#include <Windows.h>

namespace COW1

{

class String

{

public:

String(const char\* str)

: \_str(new char [ strlen(str )+1])

{

strcpy(\_str , str);

\_refCount = new int(1);

}

String(const String& s)

: \_refCount(s.\_refCount)

, \_str(s.\_str)

{

// 增加引用计数

++ \_refCount[0];

}

String& operator =(const String& s)

{

if (this->\_str != s.\_str)

{

// --旧的引用计数，如果是最后一个引用对象，则释放对象

\_Release();

// 成员变量赋值，并增加引用计数

\_str = s.\_str;

\_refCount = s.\_refCount;

++ \_refCount[0];

}

return \*this ;

}

char& operator[](size\_t index)

{

// Copy On Write

if (\*\_refCount > 1)

{

--(\*\_refCount);

char\* tmpPtr = new char[strlen(\_str) + 1];

int\* tmpCount = new int(1);

strcpy(tmpPtr, \_str);

\_str = tmpPtr;

\_refCount = tmpCount;

}

return \_str[index];

}

size\_t GetSize ()

{

return strlen (\_str);

}

char\* GetStr ()

{

return \_str ;

}

void \_Release()

{

// --旧的引用计数，如果是最后一个引用对象，则释放对象

if (--\_refCount [0] == 0)

{

delete[] \_str ;

delete [] \_refCount;

}

}

~ String()

{

\_Release();

}

private:

char\* \_str ;

int\* \_refCount ; // 引用计数

};

}

namespace COW2

{

class String

{

public:

String(const char\* str)

:\_str(new char[strlen(str) + 5])

{

// 初始化引用计数

\_str += 4;

GetRefCount(\_str) = 1;

strcpy(\_str, str);

}

String(const String& s)

:\_str(s.\_str)

{

// 增加引用计数

++GetRefCount(\_str);

}

String& operator=(const String& s)

{

if (this->\_str != s.\_str)

{

Release(\_str);

\_str = s.\_str;

++GetRefCount(\_str);

}

return \*this;

}

char& operator[](size\_t index)

{

// Copy On Write

if (GetRefCount(\_str) > 1)

{

// 减引用计数

--GetRefCount(\_str);

char\* tmpPtr = new char[strlen(\_str) + 5];

tmpPtr += 4;

GetRefCount(tmpPtr) = 1;

strcpy(tmpPtr, \_str);

\_str = tmpPtr;

}

return \_str[index];

}

~String()

{

Release(\_str);

}

size\_t GetSize()

{

return strlen(\_str);

}

char\* GetStr()

{

return \_str;

}

protected:

// 获取引用计数（注意这里返回引用，且注意传入的指针是数据起点）

inline int& GetRefCount(char\* str)

{

return \*(int\*)(str-4);

}

// 引用计数减1，若是最后一个引用对象，则释放对象。

inline void Release(char\* str)

{

if ((--GetRefCount(str)) == 0)

{

delete[] (str-4);

}

}

private:

char\* \_str;

};

}

// 测试基本的拷贝

void TestString\_COW1 ()

{

COW1::String s1("hello world!");

COW1::String s2 = s1;

cout<<"s1:"<<s1.GetStr()<<endl;

cout<<"s2:"<<s2.GetStr()<<endl<<endl;

s1[0] = 'x';

cout<<"s1:"<<s1.GetStr()<<endl;

cout<<"s2:"<<s2.GetStr()<<endl<<endl;

COW1::String s3("");

s3 = s1;

cout<<"s3:"<<s3.GetStr()<<endl<<endl;

}

// 测试COW1效率

void TestString\_COW2 ()

{

COW1::String s1 = "hello world";

int begin = GetTickCount();

for (int i = 0; i < 10000; ++i)

{

COW1::String s2 = s1;

// do something

// cout<<s2.GetStr()<<endl;

}

int end = GetTickCount();

cout<<"cost time:"<<end - begin<<endl;

}

// 测试什么时候写时拷贝

void TestString\_COW3()

{

COW1::String s1("hello world!");

COW1::String s2 = s1;

COW1::String s3 = s2;

cout<<"s1:"<<(int\*)s1.GetStr()<<endl;

cout<<"s2:"<<(int\*)s2.GetStr()<<endl;

s2[0] = 'x';

cout<<"s1:"<<(int\*)s1.GetStr()<<endl;

cout<<"s2:"<<(int\*)s2.GetStr()<<endl;

// 读时也拷贝

cout<<s3[0]<<endl;

};