

linux系統編程之錯誤處理：perror, strerror和errno

1，在系統編程中錯誤通常通過函數返回值來表示，並通過特殊變量errno來描述。

errno這個全局變量在<errno.h>頭文件中聲明如下：extern int errno;

errno是一個由POSIX和ISO C標準定義的符號，看（用）起來就好像是一個整形變量。當系統調用或庫函數發生錯誤的時候，比如以只讀方式打開一個不存在的文件時，它的值將會被改變，根據errno值的不同，我們就可以知道我們的程序發生了什麼錯誤，然後進行相應的處理。

為什麼，要強調errno看起來好像是一個整形變量呢？因為有的標準(如ISO C)只規定了errno的作用，而沒有規定它的實現方式，它可能被定義成一個變量，也有可能被定義成一個宏，這個具體要看編譯器自己的實現。早些時候，POSIX.1曾把errno定義成extern int errno這種形式，但現在這種方式比較少見了。因為以這種形式來實現errno，在多線程環境下errno變量是被多個線程共享的，這樣可能線程A發生某些錯誤改變了errno的值，線程B雖然沒有發生任何錯誤，但是當它檢測errno的值的時候，線程B會以為自己發生了錯誤。所以現在errno在Linux中被實現成extern int * __errno_location(void); #define errno (*__errno_location())，這樣每個線程都有自己的errno，不會再發生混亂了。

關於errno有三點需要特別注意：

- 1、如果系統調用或庫函數正確執行的話，errno的值是不會被清零（置0，注意這裡是不會被清零，不是不會被改變）的，假若執行函數A的時候發生了錯誤errno被改變，接下來直接執行函數B，如果函數B正確執行的話，errno還保留函數A發生錯誤時被設置的值。所以，在利用errno之前，最好先對函數的返回值進行判斷，看是否發生了錯誤，返回值錯誤再利用errno判斷時哪裡發生了錯誤。所以如果一個函數無法從返回值上判斷正誤，而只能通過errno來判斷出錯，那你在調用它之前必須手動將**errno清零**！
- 2、系統調用或庫函數正確執行，並不保證errno的值不會被改變！
- 3、任何錯誤號（即發生錯誤時errno的取值）都是非0的。

綜上所述，當需要用errno來判斷函數是否正確執行的時候，最好先將errno清零，函數執行結束時，通過其返回值判斷函數是否正確執行，若沒有正確執行，再根據errno判斷時哪裡發生了錯誤。

2. 錯誤處理函數

- perror
- strerror


perror和strerror函數都是用來打印錯誤提示信息的，它們的原型分別是：

```
#include <stdio.h>
```

```
void perror(const char *s);
```

它先打印s指向的字符串，然後輸出當前errno值所對應的錯誤提示信息，例如當前errno若為12，調用perror("ABC")，會輸出"ABC: Cannot allocate memory"。

測試程序：

```
  
#include <stdio.h>  
#include <unistd.h>  
  
int main( void )
```

```
{  
    int fd = 10 ;  
    int ret;  
    ret = close(fd);  
    if (ret == - 1 )  
        perror( " close error " );  
    return 0 ;  
}
```



測試結果：

```
zhouxy@master:~/unixenv_c$ gcc perror.c  
zhouxy@master:~/unixenv_c$ ./a.out  
close error: Bad file descriptor  
zhouxy@master:~/unixenv_c$ |
```

#include <string.h>

char *strerror(int errnum);

它返回errnum的值所對應的錯誤提示信息，例如errnum等於12的話，它就會返回"Cannot allocate memory"。

測試程序：

```
#include <stdio.h>  
#include < string .h>  
#include <unistd.h>  
#include <errno.h>  
  
int main( void )  
{  
    int fd = 10 ;  
    int ret;  
    ret = close(fd);  
    if (ret == - 1 )  
        fprintf(stderr, " close error with msg is: %s\n " ,strerror(errno));  
    return 0 ;  
}
```



測試結果：

```
zhouxy@master:~/unixenv_c$ gcc strerror.c  
zhouxy@master:~/unixenv_c$ ./a.out  
close error with msg is: Bad file descriptor  
zhouxy@master:~/unixenv_c$ |
```