

qgx2009

qgx2009.blog.chinaunix.net

首页 | 博文目录 | 关于我



qgx2009

博客访问: 1317173

博文数量: 389

博客积分: 2874

博客等级: 少校

技术积分: 3077

用户组: 普通用户

注册时间: 2009-10-24 10:34

加关注 私信

论坛 加好友

文章标签

查看博文(389)

perl(1)

网络(1)

google android(1)

AI(2)

tarantool(2)

区块链(1)

lua(2)

strongswan(0)

openssl(1)

ceph(6)

xen(1)

docker(1)

python(14)

SDN(2)

HA(4)

解密(1)

音乐(1)

music(1)

技术支持(6)

kvm-实验(12)

虚拟化(42)

精英(2)

设计(1)

sql(2)

资源(18)

脚本语言(27)

linux(53)

android(2)

vm(0)

kernel(0)

堡垒(36)

cmml(0)

网络(40)

运维(20)

arm(11)

qt(10)

c/c++(36)

查看我的博文(11)

2020年(2)

2018年(39)

2017年(27)

2016年(13)

2015年(16)

2014年(92)

2013年(163)

2011年(164)

我的博文

Beau Ja

danuhai

michael7

thomas4

messau

小尾巴也

我的博文

飞鸟13

mfo426

zhangsl

安江雨_2

jefferso

conekal

admin_1

lc954752

windflow

我的博文

linux安装mysql 5.7

以后我都不再写SQL了？！

linux内核两篇中断处理...

Linux中几种中断信号分别有什么...

Bash脚本中的引号处理...

我的博文

RK3568 Buildroot 移植中文解...

全志T3开发板Q32移植试用解...

将树莓派成为大众集团长期合...

实现ARM-RCS(机器人操作系统)...

RK3568 超微处理MobaXterm与X...

将树莓派串口自动识别项目自...

2022年“维护技术交流日”...

成功适配NVIDIA DRIVE ORIN...

源利编译 | RK3568 开发板使...

如何在Ubuntu移植文本系统包...

gmssl C语言制作证书

标签

分类: C/C++

2018-07-04 15:40:32

重点是mkcert函数

点我直达源码或打赏

```
1. #include <stdio.h>
2. #include <openssl/ssl.h>
3. #include <openssl/err.h>
4. #include <openssl/conf.h>
5. #include <openssl/x509.h>
6. #include <openssl/x509v3.h>
7.
8. #include <openssl/bn.h>
9. #include <openssl/evp.h>
10. #include <openssl/asn1.h>
11. #include <openssl/rsa.h>
12. #include <openssl/obj_mac.h>
13. #include <openssl/buffer.h>
14.
15. #ifndef OPENSSL_NO_ENGINE
16. #include <openssl/engine.h>
17. #endif
18.
19. void die_most_horribly_from_openssl_error (const char *func)
20. {
21.     /* This is the OpenSSL function that prints the contents of the
22.      * error state to the specified file handle. */
23.     BIO *mem = BIO_new(BIO_s_mem());
24.     ERR_print_errors(mem);
25.
26.     BUF_MEM *pbuf;
27.     BIO_get_mem_ptr(mem, &pbuf);
28.     printf("%s ERROR %s", func, pbuf->data);
29.
30.     BIO_free(mem);
31.     exit (EXIT_FAILURE);
32. }
33.
34. static void callback(int p, int n, void *arg)
35. {
36.     char c = '0';
37.
38.     if (p == 0) c = '.';
39.     if (p == 1) c = '+';
40.     if (p == 2) c = '-';
41.     if (p == 3) c = '\0';
42.     fputc(c, stderr);
43. }
44.
45. /* pkey: cert's public key
46.  * sign_privkey: sign private key
47.  * issuer_name: issuer name, use for cert chain, if self sign, this is null, else ca's name
48.  * cn : cert's subject cn entry
49.  * days:
50.  */
51. int mkcert(X509 **x509, EVP_PKEY **pkey, EVP_PKEY **sign_privkey, X509_NAME *issuer_name, const char *cn, int bits, int serial, int days)
52. {
53.     X509 *x;
54.     EVP_PKEY *pk;
55.     X509_NAME *name = NULL;
56.
57.     if ((pkey == NULL) || (sign_privkey == NULL))
58.     {
59.         if ((pk = EVP_PKEY_new()) == NULL)
60.         {
61.             abort();
62.             return(0);
63.         }
64.     }
65.     else
66.     {
67.         pk = *pkey;
68.     }
69.     if ((x509 == NULL) || (*x509 == NULL))
70.     {
71.         if ((x = X509_new()) == NULL)
72.             goto err;
73.     }
74.     else
75.         x = *x509;
76.
77.     X509_set_version(x, 2);
78.     ASN1_INTEGER_set(&x->serialNumber, serial);
79.     X509_gmtime_adj(&x->notBefore, 0);
80.     X509_gmtime_adj(&x->notAfter, (long)60 * 24 * days);
81.     X509_set_pubkey(x, pk);
82.
83.     name = X509_get_subject_name(x);
84.
85.     /* This function creates and adds the entry, working out the
86.      * correct string type and performing checks on its length.
87.      * Normally we'd check the return value for errors...
88.      */
89.     X509_NAME_add_entry_by_txt(name, "CN",
90.                               MBSTRING_ASC, (const unsigned char *)cn, -1, -1, 0);
91.     X509_NAME_add_entry_by_txt(name, "CN",
92.                               MBSTRING_ASC, (const unsigned char *)cn, -1, -1, 0);
93.
94.     /* Its self signed so set the issuer name to be the same as the
95.      * subject.
96.      */
97.     if (issuer_name)
98.         X509_set_issuer_name(x, issuer_name);
99.     else
100.         X509_set_issuer_name(x, issuer_name);
101.
102.     /* Add various extensions: standard extensions */
103.     add_ext(x, NID_basic_constraints, "critical,CA:true");
104.     add_ext(x, NID_key_usage, "critical,keyCertSign,cRLSign");
105.     add_ext(x, NID_subject_key_identifier, "hash");
106.
107.     /* Some Netscape specific extensions */
108.     add_ext(x, NID_netscape_cert_type, "sslCA");
109.     add_ext(x, NID_netscape_comment, "example comment extension");
110.
111. #ifdef CUSTOM_EXT
112.     /* Maybe even add our own extension based on existing */
113.     {
114.         int nid;
115.         nid = OBJ_create("1.2.3.4", "MyAlias", "My Test Alias Extension");
116.         X509v3_EXT_add_alias(nid, NID_netscape_comment);
117.         add_ext(x, nid, "example comment alias");
118.     }
119. #endif
120.
121.     if (X509_sign(x, *sign_privkey, EVP_sha1()))
122.         goto err;
123.
124.     *x509 = x;
125.     *pkey = pk;
126.     return(1);
127. err:
128.     return(0);
129. }
130.
131. /* Add extension using V3 code: we can set the config file as NULL
132.  * because we want reference to other sections.
133.  */
134. int add_ext(X509 *cert, int nid, char *value)
135. {
136.     X509_EXTENSION *ex;
137.     X509v3_CTX ctx;
138.     /* This sets the "context" of the extensions. */
139.     /* No configuration database */
140.     X509v3_set_ctx_nodb(&ctx);
141.     /* Issuer and subject cert: both the target since it is self signed,
142.      * no request and no CRL
143.      */
144.     X509v3_set_ctx(&ctx, cert, cert, NULL, NULL, 0);
145.     ex = X509v3_EXT_conf_nid(NULL, &ctx, nid, value);
146.     if (ex)
147.         return 1;
148.
149.     X509_EXTENSION_free(ex);
150.     return 0;
151. }
152.
153. int main()
154. {
155.     OpenSSL_add_all_ciphers();
156.     OpenSSL_add_all_digests();
157.     ERR_load_crypto_strings();
158.     SSL_CTX *ctx = SSL_CTX_new (SSLv23_server_method ());
159.     SSL_CTX_set_options (ctx,
160.                          SSL_OP_SINGLE_DH_USE |
161.                          SSL_OP_SINGLE_ECDSA_USE |
162.                          SSL_OP_NO_SSLV2);
163.
164.     /* Cheesily pick an elliptic curve to use with elliptic curve ciphersuites.
165.      * We just hardcode a single curve which is reasonably decent.
166.      * See http://www.mail-archive.com/openssl-dev@openssl.org/msg8957.html */
167.     EC_KEY *ecdh = EC_KEY_new_by_curve_name (NID_x25519);
168.     if (!ecdh)
169.         die_most_horribly_from_openssl_error ("EC_KEY_new_by_curve_name");
170.
171.     PEM_write_bio_ECPrivateKey(&ecdh, EC_KEY *ecdh, const EVP_CIPHER *enc,
172.                               unsigned char *kstr, int klen, pem_password_cb *cb, void *u);
173.     /* ec_key_generate_key(ecdh);
174.      * EC_KEY_generate_key(ecdh);
175.      */
176.     BIO *bio_out = BIO_new_file("DN2_private.pem", "w");
177.     PEM_write_bio_ECPrivateKey(bio_out, ecdh, NULL, NULL, 0, NULL, NULL);
178.
179.     BIO_free(bio_out);
180.
181.     //-----make self sign cert-----
182.     EVP_PKEY *evk = EVP_PKEY_new();
183.     EVP_PKEY_set1_EC_KEY(evk, ecdh);
184.     X509 *x509 = NULL;
185.
186.     mkcert(&x509, &evk, &ecdh, "openssl Group", 256, 2345, 3650);
187.     BIO *bio_x = BIO_new_file("cert.pem", "w");
188.     PEM_write_bio_X509(bio_x, x509);
189.
190.     BIO_free(bio_x);
191.
192.     EVP_PKEY_free(evk);
193.     //-----make server cert -----
194.     evk = EVP_PKEY_new(); //ca key
195.     EVP_PKEY_set1_EC_KEY(evk, ecdh);
196.
197.     EC_KEY *ec_server = EC_KEY_new_by_curve_name (NID_x25519);
198.     if (!ec_server)
199.         die_most_horribly_from_openssl_error ("EC_KEY_new_by_curve_name");
200. }
```

```
208.     EC_KEY_generate_key(&c_server);
209.     bio_out = BIO_new_file("server.pem", "w");
210.
211.
212.     PEM_write_bio_ECPrivateKey(bio_out, &c_server, NULL, NULL, 0, NULL, NULL);
213.     BIO_free(bio_out);
214.
215.     //
216.     EVP_PKEY *evserv = EVP_PKEY_new();
217.     EVP_PKEY_set1_EC_KEY(evserv, &c_server);
218.
219.     X509 *serv_x509 = NULL;
220.     X509_NAME *is_name = X509_get_subject_name(x509);
221.
222.     mcert(&serv_x509, &evserv, &evk, is_name, "gd.com", 256, 2345, 3650);
223.     BIO *bio_s = BIO_new_file("server-cert.pem", "w");
224.     PEM_write_bio_X509(bio_s, serv_x509);
225.
226.     BIO_free(bio_s);
227.     EVP_PKEY_free(evserv);
228.     EC_KEY_free(&c_server);
229.     EVP_PKEY_free(evk);
230.     //-----
231.
232.     EC_KEY_free(&cdh);
233.     SSL_CTX_free(&ctx);
234.     return 0;
235. }
```

参考github上的代码。



阅读(3147) | 评论(0) | 转发(0) |

上一篇: C语言中的宏
下一篇: C语言链表(kipilist)实现



给主人留下点什么吧！~