sz, r);
        gdFree(iptr);
        gdImageDestroy(im_out);
    }
    return OK;
}
```


File Edit View History Bookmarks Tools Help
http://localhost/labelmaker?TEXT=The+End+
Google

The End Of The Book

7.3 Apache Modules Conclusion

Whilst tricky to write and debug, this is probably the most rewarding and esoteric area where you can do real, commerically useful and safely deployable web content generation. It is easy to see how this example could be extended with parameters for colours and fonts to make a useful web content tool.

There is very little clear simple material about apache modules but start with the on-line documentation at http://httpd.apache.org/docs/2.2/developer/

One recent book worth looking at is “The Apache Modules Book” Nick Kew, Prentice Hall.