# Windows内核中是无法使用 vector 容器等数据结构的，当我们需要保存一个结构体数组时，就需要使用内核中提供的专用链表结构 LIST\_ENTRY 通过一些列链表操作函数对结构体进行装入弹出等操作，如下代码是本人总结的内核中使用链表存储多个结构体的通用案例。

首先实现一个枚举用户进程功能，将枚举到的进程存储到链表结构体内。

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



extern PVOID PsGetProcessPeb(\_In\_ PEPROCESS Process);

NTKERNELAPI NTSTATUS PsLookupProcessByProcessId(HANDLE ProcessId, PEPROCESS Process);

extern NTKERNELAPI PVOID PsGetProcessWow64Process(\_In\_ PEPROCESS Process); extern NTKERNELAPI UCHAR PsGetProcessImageFileName(IN PEPROCESS Process); extern NTKERNELAPI HANDLE PsGetProcessInheritedFromUniqueProcessId(IN PEPROCESS Process);

typedef struct

{

DWORD Pid;

UCHAR ProcessName[2048]; DWORD Handle; LIST\_ENTRY ListEntry;

}ProcessList;

// 根据进程ID返回进程EPROCESS结构体失败返回NULL PEPROCESS LookupProcess(HANDLE Pid)

{

PEPROCESS eprocess = NULL;

NTSTATUS Status = STATUS\_UNSUCCESSFUL;

Status = PsLookupProcessByProcessId(Pid, &eprocess); if (NT\_SUCCESS(Status))

{

return eprocess;

}

return NULL;

}

// 内核链表操作

// By: LyShark

BOOLEAN GetAllProcess()

{

PEPROCESS eproc = NULL;

LIST\_ENTRY linkListHead;

// 初始化链表头部

InitializeListHead(&linkListHead); ProcessList pData = NULL;

for (int temp = 0; temp < 100000; temp += 4)

{

eproc = LookupProcess((HANDLE)temp); if (eproc != NULL)

{



STRING nowProcessnameString = { 0 }; RtlInitString(&nowProcessnameString,

PsGetProcessImageFileName(eproc));

// DbgPrint("进程名: %s --> 进程PID = %d --> 父进程PPID = %d\r\n",

// PsGetProcessImageFileName(eproc), PsGetProcessId(eproc), PsGetProcessInheritedFromUniqueProcessId(eproc));

// 分配内核堆空间

pData = (ProcessList )ExAllocatePool(PagedPool, sizeof(ProcessList));

RtlZeroMemory(pData, sizeof(ProcessList));

// 设置变量

pData->Pid = (DWORD)PsGetProcessId(eproc);

RtlCopyMemory(pData->ProcessName, PsGetProcessImageFileName(eproc), strlen(PsGetProcessImageFileName(eproc)) 2);

pData->Handle =

(DWORD)PsGetProcessInheritedFromUniqueProcessId(eproc);

// 插入元素到

InsertTailList(&linkListHead, &pData->ListEntry); ObDereferenceObject(eproc);

}

}

// 输出链表内的数据

while (!IsListEmpty(&linkListHead))

{

LIST\_ENTRY pEntry = RemoveHeadList(&linkListHead);

pData = CONTAINING\_RECORD(pEntry, ProcessList, ListEntry);

DbgPrint("%d \n", pData->Pid); DbgPrint("%s \n", pData->ProcessName); DbgPrint("%d \n", pData->Handle); ExFreePool(pData);

}

return TRUE;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

DbgPrint(("Uninstall Driver Is OK \n"));

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

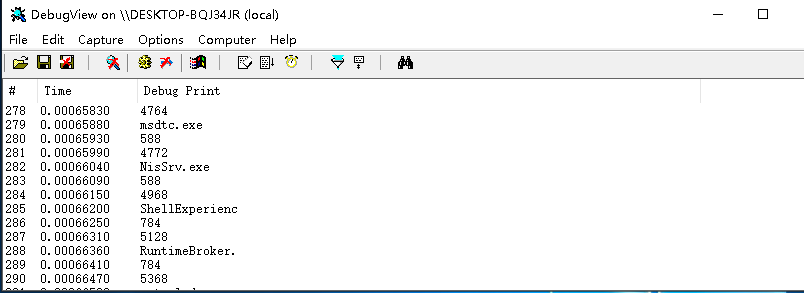
DbgPrint("hello lyshark.com \n");

GetAllProcess();

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

# 运行后将可以在DbgView中看到输出的进程信息：



如果需要返回一个结构体，则可以这样来写代码。

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



typedef struct

{

int count;

char username[256]; char password[256];

}MyData;

// 模拟返回一个结构

BOOLEAN GetProcess(PVOID OutPut)

{

RtlZeroMemory(OutPut, sizeof(MyData)); MyData data = OutPut;

data->count = 100;

RtlCopyMemory(data->username, "lyshark.com", sizeof("lyshark.com")); RtlCopyMemory(data->password, ["https://www.cnblogs.com/lyshark](http://www.cnblogs.com/lyshark)",

sizeof(["https://www.cnblogs.com/lyshark](http://www.cnblogs.com/lyshark)")); return TRUE;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

DbgPrint(("Uninstall Driver Is OK \n"));

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("hello lyshark.com \n");

PVOID Ptr = (PVOID)ExAllocatePool(NonPagedPool, sizeof(MyData));

GetProcess(Ptr);

MyData data = (MyData )Ptr;

DbgPrint("count = %d \n", data->count);

DbgPrint("username = %s \n", data->username); DbgPrint("password = %s \n", data->password);

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

# 输出效果如下：

