内核驱动加载与调试

- 符号服务器 `_NT_SYMBOL_PATH`

```
srv*D:\NtSymbols*https://msdl.microsoft.com/download/symbols
srv*D:\NtSymbols*http://sym.ax2401.com:9999/symbols
```

- 代理服务器 `_NT_SYMBOL_PROXY`

  140.238.19.197:3128

驱动的加载

安装一个驱动等同于安装一个服务

- 打开服务管理器

```
SC_HANDLE OpenSCManagerA(
  LPCSTR lpMachineName,      // 目标计算机的名称，为NULL则是本地
  LPCSTR lpDatabaseName,     // 数据库的名称，为NULL则默认打开
SERVICES_ACTIVE_DATABASE数据库
  DWORD  dwDesiredAccess     // 访问控制，为SC_MANAGER_ALL_ACCESS则是所有权限
);
```

  成功返回服务管理器句柄，失败则为 `NULL`

- 创建或打开服务

```
SC_HANDLE CreateServiceA(
  SC_HANDLE hSCManager,          // 服务管理器句柄
  LPCSTR    lpServiceName,       // 服务名称
  LPCSTR    lpDisplayName,       // 显示名称
  DWORD     dwDesiredAccess,     // 访问控制，SERVICE_ALL_ACCESS为所有权限
  DWORD     dwServiceType,       // 服务类型，SERVICE_KERNEL_DRIVER为内核驱动
  DWORD     dwStartType,         // 启动类型，SERVICE_DEMAND_START为后续调用
StartService才启动
  DWORD     dwErrorControl,      // 如果此服务无法启动，则错误的严重程度和所采取的操
作，SERVICE_ERROR_NORMAL为默认
  LPCSTR    lpBinaryPathName,    // 驱动路径
  LPCSTR    lpLoadOrderGroup,    // 服务所属组名，可为NULL
  LPDWORD   lpdwTagId,           // 可为NULL
  LPCSTR    lpDependencies,      // 可为NULL
  LPCSTR    lpServiceStartName, // 服务应该在其下运行的帐户的名称，可为NULL
  LPCSTR    lpPassword           // 账户密码，，可为NULL
);                               // CreateService成功返回服务句柄，失败则返回NULL


SC_HANDLE OpenServiceA(
  SC_HANDLE hSCManager,          // 服务管理器句柄
  LPCSTR    lpServiceName,       // 服务名称
```

```
  DWORD     dwDesiredAccess     // 访问控制
);                              // OpenService成功返回服务句柄，失败则返回NULL
```

- 启动服务：

```
BOOL StartServiceA(
  SC_HANDLE hService,               // 服务句柄
  DWORD     dwNumServiceArgs,       // lpServiceArgVectors数组的个数，可为0
  LPCSTR    *lpServiceArgVectors    // 可为NULL
);
```

- 停止与卸载服务：

```
// 控制服务
BOOL ControlService(
  SC_HANDLE        hService,        // 服务句柄
  DWORD            dwControl,       // SERVICE_CONTROL_STOP为停止
  LPSERVICE_STATUS lpServiceStatus  // SERVICE_STATUS结构的指针
);

BOOL DeleteService(
  SC_HANDLE hService    // 服务句柄
);
```

- 关闭服务句柄

```
BOOL CloseServiceHandle(
  SC_HANDLE hSCObject           // 服务句柄
);
```

实例

```
CString file_name;  // 驱动路径
SC_HANDLE hSCM;     // 服务管理器句柄
SC_HANDLE hService; // 服务句柄

bool install_drive();
bool start_drive();
bool stop_drive();
bool uninstall_drive();

bool CDriveInstallDlg::install_drive()
{
    // 打开服务管理器
    hSCM = ::OpenSCManager(NULL, NULL, SC_MANAGER_ALL_ACCESS);
    if (!hSCM)
        return false;
```

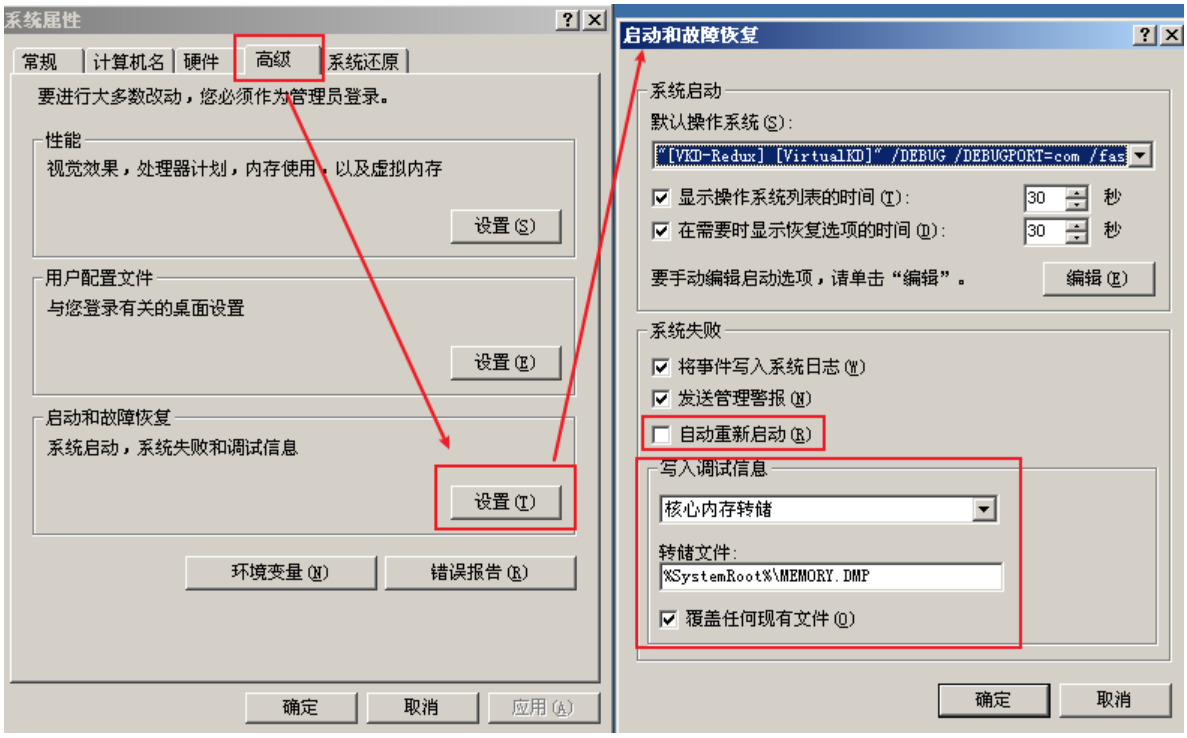
```
    // 创建服务
    hService = ::CreateService(hSCM, TEXT("MyDrive"), TEXT("MyDrive"),
SERVICE_ALL_ACCESS, SERVICE_KERNEL_DRIVER,
        SERVICE_DEMAND_START, SERVICE_ERROR_NORMAL, file_name, NULL, NULL, NULL,
NULL, NULL);
    if(!hService) {
        ::CloseServiceHandle(hSCM);
        hSCM = NULL;
        return false;
    }
    return true;
}

bool CDriveInstallDlg::start_drive()
{
    if(!::StartService(hService, 0, NULL)) {
        uninstall_drive();
        return false;
    }
    return true;
}

bool CDriveInstallDlg::stop_drive()
{
    SERVICE_STATUS status;
    return ::ControlService(hService, SERVICE_CONTROL_STOP, &status);
}

bool CDriveInstallDlg::uninstall_drive()
{
    auto ret = ::DeleteService(hService);
    ::CloseServiceHandle(hService);
    ::CloseServiceHandle(hSCM);
    return ret;
}
```

崩溃分析

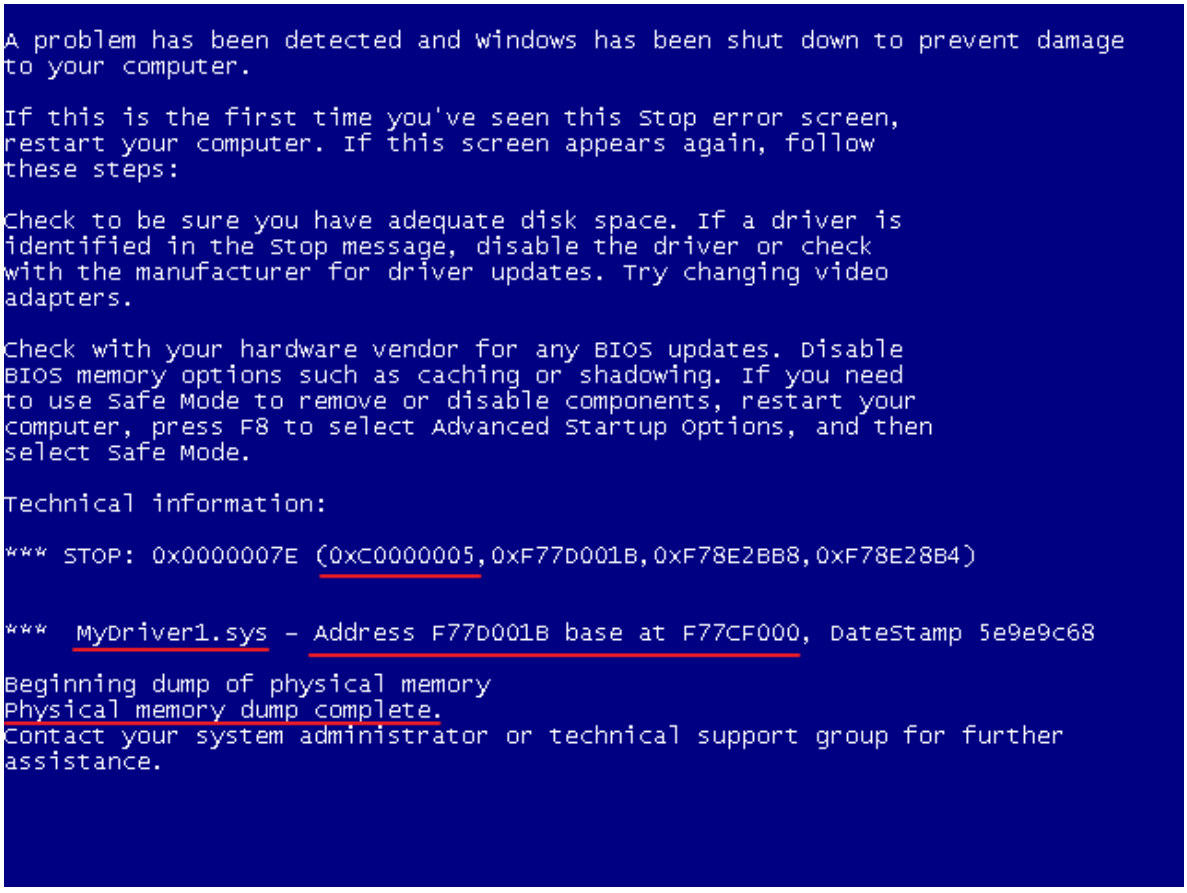
分析dump文件

dump文件设置

我的电脑属性高级选项卡中，进入启动和故障恢复设置，取消勾选**自动启动**，调试信息选择核心**内存转储**

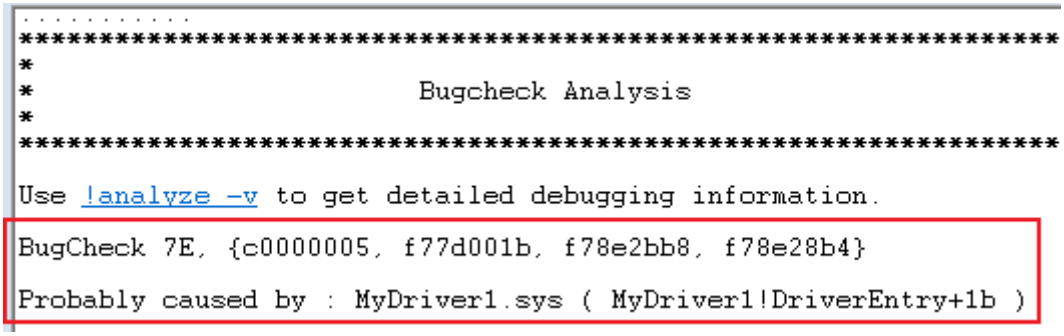


当蓝屏时，会显示故障原因、故障驱动名称和故障地址，并且dump文件已经生成



dump文件一般是C:\WINDOWS\MEMORY.DMP，并使用windbg加载dump文件（Open Crash Dump）

会报告一些崩溃信息



在windbg中可以使用`r`命令查看崩溃时的寄存器环境，`u`命令可查看反汇编等调试命令

输入（或点击）命令`!analyze -v`来智能分析，部分分析结果如下图：

```
THREAD_SHA1_HASH_MOD_FUNC:  d47cb7af7577a6c0024065851fd0ae4e24bd896b

THREAD_SHA1_HASH_MOD_FUNC_OFFSET:  a245d64f94056d056ab454f02f01cd8c25f49648

THREAD_SHA1_HASH_MOD:  7ecdb6fcf13e062c47704708c7a43841c8f08b43

FAULT_INSTR_CODE:  100c7

FAULTING_SOURCE_LINE:  e:\x÷Òµ\Èý½x¶Î\20200421\mydriver1\mydriver1\main.c

FAULTING_SOURCE_FILE:  e:\x÷Òµ\Èý½x¶Î\20200421\mydriver1\mydriver1\main.c

FAULTING_SOURCE_LINE_NUMBER:  21

FAULTING_SOURCE_CODE:
    17: {
    18:     DbgPrint("Hello Kernel! - Install");
    19:
    20:     int *p = NULL;
>   21:     *p = 1;
    22:
    23:     // ¡Á??¨¢????????
    24:     DriverObject->DriverUnload = DriverUnload;
    25:
    26:     return STATUS_SUCCESS;

SYMBOL_STACK_INDEX:  0

SYMBOL_NAME:  MyDriver1!DriverEntry+1b

FOLLOWUP_NAME:  MachineOwner
```

调试时分析

如果在双机调试中崩溃，windbg可直接调试

调试驱动

想要调试驱动程序，则需要在驱动中下断点：`DbgBreakPoint`，此函数就是`int 3`

内核API

内核主模块为`nt`，PDB为`ntkrpamp.pdb`，内核即为`ntkrpamp.exe`，从PE角度来讲，更倾向于属于DLL

分类

内核函数一般会有相关前缀来进行分类，常用的内核函数前缀如下：

- mm 内存相关
- ob 对象相关
- ps 进程线程相关
- io 输入输出管理器
- Ke 核心层
- Ex 执行层
- Zw 系列等价于R3的API，不要用Nt系列的函数，弃用
- Rtl 为内核中C库函数，替代R3的C库

内核使用的字符串

字符串使用`UNICODE_STRING`和`ANSI_STRING`来表示

```
// UNICODE_STRING
typedef struct _UNICODE_STRING {
  USHORT Length;             // 字符串的长度，不包含空字符（以字节为单位），Unicode的字符串就是字符个数*2
  USHORT MaximumLength;      // 缓冲区的总长度（以字节为单位）
  PWSTR  Buffer;             // 缓冲区
} UNICODE_STRING, *PUNICODE_STRING;

// ANSI_STRING
typedef struct _STRING {
  USHORT Length;             // 字符串的长度，不包含空字符（以字节为单位）
  USHORT MaximumLength;      // 缓冲区的总长度（以字节为单位）
  PCHAR  Buffer;             // 缓冲区
} STRING;
```

常用操作：https://likte.me/Windows_Kernel_String_Operations.html