风格的错误处理。例如，Posix风格的pthread_create函数用它的返回值来表明成功或者失败，而通过引用将新创建的线程的ID(有用的结果)返回放在它的第一个参数中。Posix风格的错误处理代码通常具有以下形式：

```
1    if ((retcode = pthread_create(&tid, NULL, thread, NULL)) != 0) {
2        fprintf(stderr, "pthread_create error: %s\n", strerror(retcode));
3        exit(0);
4    }
```

strerror函数返回retcode某个值对应的文本描述。

3. GAI风格的错误处理

getaddrinfo(GAI)和getnameinfo函数成功时返回零，失败时返回非零值。GAI错误处理代码通常具有以下形式：

```
1    if ((retcode = getaddrinfo(host, service, &hints, &result)) != 0) {
2        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(retcode));
3        exit(0);
4    }
```

gai_strerror函数返回retcode某个值对应的文本描述。

4. 错误报告函数小结

贯穿本书，我们使用下列错误报告函数来包容不同的错误处理风格：

```
#include "csapp.h"

void unix_error(char *msg);
void posix_error(int code, char *msg);
void gai_error(int code, char *msg);
void app_error(char *msg);
```

返回：无。

正如它们的名字表明的那样，unix_error、posix_error和gai_error函数报告Unix风格的错误、Posix风格的错误和GAI风格的错误，然后终止。包括app_error函数是为了方便报告应用错误。它只是简单地打印它的输入，然后终止。图A-1展示了这些错误报告函数的代码。

code/src/csapp.c

```
1   void unix_error(char *msg) /* Unix-style error */
2   {
3       fprintf(stderr, "%s: %s\n", msg, strerror(errno));
4       exit(0);
5   }
6
7   void posix_error(int code, char *msg) /* Posix-style error */
8   {
9       fprintf(stderr, "%s: %s\n", msg, strerror(code));
10      exit(0);
11  }
12
13  void gai_error(int code, char *msg) /* Getaddrinfo-style error */
```

图A-1 错误报告函数