

在内核编程中字符串有两种格式 ANSI_STRING 与 UNICODE_STRING，这两种格式是微软推出的安全版本的字符串结构体，也是微软推荐使用的格式，通常情况下 ANSI_STRING 代表的类型是 char * 也就是 ANSI 多字节模式的字符串，而 UNICODE_STRING 则代表的是 wchar* 也就是 UNICODE 类型的字符，如下文章将介绍这两种字符格式在内核中是如何转换的。

在内核开发模式下 初始化字符串 也需要调用专用的初始化函数，如下分别初始化 ANSI 和 UNICODE 字符串，我们来看看代码是如何实现的。

```
#include <ntifs.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    // 定义内核字符串
    ANSI_STRING ansi;
    UNICODE_STRING unicode;
    UNICODE_STRING str;

    // 定义普通字符串
    char * char_string = "hello lyshark";
    wchar_t *wchar_string = (WCHAR*)"hello lyshark";

    // 初始化字符串的多种方式
    RtlInitAnsiString(&ansi, char_string);
    RtlInitUnicodeString(&unicode, wchar_string);
    RtlUnicodeStringInit(&str, L"hello lyshark");

    // 改变原始字符串（乱码位置，此处仅用于演示赋值方式）
    char_string[0] = (CHAR)"A";           // char类型每个占用1字节
    char_string[1] = (CHAR)"B";

    wchar_string[0] = (WCHAR)"A";          // wchar类型每个占用2字节
    wchar_string[2] = (WCHAR)"B";

    // 输出字符串 %Z
    DbgPrint("输出ANSI: %Z \n", &ansi);
    DbgPrint("输出WCHAR: %Z \n", &unicode);
    DbgPrint("输出字符串: %wZ \n", &str);

    DbgPrint("驱动加载成功 \n");
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

代码输出效果：

DebugView on \\DESKTOP-B53PAVI (local)		
File Edit Capture Options Computer Help		
#	Time	Debug Print
56	31.80723572	666406250 - STORMINI: StorNVMe - POWER: IDLE
57	31.80726433	666406250 - STORMINI: StorNVMe - POWER: ACTIVE
58	32.80714798	676406250 - STORMINI: StorNVMe - POWER: IDLE
59	32.80838394	676406250 - STORMINI: StorNVMe - POWER: ACTIVE
60	33.82435608	686562500 - STORMINI: StorNVMe - POWER: IDLE
61	34.51197052	693437500 - STORMINI: StorNVMe - POWER: ACTIVE
62	34.51274872	693437500 - STORMINI: StorNVMe - POWER: IDLE
63	34.51290894	693437500 - STORMINI: StorNVMe - POWER: ACTIVE
64	40.36994171	752031250 - STORMINI: StorNVMe - POWER: IDLE
65	40.37130737	752031250 - STORMINI: StorNVMe - POWER: ACTIVE
66	40.55623627	输出ANSI: d磗lo lyshark
67	40.55624008	输出WCHAR: ?11?lyshark输出字符串: hello lyshark
68	40.55624390	驱动加载成功
69	41.33798981	驱动卸载成功
70	41.37001038	762031250 - STORMINI: StorNVMe - POWER: IDLE
71	41.85396957	766875000 - STORMINI: StorNVMe - POWER: ACTIVE
72	47.71379852	825468750 - STORMINI: StorNVMe - POWER: ACTIVE

内核中还可实现字符串与整数之间的灵活转换，内核中提供了 `RtlUnicodeStringToInteger` 这个函数来实现字符串转整数，与之对应的 `RtlIntegerToUnicodeString` 则是将整数转为字符串这两个内核函数也是非常常用的。

```
#include <ntifs.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

// Power: lyshark
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    NTSTATUS flag;
    ULONG number;

    DbgPrint("hello lyshark \n");

    UNICODE_STRING uncode_buffer_source = { 0 };
    UNICODE_STRING uncode_buffer_target = { 0 };

    // 字符串转为数字
    // By: LyShark
    RtlInitUnicodeString(&uncode_buffer_source, L"100");
    flag = RtlUnicodeStringToInteger(&uncode_buffer_source, 10, &number);

    if (NT_SUCCESS(flag))
    {
        DbgPrint("字符串 -> 数字: %d \n", number);
    }

    // 数字转为字符串
    uncode_buffer_target.Buffer = (PWSTR)ExAllocatePool(PagedPool, 1024);
    uncode_buffer_target.MaximumLength = 1024;
```

```

flag = RtlIntegerToUnicodeString(number, 10, &unicode_buffer_target);

if (NT_SUCCESS(flag))
{
    DbgPrint("数字 -> 字符串: %wZ \n", &unicode_buffer_target);
}

// 释放堆空间
RtlFreeUnicodeString(&unicode_buffer_target);

DbgPrint("驱动加载成功 \n");
Driver->DriverUnload = UnDriver;
return STATUS_SUCCESS;
}

```

代码输出效果:

#	Time	Debug Print
87	773.88616943	8087187500 - STORMINI: StorNVMe - POWER: IDLE
88	773.88623047	8087187500 - STORMINI: StorNVMe - POWER: ACTIVE
89	774.88714600	8097187500 - STORMINI: StorNVMe - POWER: IDLE
90	775.15966797	8099843750 - STORMINI: StorNVMe - POWER: ACTIVE
91	776.16027832	8109843750 - STORMINI: StorNVMe - POWER: IDLE
92	776.55065918	8113750000 - STORMINI: StorNVMe - POWER: ACTIVE
93	776.89355469	hello lyshark
94	776.89355469	字符串 -> 数字: 100
95	776.89355469	数字 -> 字符串: 100
96	776.89355469	驱动加载成功
97	777.55548096	8123906250 - STORMINI: StorNVMe - POWER: IDLE
98	777.89257813	8127187500 - STORMINI: StorNVMe - POWER: ACTIVE
99	778.89685059	8137343750 - STORMINI: StorNVMe - POWER: IDLE
100	779.09533691	驱动卸载成功
101	779.88623047	8147187500 - STORMINI: StorNVMe - POWER: ACTIVE
102	779.88653564	8147187500 - STORMINI: StorNVMe - POWER: IDLE
103	780.16912842	8150000000 - STORMINI: StorNVMe - POWER: ACTIVE

继续看另一种转换模式，将 UNICODE_STRING 结构转换成 ANSI_STRING 结构，代码中调用了 `RtlUnicodeStringToAnsiString` 内核函数，该函数也是微软提供的。

```

#include <ntifs.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

// Power: lyshark
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark \n");

    UNICODE_STRING unicode_buffer_source = { 0 };
    ANSI_STRING ansi_buffer_target = { 0 };

    // 初始化 UNICODE 字符串

```

```

RtlInitUnicodeString(&unicode_buffer_source, L"hello lyshark");

// 转换函数
NTSTATUS flag = RtlUnicodeStringToAnsiString(&ansi_buffer_target,
&unicode_buffer_source, TRUE);

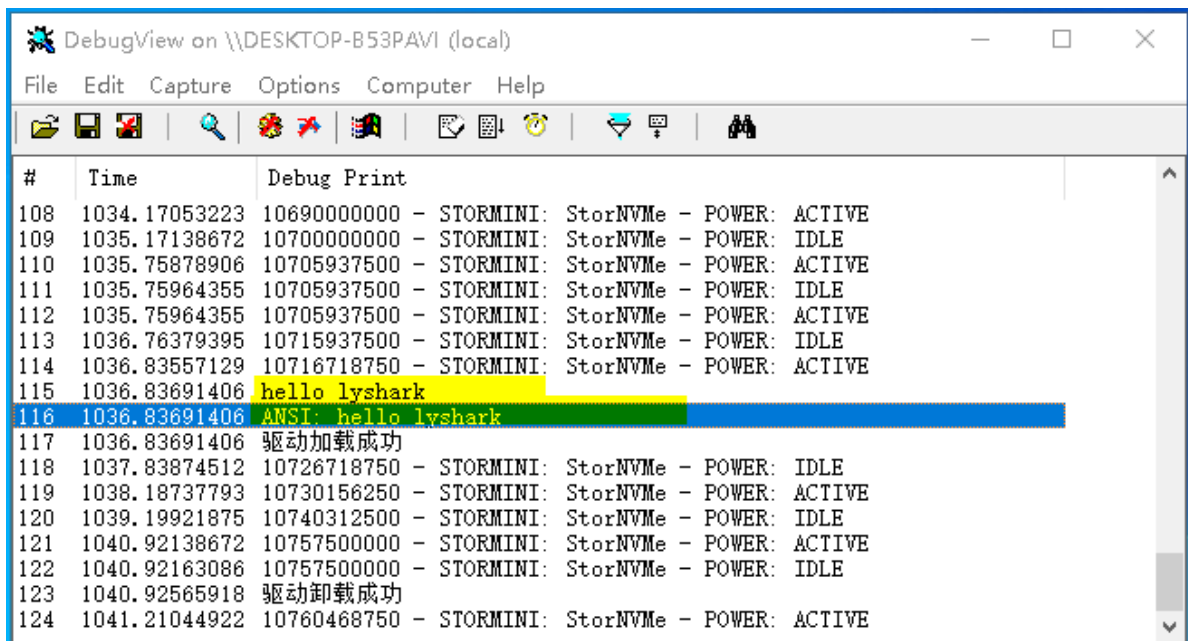
if (NT_SUCCESS(flag))
{
    DbgPrint("ANSI: %Z \n", &ansi_buffer_target);
}

// 销毁ANSI字符串
RtlFreeAnsiString(&ansi_buffer_target);

DbgPrint("驱动加载成功 \n");
Driver->DriverUnload = UnDriver;
return STATUS_SUCCESS;
}

```

代码输出效果:



DebugView on \\DESKTOP-B53PAVI (local)

#	Time	Debug Print
108	1034.17053223	10690000000 - STORMINI: StorNVMe - POWER: ACTIVE
109	1035.17138672	10700000000 - STORMINI: StorNVMe - POWER: IDLE
110	1035.75878906	10705937500 - STORMINI: StorNVMe - POWER: ACTIVE
111	1035.75964355	10705937500 - STORMINI: StorNVMe - POWER: IDLE
112	1035.75964355	10705937500 - STORMINI: StorNVMe - POWER: ACTIVE
113	1036.76379395	10715937500 - STORMINI: StorNVMe - POWER: IDLE
114	1036.83557129	10716718750 - STORMINI: StorNVMe - POWER: ACTIVE
115	1036.83691406	hello lyshark
116	1036.83691406	ANSI: hello lyshark
117	1036.83691406	驱动加载成功
118	1037.83874512	10726718750 - STORMINI: StorNVMe - POWER: IDLE
119	1038.18737793	10730156250 - STORMINI: StorNVMe - POWER: ACTIVE
120	1039.19921875	10740312500 - STORMINI: StorNVMe - POWER: IDLE
121	1040.92138672	10757500000 - STORMINI: StorNVMe - POWER: ACTIVE
122	1040.92163086	10757500000 - STORMINI: StorNVMe - POWER: IDLE
123	1040.92565918	驱动卸载成功
124	1041.21044922	10760468750 - STORMINI: StorNVMe - POWER: ACTIVE

如果将上述过程反过来，将 ANSI_STRING 转换为 UNICODE_STRING 结构，则需要调用 RtlAnsiStringToUnicodeString 这个内核专用函数实现。

```

#include <ntifs.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

// Power: lyshark
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark \n");
}

```

```

UNICODE_STRING uncode_buffer_source = { 0 };
ANSI_STRING ansi_buffer_target = { 0 };

// 初始化字符串
RtlInitString(&ansi_buffer_target, "hello lyshark");

// 转换函数
NTSTATUS flag = RtlAnsiStringToUnicodeString(&uncode_buffer_source,
&ansi_buffer_target, TRUE);
if (NT_SUCCESS(flag))
{
    DbgPrint("UNICODE: %wZ \n", &uncode_buffer_source);
}

// 销毁UNICODE字符串
RtlFreeUnicodeString(&uncode_buffer_source);

DbgPrint("驱动加载成功 \n");
Driver->DriverUnload = UnDriver;
return STATUS_SUCCESS;
}

```

代码输出效果:

#	Time	Debug Print
122	1040.92163086	10757500000 - STORMINI: StorNVMe - POWER: IDLE
123	1040.92565918	驱动卸载成功
124	1041.21044922	10760468750 - STORMINI: StorNVMe - POWER: ACTIVE
125	1183.64953613	12184843750 - STORMINI: StorNVMe - POWER: IDLE
126	1184.31970215	12191562500 - STORMINI: StorNVMe - POWER: ACTIVE
127	1184.32006836	12191562500 - STORMINI: StorNVMe - POWER: IDLE
128	1184.32043457	12191562500 - STORMINI: StorNVMe - POWER: ACTIVE
129	1184.36450195	hello lyshark
130	1184.36450195	UNICODE: hello lyshark
131	1184.36450195	驱动加载成功
132	1185.32958984	12201562500 - STORMINI: StorNVMe - POWER: IDLE
133	1185.35583496	12201875000 - STORMINI: StorNVMe - POWER: ACTIVE
134	1185.49902344	驱动卸载成功
135	1186.35632324	12211875000 - STORMINI: StorNVMe - POWER: IDLE
136	1187.65563965	12224843750 - STORMINI: StorNVMe - POWER: ACTIVE
137	1187.65625000	12224843750 - STORMINI: StorNVMe - POWER: IDLE
138	1187.65637207	12224843750 - STORMINI: StorNVMe - POWER: ACTIVE

如上代码是内核通用结构体之间的转换类型，又是还需要将各类结构体转为普通的字符类型，例如下方的两个案例：

例如将 `UNICODE_STRING` 转为 `CHAR*` 类型。

```

#define _CRT_SECURE_NO_WARNINGS
#include <ntifs.h>
#include <windef.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

```

```
// powerBY: LyShark
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark \n");

    UNICODE_STRING unicode_buffer_source = { 0 };
    ANSI_STRING ansi_buffer_target = { 0 };
    char szBuf[1024] = { 0 };

    // 初始化 UNICODE 字符串
    RtlInitUnicodeString(&unicode_buffer_source, L"hello lyshark");

    // 转换函数
    NTSTATUS flag = RtlUnicodeStringToAnsiString(&ansi_buffer_target,
&unicode_buffer_source, TRUE);

    if (NT_SUCCESS(flag))
    {
        strcpy(szBuf, ansi_buffer_target.Buffer);
        DbgPrint("输出char*字符串: %s \n", szBuf);
    }

    // 销毁ANSI字符串
    RtlFreeAnsiString(&ansi_buffer_target);

    DbgPrint("驱动加载成功 \n");
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

代码输出效果:

#	Time	Debug Print
137	1187.65625000	12224843750 - STORMINI: StorNVMe - POWER: IDLE
138	1187.65637207	12224843750 - STORMINI: StorNVMe - POWER: ACTIVE
139	1472.04418945	15068750000 - STORMINI: StorNVMe - POWER: ACTIVE
140	1472.04443359	15068750000 - STORMINI: StorNVMe - POWER: IDLE
141	1472.04455566	15068750000 - STORMINI: StorNVMe - POWER: ACTIVE
142	1472.04675293	hello lyshark
143	1472.04675293	输出char*字符串: hello lyshark
144	1472.04675293	驱动加载成功
145	1473.04760742	15078750000 - STORMINI: StorNVMe - POWER: IDLE
146	1473.38745117	15082187500 - STORMINI: StorNVMe - POWER: ACTIVE
147	1474.39929199	15092343750 - STORMINI: StorNVMe - POWER: IDLE
148	1474.50231934	15093281250 - STORMINI: StorNVMe - POWER: ACTIVE
149	1475.50756836	15103437500 - STORMINI: StorNVMe - POWER: IDLE
150	1477.35461426	驱动卸载成功
151	1477.47595215	15123125000 - STORMINI: StorNVMe - POWER: ACTIVE
152	1477.47631836	15123125000 - STORMINI: StorNVMe - POWER: IDLE
153	1477.47680664	15123125000 - STORMINI: StorNVMe - POWER: ACTIVE

如果反过来, 将 `CHAR*` 类型转为 `UNICODE_STRING` 结构呢, 可以进行中转最终转为 `UNICODE_STRING` 结构体。

```
#define _CRT_SECURE_NO_WARNINGS
```

```

#include <ntifs.h>
#include <windef.h>
#include <ntstrsafe.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("驱动卸载成功 \n");
}

// powerBY: LyShark
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark \n");

    UNICODE_STRING uncode_buffer_source = { 0 };
    ANSI_STRING ansi_buffer_target = { 0 };

    // 设置CHAR*
    char szBuf[1024] = { 0 };
    strcpy(szBuf, "hello lyshark");

    // 初始化ANSI字符串
    RtlInitString(&ansi_buffer_target, szBuf);

    // 转换函数
    NTSTATUS flag = RtlAnsiStringToUnicodeString(&uncode_buffer_source,
&ansi_buffer_target, TRUE);
    if (NT_SUCCESS(flag))
    {
        DbgPrint("UNICODE: %wZ \n", &uncode_buffer_source);
    }

    // 销毁UNICODE字符串
    RtlFreeUnicodeString(&uncode_buffer_source);

    DbgPrint("驱动加载成功 \n");
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}

```

代码输出效果：

DebugView on \\DESKTOP-B53PAVI (local)		
File Edit Capture Options Computer Help		
#	Time	Debug Print
160	1600.48962402	16353281250 - STORMINI: StorNVMe - POWER: IDLE
161	1600.84045410	16356718750 - STORMINI: StorNVMe - POWER: ACTIVE
162	1601.84057617	16366718750 - STORMINI: StorNVMe - POWER: IDLE
163	1601.94750977	驱动卸载成功
164	1602.57849121	16374062500 - STORMINI: StorNVMe - POWER: ACTIVE
165	1602.57934570	16374062500 - STORMINI: StorNVMe - POWER: IDLE
166	1604.77258301	hello lyshark
167	1604.77258301	UNICODE: hello lyshark
168	1604.77258301	驱动加载成功
169	1604.78942871	16396250000 - STORMINI: StorNVMe - POWER: ACTIVE
170	1604.78967285	16396250000 - STORMINI: StorNVMe - POWER: IDLE
171	1604.78979492	16396250000 - STORMINI: StorNVMe - POWER: ACTIVE
172	1605.79370117	16406250000 - STORMINI: StorNVMe - POWER: IDLE
173	1606.07885742	16409062500 - STORMINI: StorNVMe - POWER: ACTIVE
174	1607.05871582	驱动卸载成功
175	1607.08361816	16419218750 - STORMINI: StorNVMe - POWER: IDLE
176	1607.58032227	16424062500 - STORMINI: StorNVMe - POWER: ACTIVE