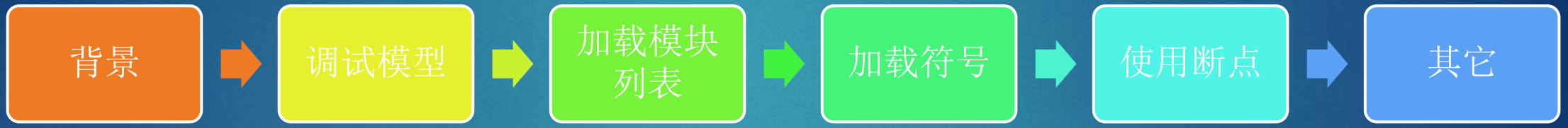
使用DCI OPEN会话调试LINUX内核

张银奎

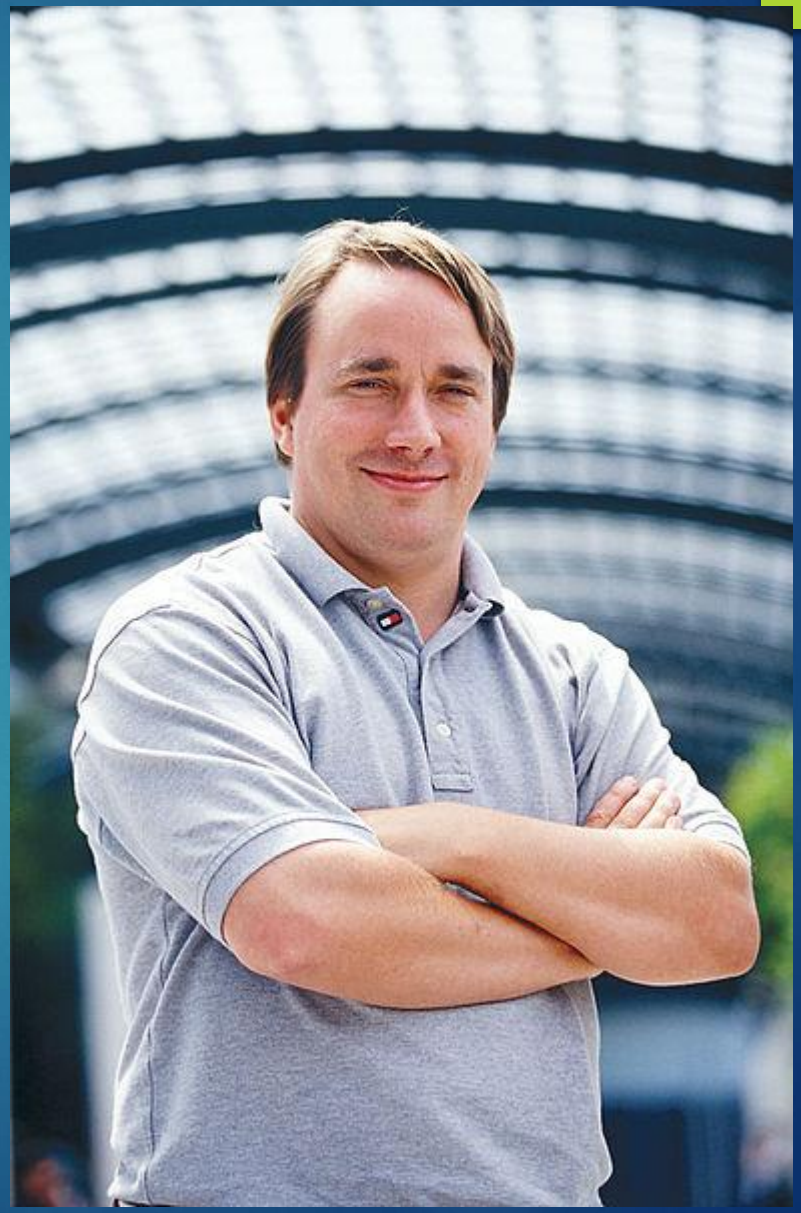
2020/6/13 上海

张银奎，Raymond Zhang，格蠹老雷，《软件调试》和《格蠹汇编》作者
http://advdbg.org http://weibo.com/dbgger 格友公众号


背景
调试模型
加载模块列表
加载符号
使用断点
其它

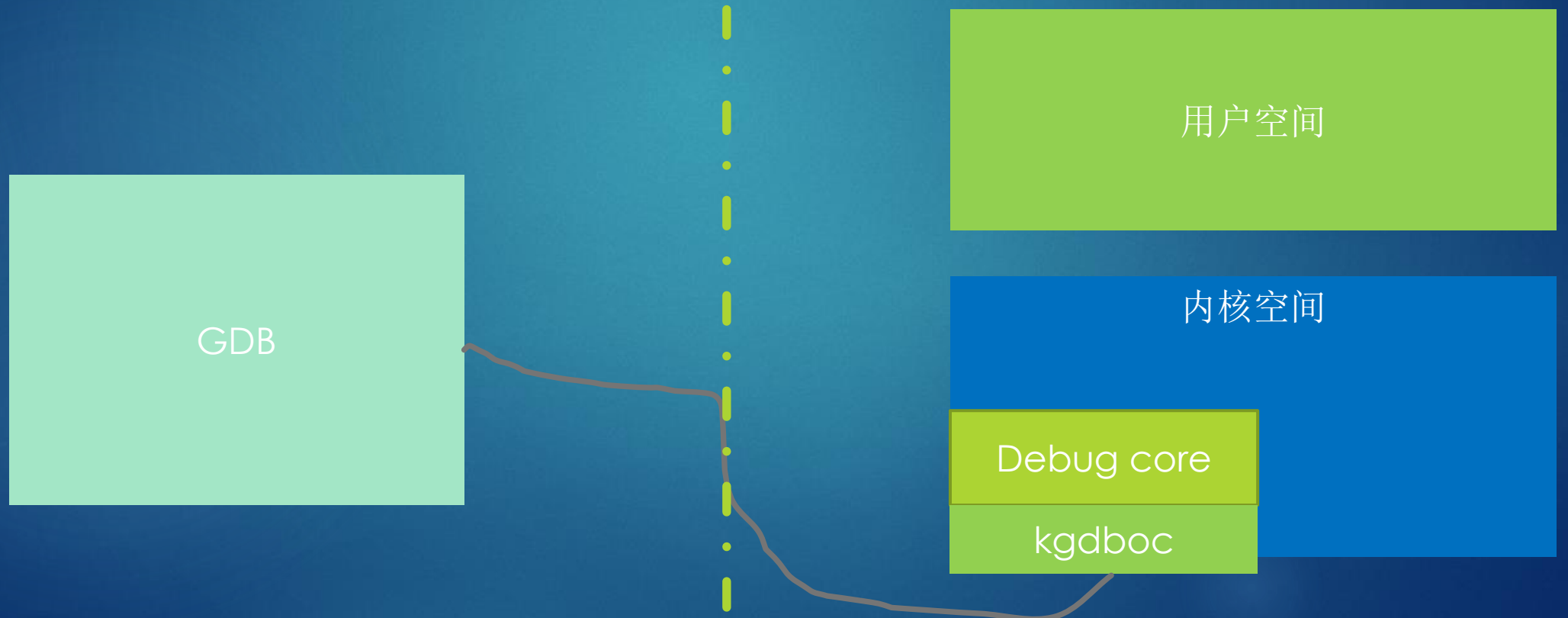
Re: Availability of kdb

- Date: Wed, 6 Sep 2000 12:52:29 -0700 (PDT)
- From: Linus Torvalds
- “I don't like debuggers. Never have, probably never will.”
- “So I don‘t make it part of the standard distribution…”



KGDB

- 源代码级内核调试器
  - GDB前端 + 内核中的调试引擎(后端)



简史

- KGDB was implemented as part of the NetBSD kernel in 1997, and FreeBSD in version 2.2.
- The concept and existing remote gdb protocol were later adapted as a patch to the Linux kernel.
- A scaled-down version of the Linux patch was integrated into the official Linux kernel in version **2.6.26**.
- 2.6.35 - KDB was merged, and uses the same backend as KGDB.
- 2.6.36 - The atomic kernel mode setting (KMS) API was merged (currently on the Intel i915 class of video cards are supported)
- 2.6.37 - Radeon and Nouveau atomic KMS support merged along with improved KDB keyboard support

2.6.26

\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\debug_core.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\debug_core.h
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\gdbstub.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\.gitignore
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_bp.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_bt.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_cmds
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_debugger.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_io.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_keyboard.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_main.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_private.h
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\kdb_support.c
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\kdb\Makefile
\kernel-2.6.35-mfld.src\kernel-2.6.35\kernel\debug\Makefile

- **Subject**: Linux 2.6.26-rc1
- **Date**: Sat, 3 May 2008 12:38:20 -0700 (PDT)
- "Another feature that is notable not for its size, but because people have tried to get me to merge it for some long is kgdb support. Which really turned out pretty small and clean, once people started putting their effort into making it so."

维护者

- Amit Kale maintained the Linux KGDB from 2000 to 2004.
- From 2004 to 2006, it was maintained by Linsyssoft Technologies, after which Jason Wessel at Wind River Systems, Inc. took over as the official maintainer.
- Ingo Molnar and Jason Wessel created a slimmed-down and cleaned up version of KGDB which was called "kgdb light" (without Ethernet support and many other hacks). This was the one merged into the 2.6.26 kernel. This version of kgdb supports only RS-232 connectivity, using a special driver which can split debugger inputs and console inputs such that only a single serial port is required.
- Jason Wessel(Wind River)和Jesse Barnes(Intel)曾经在Linux峰会上多次介绍KGDB

```
arch/um/drivers/ubd_kern.c:                     printk(KERN_ERR "Failed to initialize ubd device %d :"
arch/um/drivers/ubd_kern.c:             printk(KERN_INFO "ubd: Synchronous mode\n");
arch/um/drivers/ubd_kern.c:             printk(KERN_ERR
arch/um/drivers/ubd_kern.c:             printk(KERN_ERR "um_request_irq failed - errno = %d\n", -err);
arch/um/drivers/ubd_kern.c:                     printk(KERN_ERR "%s: Can't open \"%s\": errno = %d\n",
arch/um/drivers/ubd_kern.c:                     printk("write to io thread failed, "
arch/um/drivers/ubd_kern.c:             printk("do_io - bitmap lseek failed : err = %d\n", -n);
arch/um/drivers/ubd_kern.c:             printk("do_io - bitmap update failed, err = %d fd = %d\n", -n,
arch/um/drivers/ubd_kern.c:                     printk("do_io - sync failed err = %d "
arch/um/drivers/ubd_kern.c:                     printk("do_io - lseek failed : err = %d\n", -err);
arch/um/drivers/ubd_kern.c:                                     printk("do_io - read failed, err = %d "
arch/um/drivers/ubd_kern.c:                             printk("do_io - write failed err = %d "
arch/um/drivers/ubd_kern.c:                             printk("io_thread - read failed, fd = %d, "
arch/um/drivers/ubd_kern.c:                             printk("io_thread - short read, fd = %d, "
arch/um/drivers/ubd_kern.c:                     printk("io_thread - write failed, fd = %d, err = %d\n",
arch/um/drivers/daemon_kern.c:  printk("daemon backend (uml_switch version %d) - %s:%s",
arch/um/drivers/daemon_kern.c:  printk("\n");
arch/um/drivers/daemon_kern.c:          printk(KERN_WARNING "daemon_setup : Ignoring data socket "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "new_addr: allocation of sockaddr_un "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control socket failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control connect failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : data socket failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : data bind failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "new_addr: allocation of sockaddr_un "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control setup request "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : read of data socket failed, "
arch/um/drivers/vde_user.c:             printk(UM_KERN_ERR "vde_user_init: vde_open failed, "
arch/um/drivers/vde_user.c:     printk(UM_KERN_INFO "vde backend - connection opened\n");
arch/um/drivers/vde_user.c:     printk(UM_KERN_WARNING "vde_open - we have no VDECONN to open");
arch/um/drivers/vde_user.c:             printk(UM_KERN_INFO "vde backend - closing connection\n");
arch/um/drivers/vde_user.c:     printk(UM_KERN_WARNING "vde_remove - we have no VDECONN to remove");
arch/um/drivers/vde_user.c:             printk(UM_KERN_ERR "vde_init_libstuff - vde_open_args "
arch/um/drivers/vde_user.c:     args->port ?  printk("port %d", args->port) :
arch/um/drivers/vde_user.c:             printk("undefined port");
```

```
ge@gewubox:~/work/linux-3.12.2$ grep -R "printk" * | wc -l
63170
```

问君用printk有几多愁

Home > Applications

Linus Torvalds suspends key Linux developer

Kernel panic as Systemd dev pokes the bear

RELATED

Software ▸ Developer

Torvalds rails at Linux developer: 'I'm f*cking tired of your code'

Kay Sievers banished to fuming Finn's doghouse

https://lkml.org/lkml/2014/4/2/420
http://www.theregister.co.uk/2014/04/05/torvalds_sievers_dust_up/

Date Wed, 2 Apr 2014 11:57:41 -0700
Subject Re: [RFC PATCH] cmdline: Hide "debug" from /proc/cmdline
From Linus Torvalds <>

On Wed, Apr 2, 2014 at 11:42 AM, Steven Rostedt <rostedt@goodmis.org> wrote:
>
> The response is:
>
> "Generic terms are generic, not the first user owns them."

And by "their" you mean Kay Sievers.

Key, I'm f*cking tired of the fact that you don't fix problems in the code *you* write, so that the kernel then has to work around the problems you cause.

Greg - just for your information, I will *not* be merging any code from Kay into the kernel until this constant pattern is fixed.

This has been going on for *years*, and doesn't seem to be getting any better. This is relevant to you because I have seen you talk about the kdbus patches, and this is a heads-up that you need to keep them separate from other work. Let distributions merge it as they need to and maybe we can merge it once it has been proven to be stable by whatever distro that was willing to play games with the developers.

But I'm not willing to merge something where the maintainer is known to not care about bugs and regressions and then forces people in other projects to fix their project. Because I am *not* willing to take patches from people who don't clean up after their problems, and don't admit that it's their problem to fix.

Kay - one more time: you caused the problem, you need to fix it. None of this "I can do whatever I want, others have to clean up after me" crap.

Linus

On Thu, Apr 03, 2014 at 08:17:33AM -0700, Tim Bird wrote:
>
> I had no idea systemd was so verbose and was abusing the kernel
> log buffers so badly. I'm not a big fan of the rate-limiting, as this just
> seems to encourage this kind of abuse.

That was a bug in systemd, and has been fixed up in the latest versions,
so it shouldn't happen anymore, even with debugging enabled.

thanks,

greg k-h
--

http://lkml.iu.edu/hypermail/linux/kernel/1404.0/01945.html

github.com/kaysievers

Kay Sievers

kaysievers

Berlin, Germany

Sign in to view email

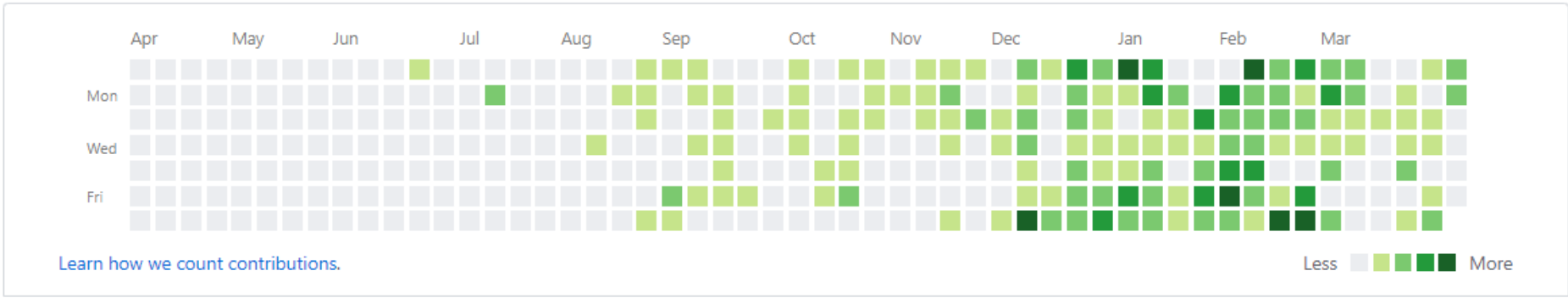
Block or report user

Overview | Repositories 0 | Projects 0 | Stars 0 | Followers 78 | Following 0

Popular repositories

kaysievers doesn't have any public repositories yet.

1,141 contributions in the last year



Contribution activity

April 2020

Created 34 commits in 2 repositories

versioduo/midiusb 33 commits

versioduo/download 1 commit

2020

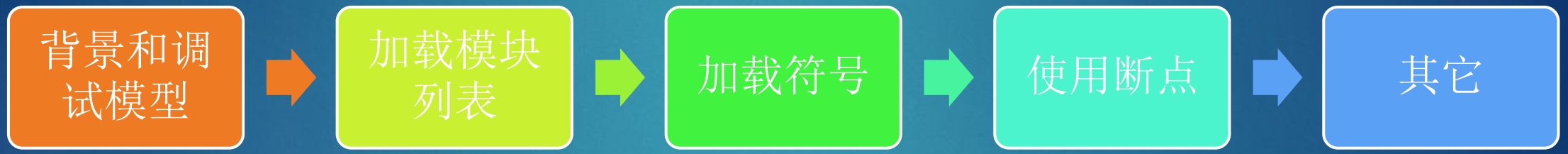
2019

2018

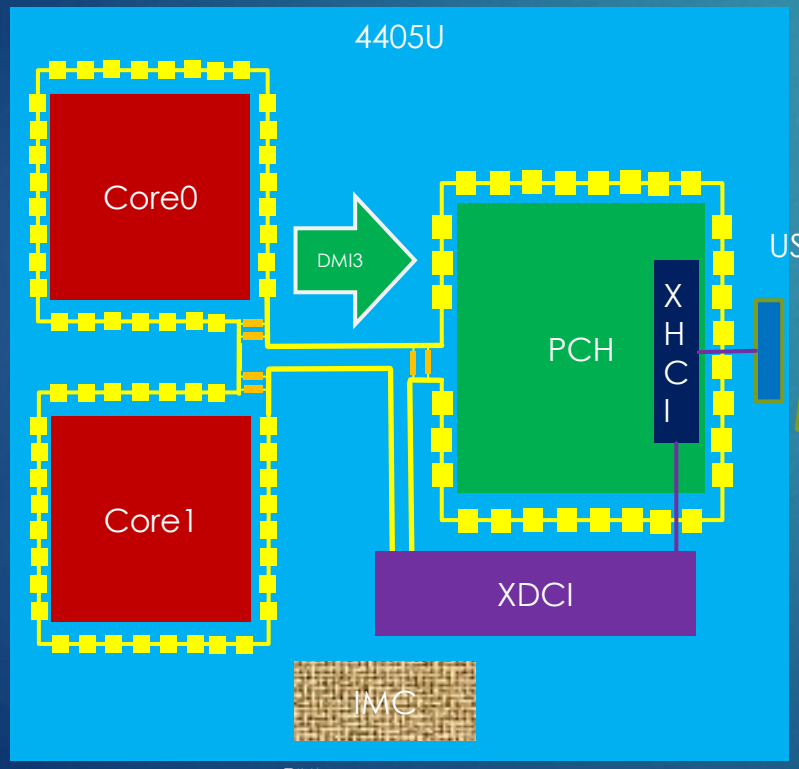
2017

versioduo/
arduino-board-package

Versio Duo
Software and Electronics for Musical Devices


背景和调试模型
加载模块列表
加载符号
使用断点
其它

调试模型



DCI

USB3.0接口

OpenIPC_x64.exe

NanoCode

DbgEng.dll

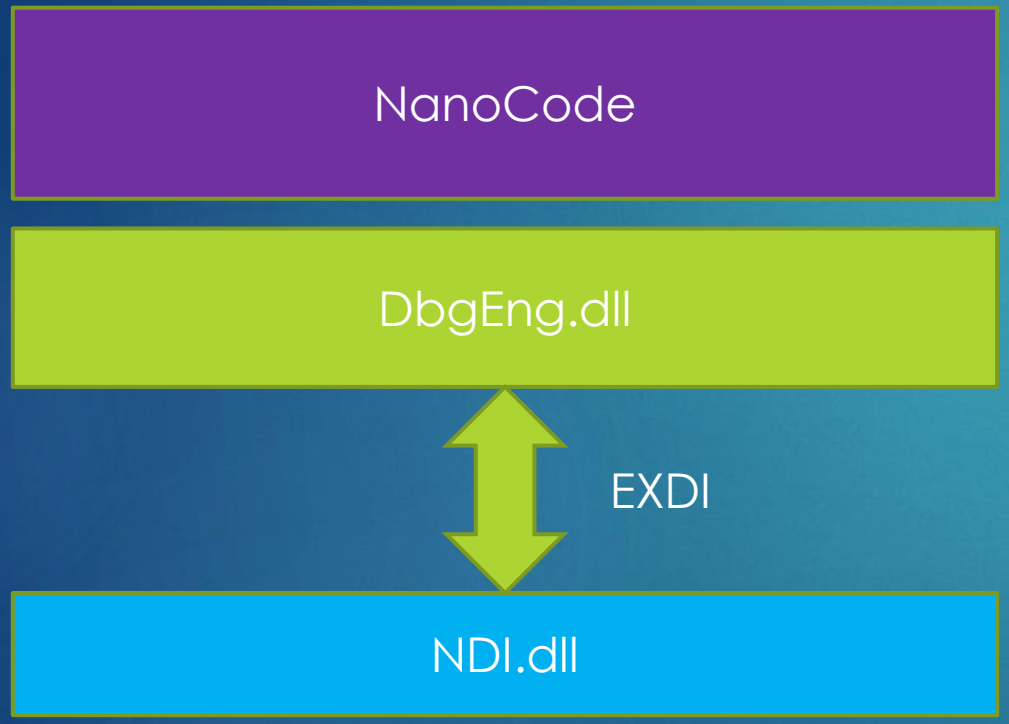
NDI.dll

Dciusb.sys

X86硬件

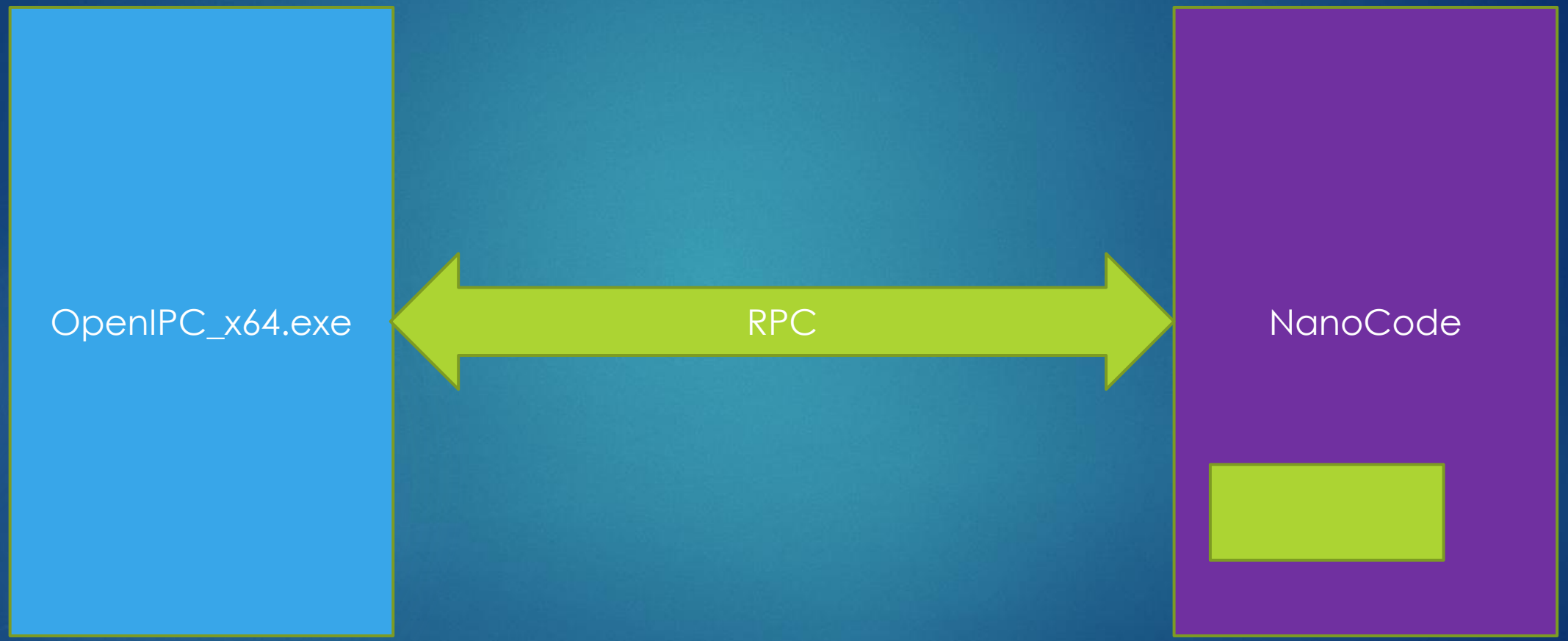
调试主机

EXDI



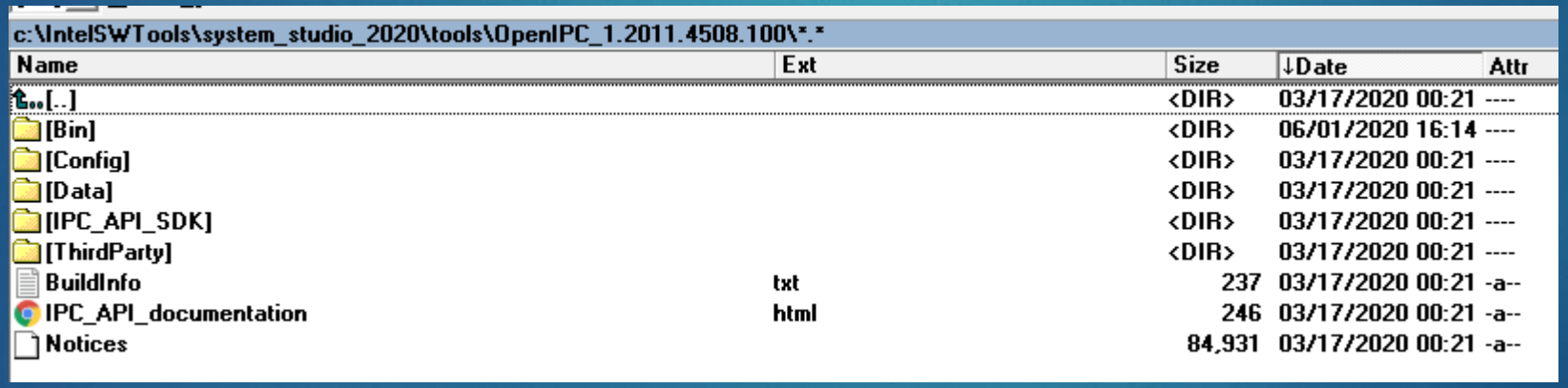
- eXtended Debug Interface
- DbgEng与外部调试器的接口
- 最初版本似乎为1999年
  - Greg Hogdal 11-Nov-1999
- 目前使用的v3版本
  - Ivan Shcherbakov (ivshcher) 31-Oct-2013
- NDI实现了一个进程内的COM服务器

进程关系图



IPC API

- INTEL的OpenIPC开发接口
- 似乎公开
- 但其实没有公开



c:\IntelSWTools\system_studio_2020\tools\OpenIPC_1.2011.4508.100*.*

Name	Ext	Size	↓Date	Attr
[..]		<DIR>	03/17/2020 00:21	----
[Bin]		<DIR>	06/01/2020 16:14	----
[Config]		<DIR>	03/17/2020 00:21	----
[Data]		<DIR>	03/17/2020 00:21	----
[IPC_API_SDK]		<DIR>	03/17/2020 00:21	----
[ThirdParty]		<DIR>	03/17/2020 00:21	----
BuildInfo	txt	237	03/17/2020 00:21	-a--
IPC_API_documentation	html	246	03/17/2020 00:21	-a--
Notices		84,931	03/17/2020 00:21	-a--

巨大的诱惑

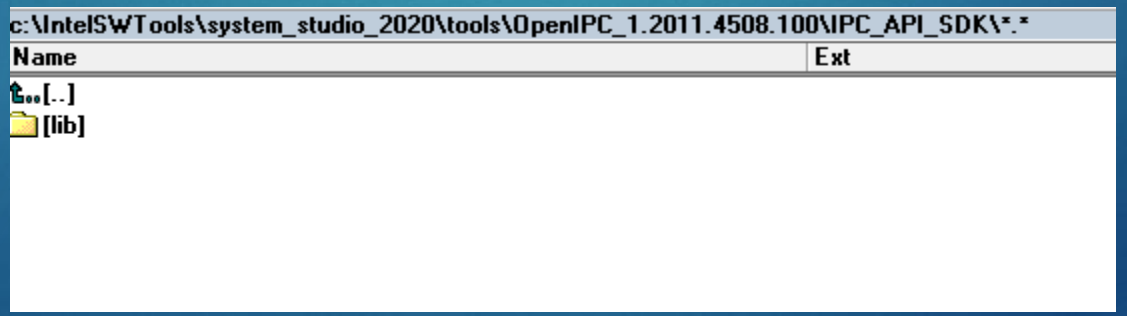
IPC_API_documentation.html - Notepad

File Edit Format View Help

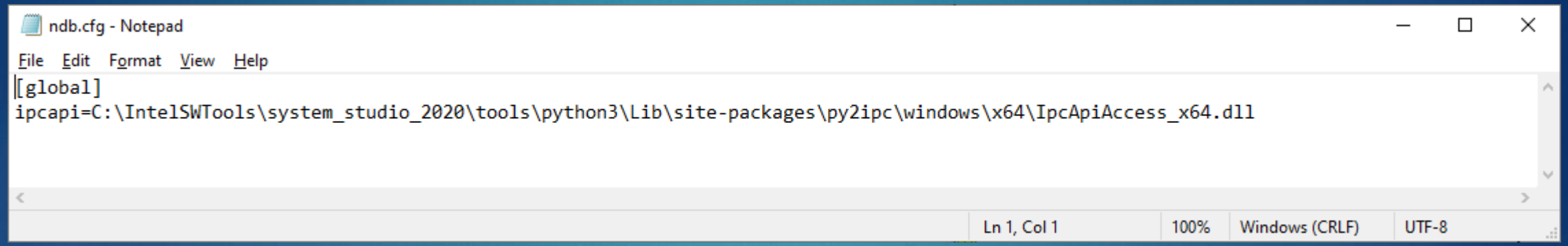
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="Refresh"
 content="0; URL=IPC_API_SDK\doc\html\index.html">
  <title>IPC API Documentation</title>
</head>
<body>

<br>
</body>
</html>
```

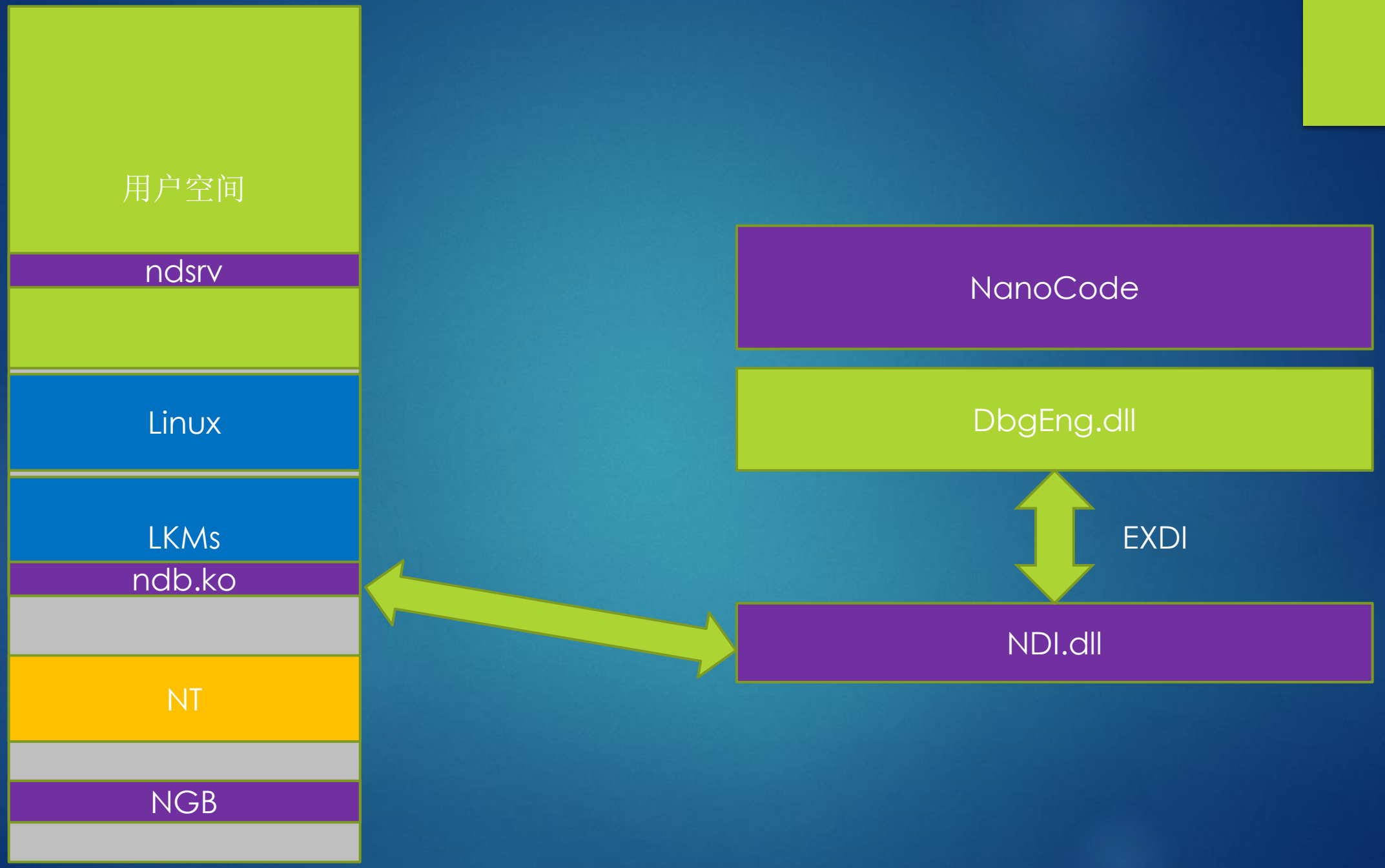
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

- Doc目录不存在



配置文件


ndb.cfg - Notepad
File Edit Format View Help
[global]
ipcapi=C:\IntelSWTools\system_studio_2020\tools\python3\Lib\site-packages\py2ipc\windows\x64\IpcApiAccess_x64.dll
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8


用户空间
ndsrv
Linux
LKMs
ndb.ko
NT
NGB
NanoCode
DbgEng.dll
EXDI
NDI.dll

NDB.ko

- Nano Debugger Beacon
- 也是NDB在目标端的代表
- 主要功能：
  - 产生Nano Geography Block
  - 传递关键的内核结构信息
  - 其它辅助功能

NdSrv

Nd Front

Linux Kernel and Other Modules

NdBeacon

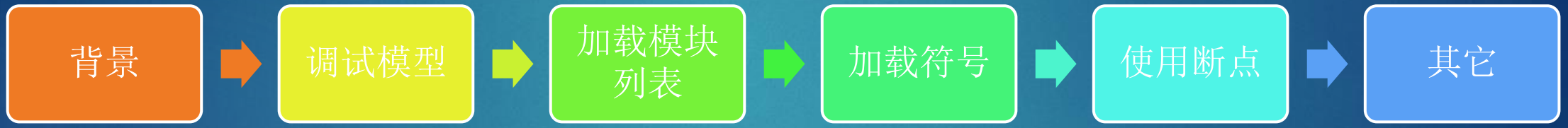
Kernel Geo Block

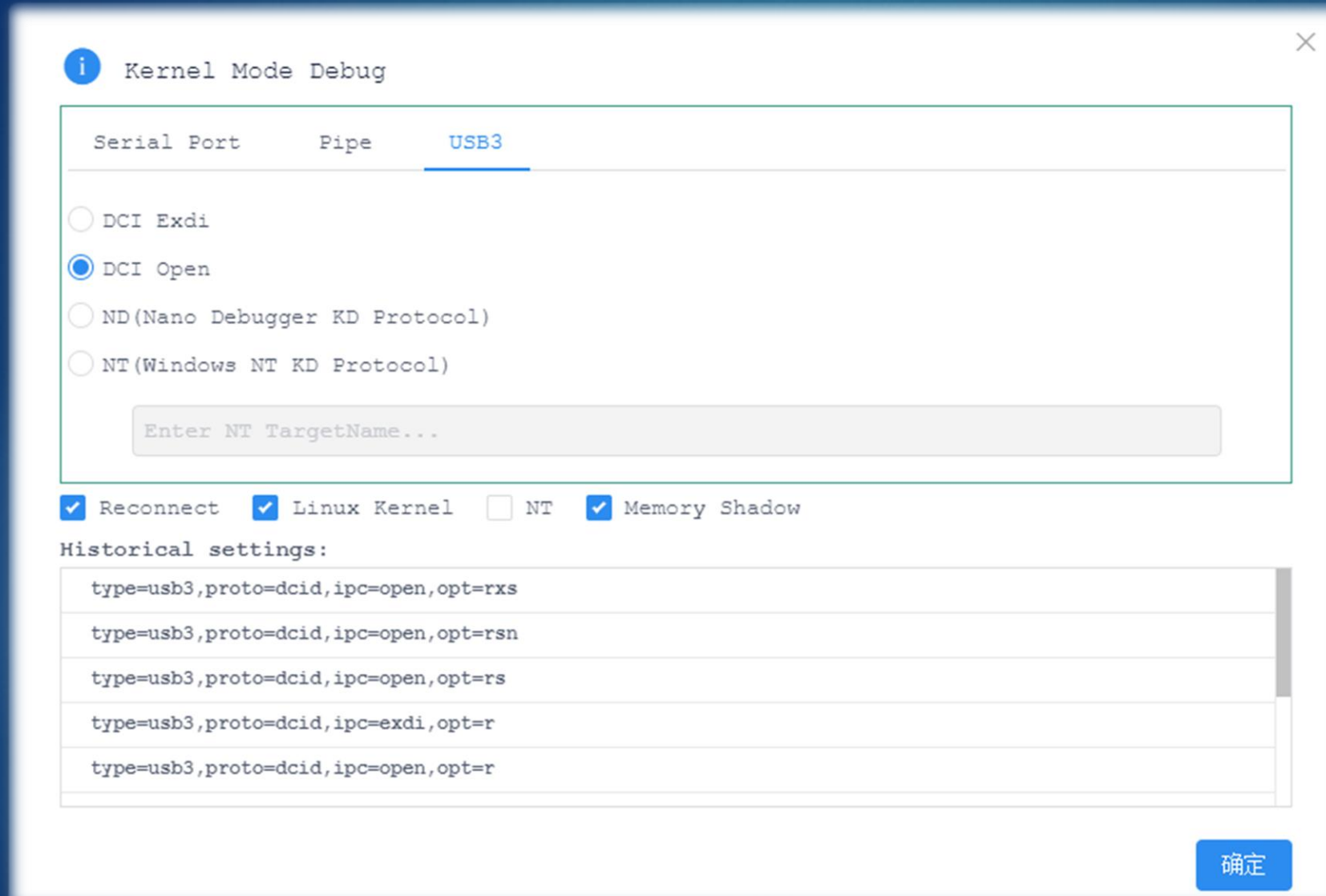
DCI/JTAG

DCI Driver

Hardware

Target Host


背景
调试模型
加载模块列表
加载符号
使用断点
其它



选中

- Linux Kernel
- Memory Shadow
- Reconnect

EXPLORER

OPEN EDITORS
- MAINTAINERS
- Nano Home
- Nano Course
- Nano Debugger

LINUX-4.16

OUTLINE

MAINTAINERS | Nano Home | Nano Course | Nano Debugger

File View Output

```
Nano Debugger (NDB) 1.0.252
Starting...
Starting KD session type=usb3,proto=dcid,ipc=open,opt=rxs

Microsoft (R) Windows Debugger Version 10.0.17763.132 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

All logic CPU threads detected: 1000 1001 1002 1003
Switched to processor 0, its device id is 0x1000
Kernel Debugger connection established
Found NGB marker in target memory at fffffffff5ff000
Connected to Windows 7 7601 x64 target at (Sat Jun 13 15:27:19.610 2020 (UTC + 8:00)), ptr64 TRUE

************* Path validation summary **************
Response Time (ms) Location
Deferred SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Unable to create shared user data image
KdDebuggerData.KernBase < SystemRangeStart
Windows 7 Kernel Version 7601 MP (4 procs) Free x64
Machine Name:
Kernel base = 0xffeeffee`80000000 PsLoadedModuleList = 0xfffd0000`00004028
System Uptime: not available
ffffffff`a2436897 65488b0425005c0100 mov rax,qword ptr gs:[15C00h]
```

0: kd>

0% 100%

Nano 欢迎您-gdman

Type here to search

CH 3:27 PM 6/13/2020

```
Nano Debugger (NDB) 1.0.252
Starting...
Starting KD session type=usb3,proto=dcid,ipc=open,opt=rxs

Microsoft (R) Windows Debugger Version 10.0.17763.132 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

All logic CPU threads detected: 1000 1001 1002 1003
Switched to processor 0, its device id is 0x1000
Kernel Debugger connection established
Found NGB marker in target memory at fffffffff5ff000
Connected to Windows 7 7601 x64 target at (Sat Jun 13 15:27:19.610 2020 (UTC + 8:00)), ptr64 TRUE

************* Path validation summary **************
Response Time (ms) Location
Deferred SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
Unable to create shared user data image
KdDebuggerData.KernBase < SystemRangeStart
Windows 7 Kernel Version 7601 MP (4 procs) Free x64
Machine Name:
Kernel base = 0xffeeffee`80000000 PsLoadedModuleList = 0xfffd0000`00004028
System Uptime: not available
fffffff`a2436897 65488b0425005c0100 mov rax,qword ptr gs:[15C00h]
```

```
start end module name
ffeeffee`7ffff000 ffeeffee`809ff000 nt (deferred)
ffffffff`7ffff000 ffffffff`9ffff000 lk (deferred)
ffffffff`c02cc000 ffffffff`c02d1000 ulpi (deferred)
ffffffff`c02d2000 ffffffff`c02dd000 autofs4 (deferred)
ffffffff`c02e9000 ffffffff`c02f5000 video (deferred)
ffffffff`c02f6000 ffffffff`c02fa000 hid_generic (deferred)
ffffffff`c02fb000 ffffffff`c0300000 dwc3_pci (deferred)
ffffffff`c0305000 ffffffff`c030d000 libahci (deferred)
ffffffff`c0312000 ffffffff`c0317000 realtek (deferred)
ffffffff`c0318000 ffffffff`c0322000 ahci (deferred)
ffffffff`c032a000 ffffffff`c033e000 r8169 (deferred)
ffffffff`c0346000 ffffffff`c036b000 psmouse (deferred)
ffffffff`c036c000 ffffffff`c0376000 x_tables (deferred)
ffffffff`c0378000 ffffffff`c0385000 udc_core (deferred)
ffffffff`c038d000 ffffffff`c03ac000 dwc3 (deferred)
ffffffff`c03ad000 ffffffff`c03ba000 usbhid (deferred)
ffffffff`c03bb000 ffffffff`c03c3000 ip_tables (deferred)
ffffffff`c03c6000 ffffffff`c03cd000 wmi (deferred)
ffffffff`c03ce000 ffffffff`c03ed000 hid (deferred)
ffffffff`c03f2000 ffffffff`c03ff000 parport (deferred)
ffffffff`c0405000 ffffffff`c040a000 lp (deferred)
ffffffff`c040f000 ffffffff`c0415000 ppdev (deferred)
ffffffff`c0416000 ffffffff`c041f000 parport_pc (deferred)
ffffffff`c043a000 ffffffff`c043e000 ndb (deferred)
```

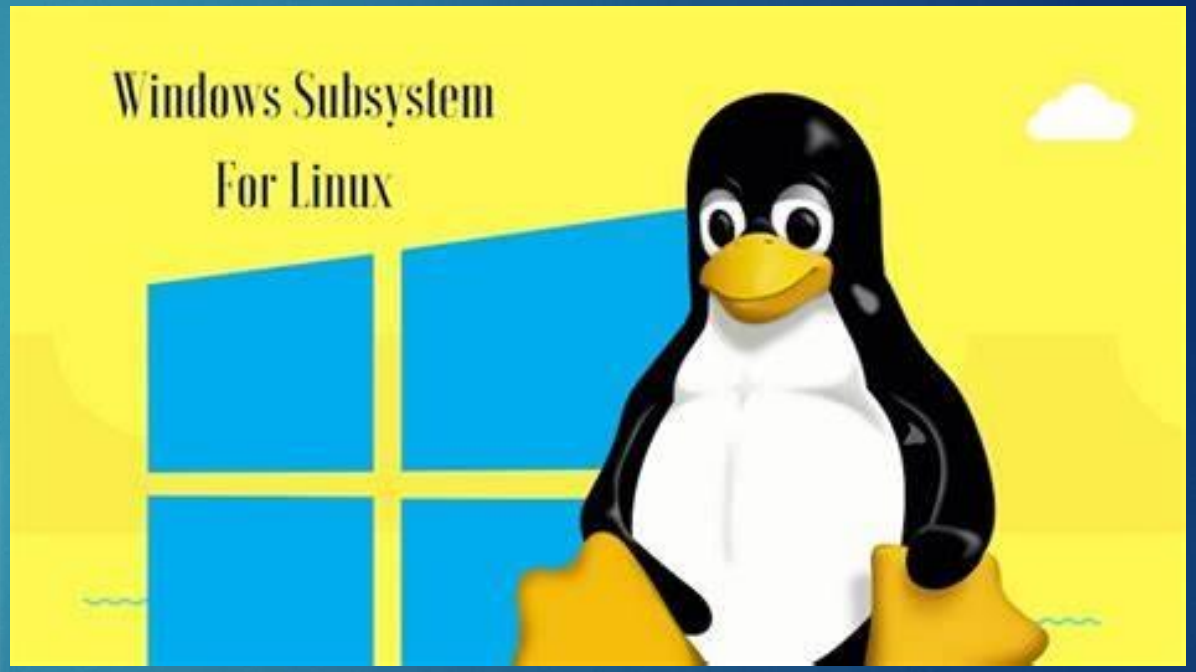
```
start end module name
ffeeffee`7ffff000 ffeeffee`809ff000 nt (deferred)
ffffffff`7ffff000 ffffffff`9ffff000 lk (deferred)
ffffffff`c02cc000 ffffffff`c02d1000 ulpi (deferred)
ffffffff`c02d2000 ffffffff`c02dd000 autofs4 (deferred)
ffffffff`c02e9000 ffffffff`c02f5000 video (deferred)
ffffffff`c02f6000 ffffffff`c02fa000 hid_generic (deferred)
ffffffff`c02fb000 ffffffff`c0300000 dwc3_pci (deferred)
ffffffff`c0305000 ffffffff`c030d000 libahci (deferred)
ffffffff`c0312000 ffffffff`c0317000 realtek (deferred)
ffffffff`c0318000 ffffffff`c0322000 ahci (deferred)
ffffffff`c032a000 ffffffff`c033e000 r8169 (deferred)
ffffffff`c0346000 ffffffff`c036b000 psmouse (deferred)
ffffffff`c036c000 ffffffff`c0376000 x_tables (deferred)
ffffffff`c0378000 ffffffff`c0385000 udc_core (deferred)
ffffffff`c038d000 ffffffff`c03ac000 dwc3 (deferred)
ffffffff`c03ad000 ffffffff`c03ba000 usbhid (deferred)
ffffffff`c03bb000 ffffffff`c03c3000 ip_tables (deferred)
ffffffff`c03c6000 ffffffff`c03cd000 wmi (deferred)
ffffffff`c03ce000 ffffffff`c03ed000 hid (deferred)
ffffffff`c03f2000 ffffffff`c03ff000 parport (deferred)
ffffffff`c0405000 ffffffff`c040a000 lp (deferred)
ffffffff`c040f000 ffffffff`c0415000 ppdev (deferred)
ffffffff`c0416000 ffffffff`c041f000 parport_pc (deferred)
ffffffff`c043a000 ffffffff`c043e000 ndb (deferred)
```

NT的影子

- DdgEng
- 两大内核进入交融的时代
- WSL – Windows Subsystem for Linux

- NT's shadow in Linux

```
tart end module name
feeffee`7ffff000 ffeeffee`809ff000 nt (deferred)
mage path: nt
mage name: nt
imestamp: Mon Jun 10 10:35:18 2024 (66666666)
heckSum: 00000000
mageSize: 00A00000
ranslations: 0000.04b0 0000.04e4 0409.04b0 0409.04e4
```



内核老大

```
.sympath+ c:\temp
Symbol search path is: SRV*c:\symbols*http://msdl.microsoft.com/download/symbols;c:\temp
Expanded Symbol search path is: srv*c:\symbols*http://msdl.microsoft.com/download/symbols;c:\temp

************** Path validation summary **************
Response Time (ms) Location
Deferred SRV*c:\symbols*http://                    nload/symbols
OK c:\temp
lm vm lk
start end module name
ffffffff`7ffff000 ffffffff`9ffff000 lk (private pdb symbols) c:\temp\linux
Loaded symbol image file: c:\temp\linux
Image path: linux
Image name: linux
Timestamp: Mon Oct 9 04:10:08 1995 (30783020)
CheckSum: 00000000
ImageSize: 20000000
Translations: 0000.04b0 0000.04e4 0409.04b0 0409.04e4
```

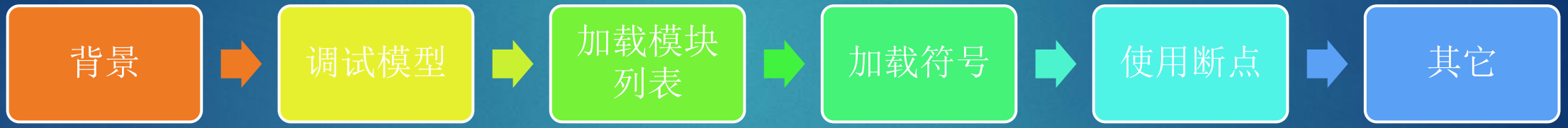
小名lk

Linux Kernel的简写
便于输入
便于交流

必也正名乎？

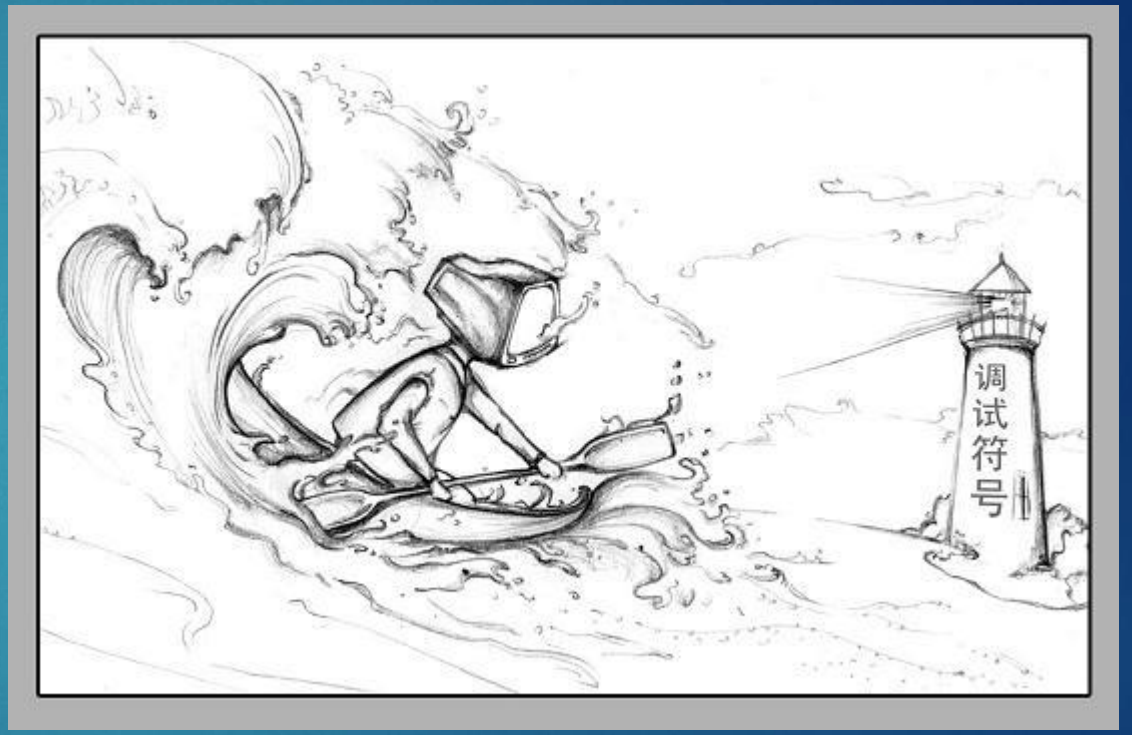
名篇欣赏

子路曰：“卫君待子而为政，子将奚先？”
子曰：“必也正名乎。”
子路曰：“有是哉，子之迂也！奚其正？”
子曰：“野哉由也！君子于其所不知，盖阙如也。名不正，则言不顺；言不顺，则事不成；事不成，则礼乐不兴；礼乐不兴，则刑罚不中；刑罚不中，则民无所措手足。故君子名之必可言也，言之必可行也。君子于其言，无所苟而已矣。”


背景
调试模型
加载模块列表
加载符号
使用断点
其它

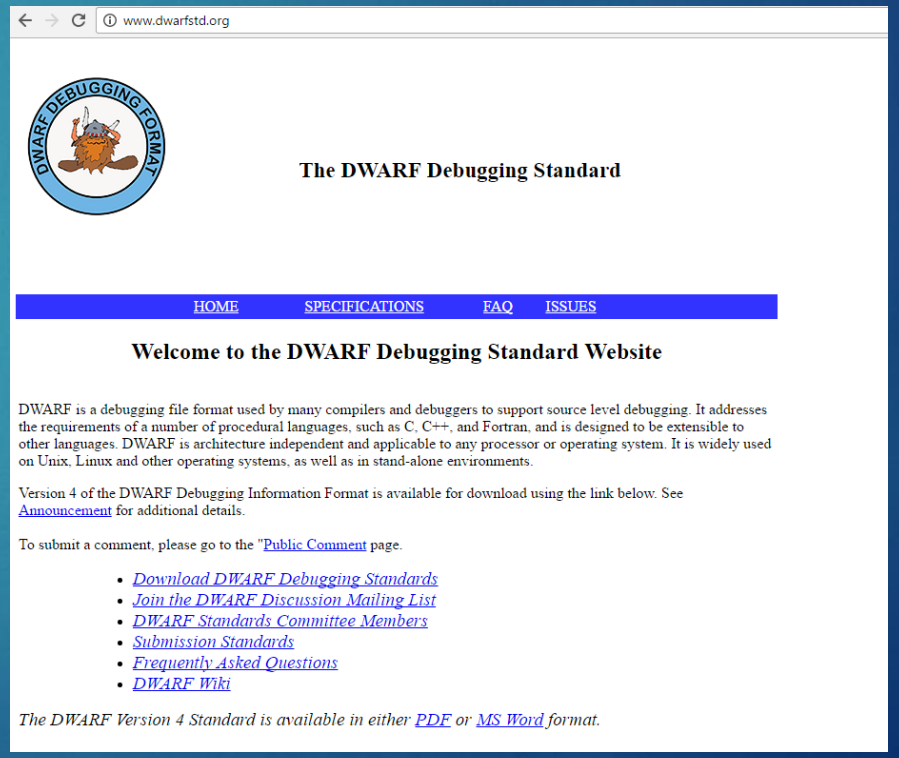
调试符号

- 编译器对调试的重大贡献
  - 编译过程的副产品
- 衔接二进制程序与源程序的桥梁
- 对调试有着重要意义
- 源代码级调试必须
- 二进制跟踪时的灯塔



DWARF

- DWARF Debugging Information Format
- http://www.dwarfstd.org
- 功莫大焉



以ELF为容器

```
ge@gewubox:~/work/llaolao3$ readelf -h baner
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048530
  Start of program headers:          52 (bytes into file)
  Start of section headers:          5824 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         36
  Section header string table index: 33
```

段表

```
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048154 000154 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048168 000168 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048188 000188 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        080481ac 0001ac 000020 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481cc 0001cc 0000f0 10   A  6   1  4
  [ 6] .dynstr           STRTAB          080482bc 0002bc 00009f 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          0804835c 00035c 00001e 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         0804837c 00037c 000030 00   A  6   1  4
  [ 9] .rel.dyn          REL             080483ac 0003ac 000008 08   A  5   0  4
  [10] .rel.plt          REL             080483b4 0003b4 000068 08   A  5  12  4
  [11] .init             PROGBITS        0804841c 00041c 00002e 00  AX  0   0  4
  [12] .plt              PROGBITS        08048450 000450 0000e0 04  AX  0   0 16
  [13] .text             PROGBITS        08048530 000530 00050c 00  AX  0   0 16
  [14] .fini             PROGBITS        08048a3c 000a3c 00001a 00  AX  0   0  4
  [15] .rodata           PROGBITS        08048a58 000a58 00013b 00   A  0   0  4
  [16] .eh_frame_hdr     PROGBITS        08048b94 000b94 000044 00   A  0   0  4
  [17] .eh_frame         PROGBITS        08048bd8 000bd8 000108 00   A  0   0  4
  [18] .ctors            PROGBITS        08049f14 000f14 000008 00  WA  0   0  4
  [19] .dtors            PROGBITS        08049f1c 000f1c 000008 00  WA  0   0  4
  [20] .jcr              PROGBITS        08049f24 000f24 000004 00  WA  0   0  4
  [21] .dynamic          DYNAMIC         08049f28 000f28 0000c8 08  WA  6   0  4
  [22] .got              PROGBITS        08049ff0 000ff0 000004 04  WA  0   0  4
  [23] .got.plt          PROGBITS        08049ff4 000ff4 000040 04  WA  0   0  4
  [24] .data             PROGBITS        0804a034 001034 000008 00  WA  0   0  4
  [25] .bss              NOBITS          0804a03c 00103c 000008 00  WA  0   0  4
  [26] .comment          PROGBITS        00000000 00103c 00002a 01  MS  0   0  1
  [27] .debug_aranges    PROGBITS        00000000 001066 000020 00      0   0  1
  [28] .debug_info       PROGBITS        00000000 001086 0001e8 00      0   0  1
  [29] .debug_abbrev     PROGBITS        00000000 00126e 000110 00      0   0  1
  [30] .debug_line       PROGBITS        00000000 00137e 000091 00      0   0  1
  [31] .debug_str        PROGBITS        00000000 00140f 0000bf 01  MS  0   0  1
  [32] .debug_loc        PROGBITS        00000000 0014ce 0000a8 00      0   0  1
  [33] .shstrtab         STRTAB          00000000 001576 000147 00      0   0  1
  [34] .symtab           SYMTAB          00000000 001c60 000530 10     35  51  4
  [35] .strtab           STRTAB          00000000 002190 0002ca 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

DWARF符号

感受DWARF – 编译单元

```
Contents of the .debug_info section:

  Compilation Unit @ offset 0x0:
   Length:        0x1e4 (32-bit)
   Version:       2
   Abbrev Offset: 0
   Pointer Size:  4
 <0><b>: Abbrev Number: 1 (DW_TAG_compile_unit)
    <c>   DW_AT_producer    : (indirect string, offset: 0x38): GNU C 4.6.3
    <10>   DW_AT_language    : 1    (ANSI C)
    <11>   DW_AT_name        : (indirect string, offset: 0x0): baner.c
    <15>   DW_AT_comp_dir    : (indirect string, offset: 0xa8): /home/ge/work/llaolao3
    <19>   DW_AT_low_pc      : 0x80485e4
    <1d>   DW_AT_high_pc     : 0x804898f
    <21>   DW_AT_stmt_list   : 0x0
 <1><25>: Abbrev Number: 2 (DW_TAG_base_type)
    <26>   DW_AT_byte_size   : 1
    <27>   DW_AT_encoding    : 8    (unsigned char)
    <28>   DW_AT_name        : (indirect string, offset: 0x11): unsigned char
```

TAG(标签)

```
<1><a0>: Abbrev Number: 8 (DW_TAG_subprogram)
   <a1>   DW_AT_external    : 1
   <a2>   DW_AT_name        : (indirect string, offset: 0x8): hd_ioctl
   <a6>   DW_AT_decl_file   : 1
   <a7>   DW_AT_decl_line   : 14
   <a8>   DW_AT_prototyped  : 1
   <a9>   DW_AT_low_pc      : 0x80485fd
   <ad>   DW_AT_high_pc     : 0x804876a
   <b1>   DW_AT_frame_base  : 0x38 (location list)
   <b5>   DW_AT_sibling     : <0x11d>
<2><b9>: Abbrev Number: 9 (DW_TAG_formal_parameter)
   <ba>   DW_AT_name        : fd
   <bd>   DW_AT_decl_file   : 1
   <be>   DW_AT_decl_line   : 14
   <bf>   DW_AT_type        : <0x4f>
   <c3>   DW_AT_location    : 2 byte block: 91 0   (DW_OP_fbreg: 0)
<2><c6>: Abbrev Number: 9 (DW_TAG_formal_parameter)
   <c7>   DW_AT_name        : cmd
   <cb>   DW_AT_decl_file   : 1
   <cc>   DW_AT_decl_line   : 14
   <cd>   DW_AT_type        : <0x6b>
   <d1>   DW_AT_location    : 2 byte block: 91 4   (DW_OP_fbreg: 4)
<2><d4>: Abbrev Number: 9 (DW_TAG_formal_parameter)
   <d5>   DW_AT_name        : arg
   <d9>   DW_AT_decl_file   : 1
   <da>   DW_AT_decl_line   : 14
   <db>   DW_AT_type        : <0x6b>
   <df>   DW_AT_location    : 2 byte block: 91 8   (DW_OP_fbreg: 8)
<2><e2>: Abbrev Number: 10 (DW_TAG_variable)
   <e3>   DW_AT_name        : val
   <e7>   DW_AT_decl_file   : 1
   <e8>   DW_AT_decl_line   : 16
   <e9>   DW_AT_type        : <0x4f>
   <ed>   DW_AT_location    : 2 byte block: 91 6c  (DW_OP_fbreg: -20)
<2><f0>: Abbrev Number: 11 (DW_TAG_lexical_block)
   <f1>   DW_AT_low_pc      : 0x8048628
   <f5>   DW_AT_high_pc     : 0x804874d
```

产生符号

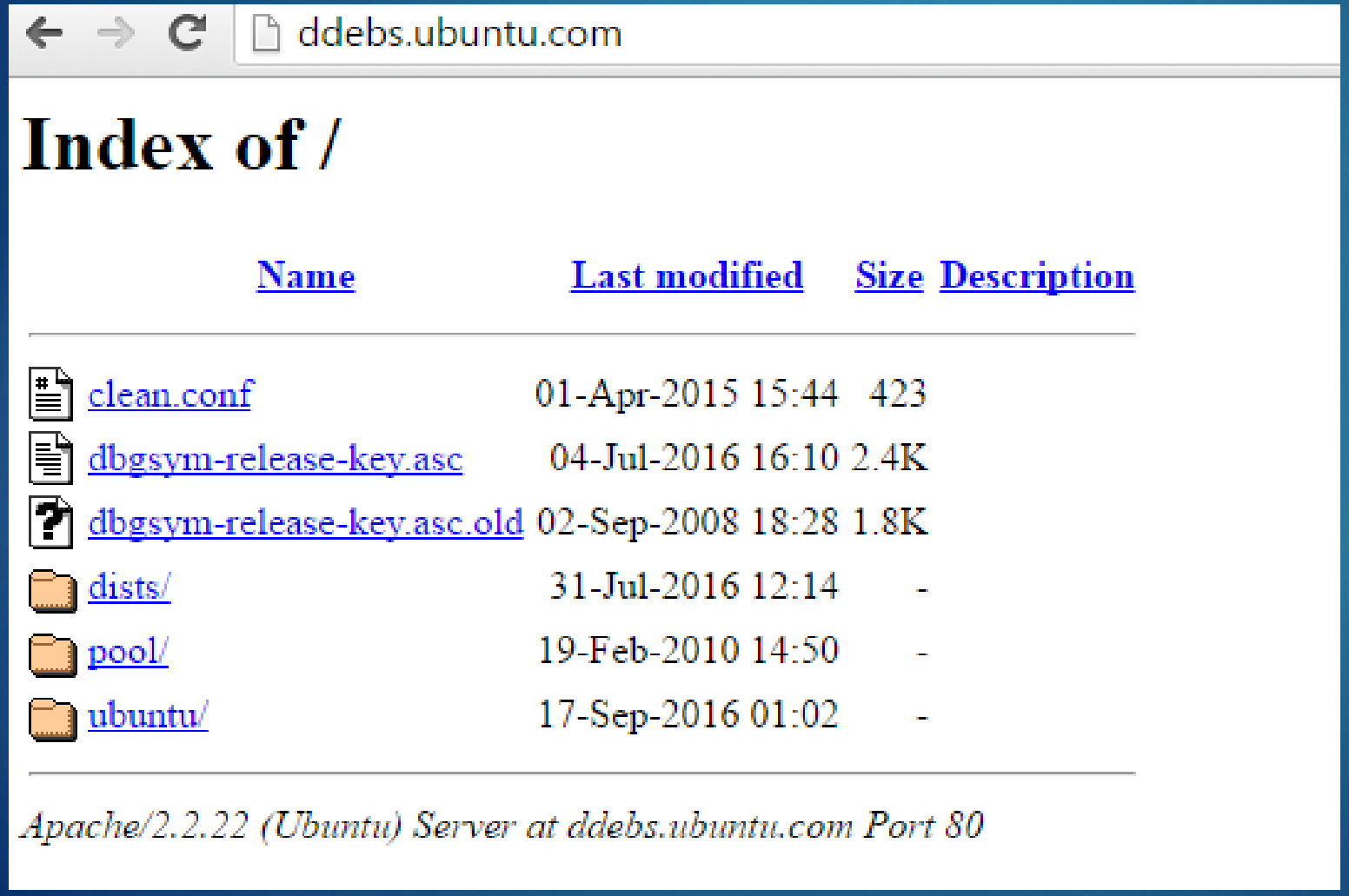
- $ gcc –g –o baner baner.c
- -g --gen-debug generate debugging information

```
--debug-prefix-map OLD=NEW
                        map OLD to NEW in debug information
--defsym SYM=VAL        define symbol SYM to given value
--execstack             require executable stack for this object
--noexecstack           don't require executable stack for this object
--size-check=[error|warning]
                        ELF .size directive check (default --size-check=error)
-f                      skip whitespace and comment preprocessing
-g --gen-debug          generate debugging information
--gstabs                generate STABS debugging information
--gstabs+               generate STABS debug info with GNU extensions
--gdwarf-2              generate DWARF2 debugging information
--hash-size=<value>     set the hash table size close to <value>
--help                  show this message and exit
```

Ubuntu的符号服务器

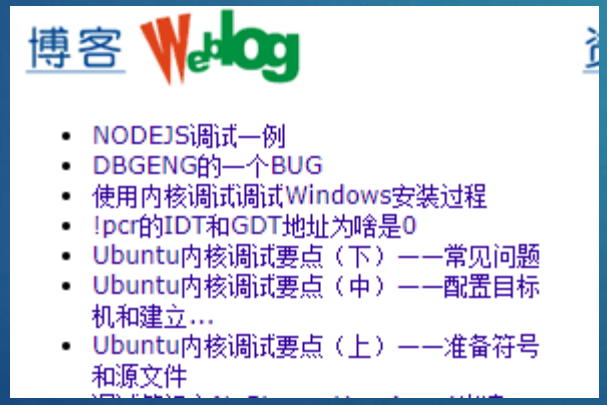
- http://ddebs.ubuntu.com/pool/main/l/linux/

服务器的根目录

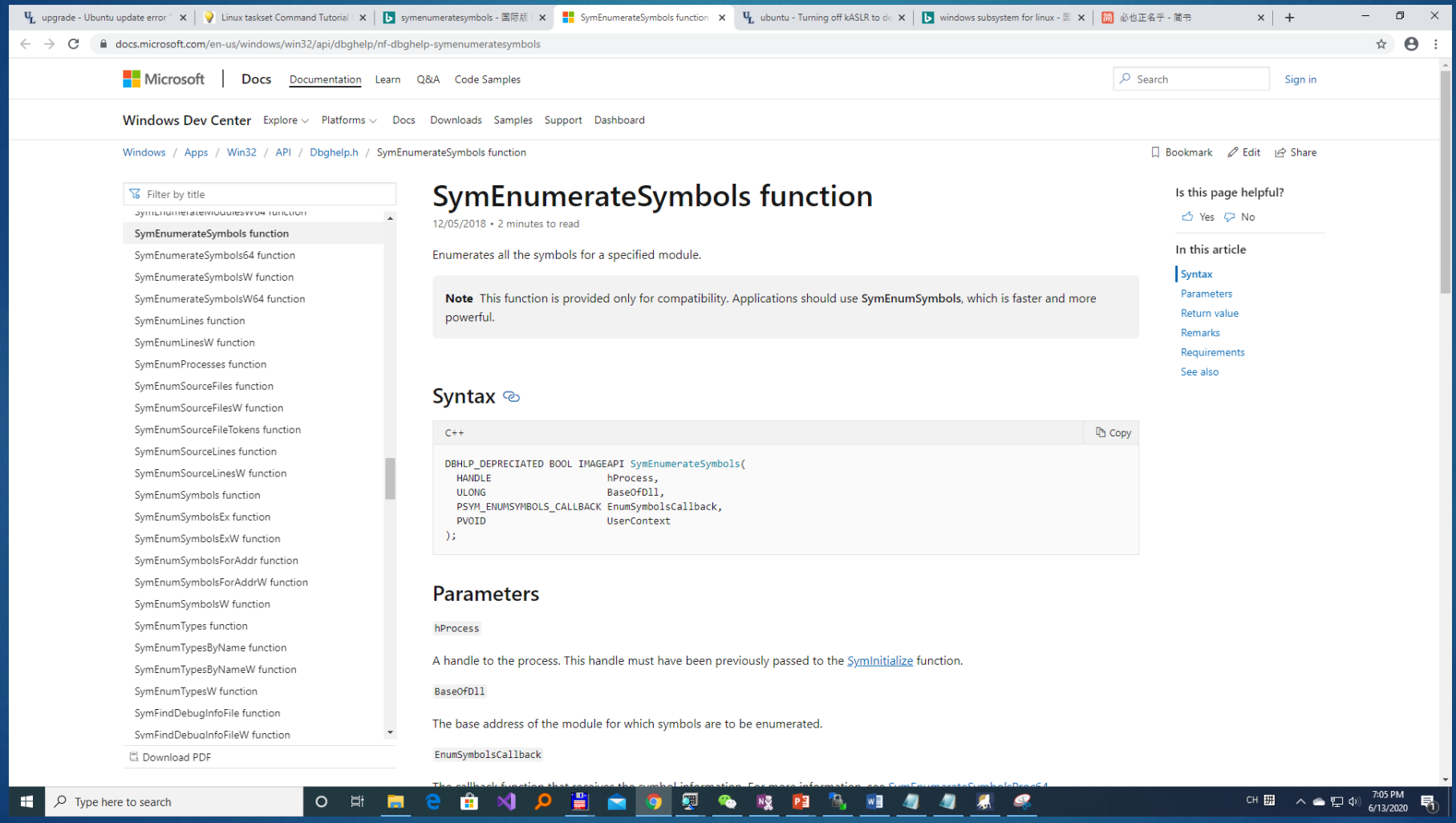

ddebs.ubuntu.com
Index of /
Name
Last modified
Size
Description
clean.conf 01-Apr-2015 15:44 423
dbgsym-release-key.asc 04-Jul-2016 16:10 2.4K
dbgsym-release-key.asc.old 02-Sep-2008 18:28 1.8K
dists/ 31-Jul-2016 12:14 -
pool/ 19-Feb-2010 14:50 -
ubuntu/ 17-Sep-2016 01:02 -
Apache/2.2.22 (Ubuntu) Server at ddebs.ubuntu.com Port 80

安装Ubuntu的符号

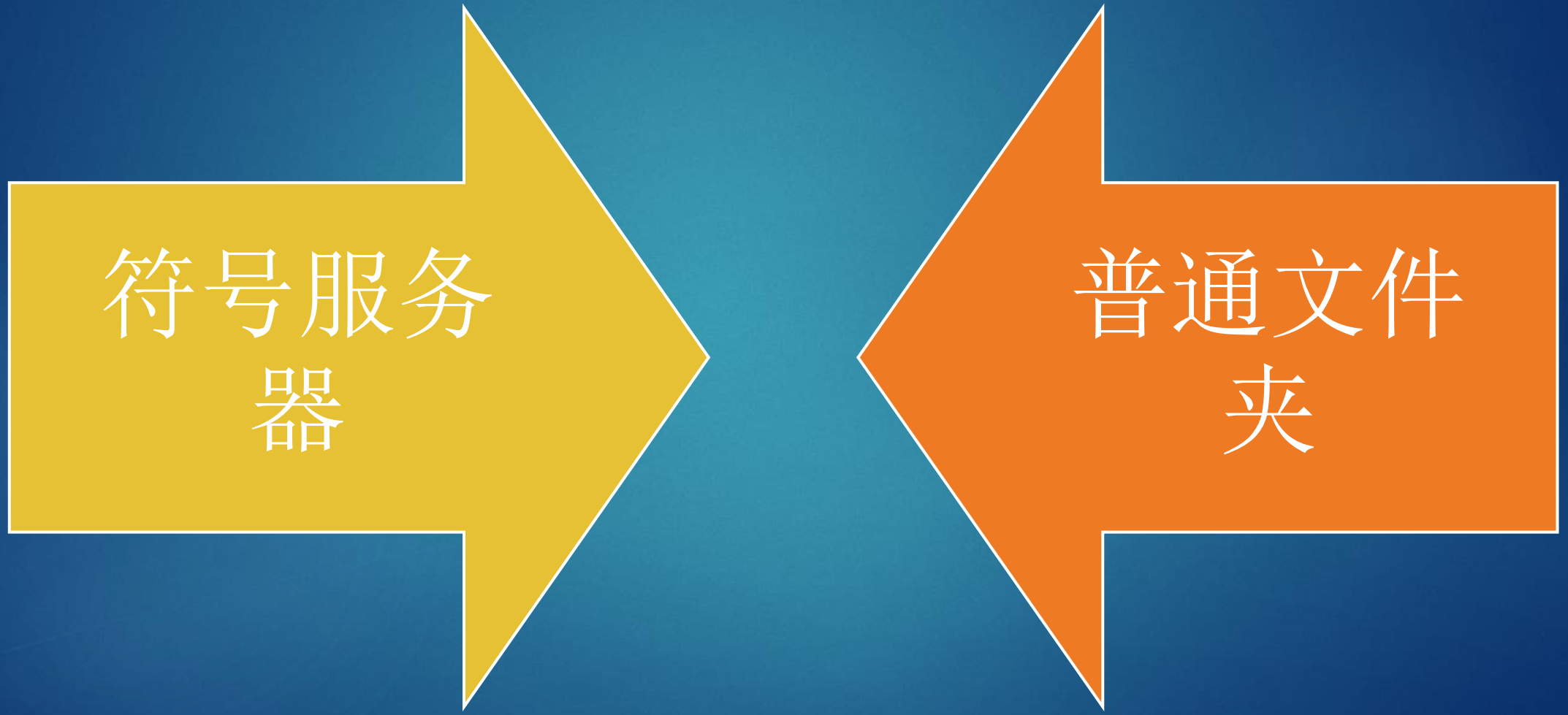
- 设置ddebs
- 下载key
- 下载和安装符号包
- http://advdbg.org/blogs/advdbg_system/articles/7147.aspx



DbgHelp API


upgrade - Ubuntu update error
Linux taskset Command Tutorial
symenumeratesymbols - 国际版
SymEnumerateSymbols function
ubuntu - Turning off kASLR to de
windows subsystem for linux -
必也正名乎 - 简书
docs.microsoft.com/en-us/windows/win32/api/dbghelp/nf-dbghelp-symenumeratesymbols
Microsoft | Docs
Documentation
Learn
Q&A
Code Samples
Search
Sign in
Windows Dev Center
Explore
Platforms
Docs
Downloads
Samples
Support
Dashboard
Windows / Apps / Win32 / API / Dbghelp.h / SymEnumerateSymbols function
Bookmark
Edit
Share
Filter by title
SymEnumerateSymbols function
SymEnumerateSymbols64 function
SymEnumerateSymbolsW function
SymEnumerateSymbolsW64 function
SymEnumLines function
SymEnumLinesW function
SymEnumProcesses function
SymEnumSourceFiles function
SymEnumSourceFilesW function
SymEnumSourceFileTokens function
SymEnumSourceLines function
SymEnumSourceLinesW function
SymEnumSymbols function
SymEnumSymbolsEx function
SymEnumSymbolsExW function
SymEnumSymbolsForAddr function
SymEnumSymbolsForAddrW function
SymEnumSymbolsW function
SymEnumTypes function
SymEnumTypesByName function
SymEnumTypesByNameW function
SymEnumTypesW function
SymFindDebugInfoFile function
SymFindDebugInfoFileW function
Download PDF
SymEnumerateSymbols function
12/05/2018 • 2 minutes to read
Enumerates all the symbols for a specified module.
Note This function is provided only for compatibility. Applications should use SymEnumSymbols, which is faster and more powerful.
Syntax
C++
Copy
DBHLP_DEPRECIATED BOOL IMAGEAPI SymEnumerateSymbols(
HANDLE hProcess,
ULONG BaseOfDll,
PSYM_ENUMSYMBOLS_CALLBACK EnumSymbolsCallback,
PVOID UserContext
);
Parameters
hProcess
A handle to the process. This handle must have been previously passed to the SymInitialize function.
BaseOfDll
The base address of the module for which symbols are to be enumerated.
EnumSymbolsCallback
Is this page helpful?
Yes
No
In this article
Syntax
Parameters
Return value
Remarks
Requirements
See also
Type here to search
CH
7:05 PM
6/13/2020

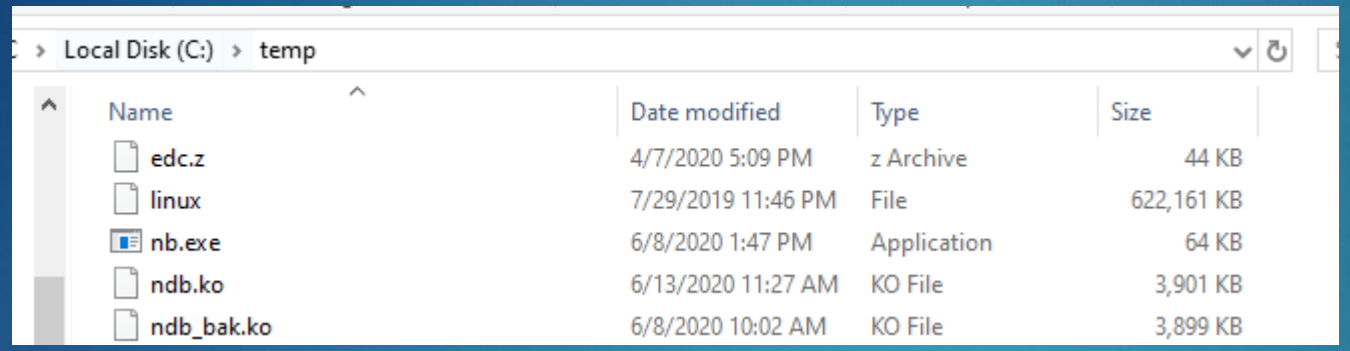
两种方式



设置普通文件夹

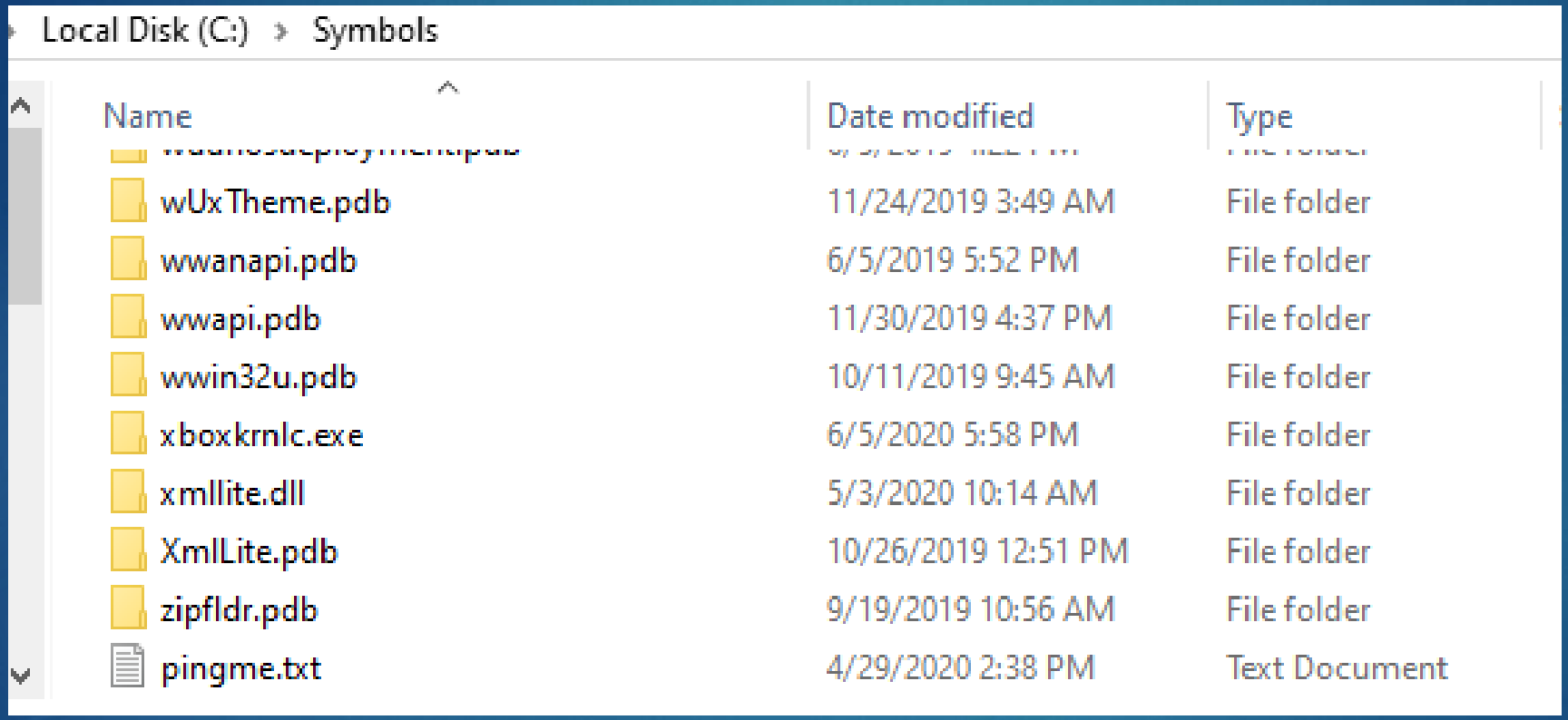
.sympath+ <本地路径>

一个例子



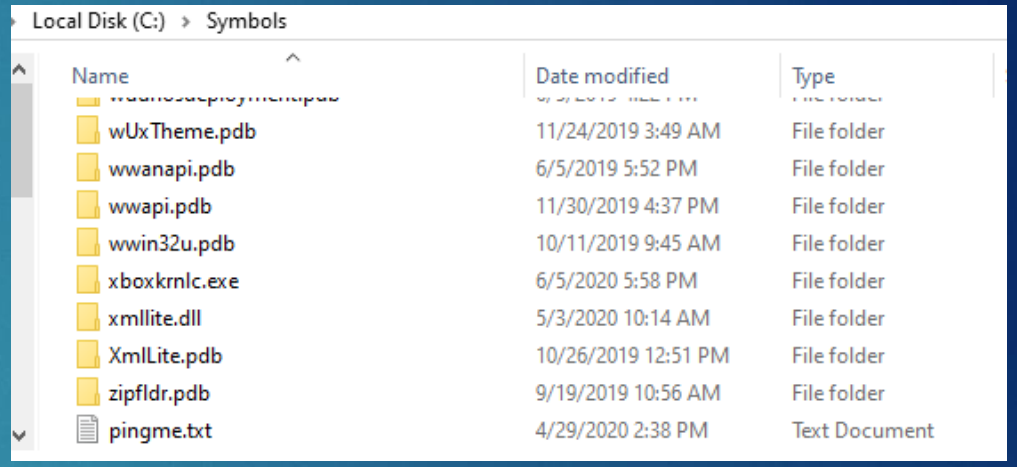
.sympath+ c:\temp

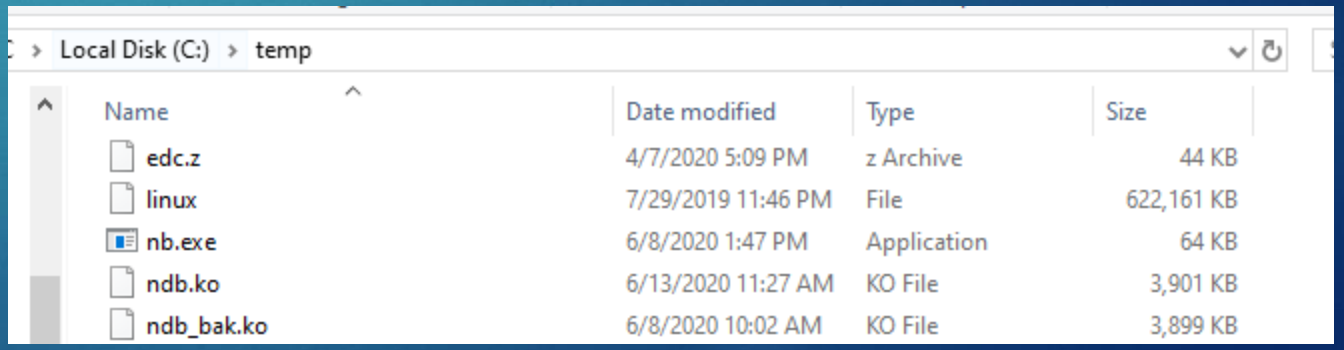
不要放在下游符号库中


Local Disk (C:) › Symbols
Name
Date modified
Type
wUxTheme.pdb
11/24/2019 3:49 AM
File folder
wwanapi.pdb
6/5/2019 5:52 PM
File folder
wwapi.pdb
11/30/2019 4:37 PM
File folder
wwin32u.pdb
10/11/2019 9:45 AM
File folder
xboxkrnlc.exe
6/5/2020 5:58 PM
File folder
xmllite.dll
5/3/2020 10:14 AM
File folder
XmlLite.pdb
10/26/2019 12:51 PM
File folder
zipfldr.pdb
9/19/2019 10:56 AM
File folder
pingme.txt
4/29/2020 2:38 PM
Text Document

不要放在下游符号库中

因为搜索的方式和规则不一样





延迟加载策略

- 是策略
- 不是BUG
- 不用的先不加载
- 加载很慢
- 时间宝贵

```
start end module name
ffeeffee`7ffff000 ffeeffee`809ff000 nt (deferred)
ffffffff`7ffff000 ffffffff`9ffff000 lk (private pdb symbols) c:\temp\linux
ffffffff`c0001000 ffffffff`c0006000 ulpi (deferred)
ffffffff`c0007000 ffffffff`c0012000 autofs4 (deferred)
ffffffff`c001e000 ffffffff`c002a000 video (deferred)
ffffffff`c002b000 ffffffff`c002f000 hid_generic (deferred)
ffffffff`c0030000 ffffffff`c0038000 libahci (deferred)
ffffffff`c003d000 ffffffff`c0042000 dwc3_pci (deferred)
ffffffff`c0047000 ffffffff`c004c000 realtek (deferred)
ffffffff`c0050000 ffffffff`c005a000 ahci (deferred)
ffffffff`c0062000 ffffffff`c0076000 r8169 (deferred)
ffffffff`c007e000 ffffffff`c00a3000 psmouse (deferred)
ffffffff`c00a4000 ffffffff`c00ae000 x_tables (deferred)
ffffffff`c00b0000 ffffffff`c00bd000 udc_core (deferred)
ffffffff`c00c5000 ffffffff`c00e4000 dwc3 (deferred)
ffffffff`c00e5000 ffffffff`c00f2000 usbhid (deferred)
ffffffff`c00f5000 ffffffff`c00fa000 lp (deferred)
ffffffff`c00fb000 ffffffff`c0102000 wmi (deferred)
ffffffff`c0103000 ffffffff`c0122000 hid (deferred)
ffffffff`c0123000 ffffffff`c012b000 ip_tables (deferred)
ffffffff`c012c000 ffffffff`c0139000 parport (deferred)
ffffffff`c0142000 ffffffff`c0148000 ppdev (deferred)
ffffffff`c014d000 ffffffff`c0156000 parport_pc (deferred)
ffffffff`c016f000 ffffffff`c0173000 ndb T (private pdb symbols) c:\temp\ndb.ko
```

DWARF的调整

- 为了节约空间，使用了很多字节形式的编码
- 为了支持栈回溯，支持脚本
- 解析起来很费时间

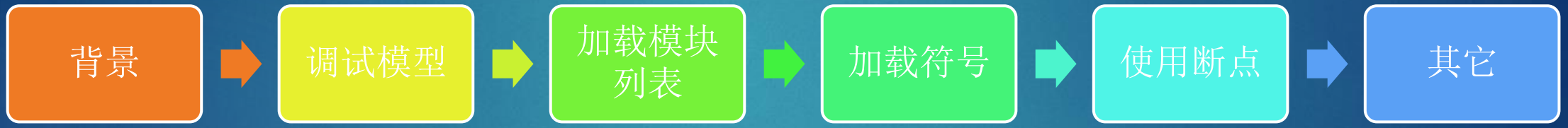
使用x命令检索符号

```
x ndb!*proc_*
*** WARNING: Unable to verify timestamp for ndb.ko
<unavailable> proc_handler = <value unavailable>
<unavailable> proc_dir_entry = <value unavailable>
Type information missing error for proc_ndb_entry
ffffffff`c0170040 ndb!proc_ndb_fops = const file_operations
ffffffff`c01703f0 ndb!proc_ndb_write (struct file *, const char *, size_t, loff_t *)
ffffffff`c0170040 ndb!proc_ndb_read (struct file *, char *, size_t, loff_t *)
```

关于内核的符号

- 重名为linux
- 放在一个普通目录
- .sympathy+ <这个目录>
- X lk!vfs_rea* 触发

- 不要执行x lk!*，可能要等很多分钟


背景
调试模型
加载模块列表
加载符号
使用断点
其它

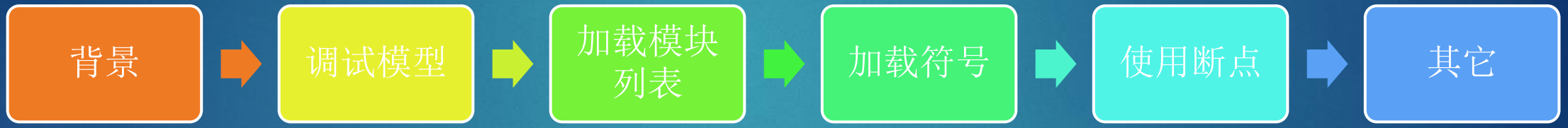
软件断点

- Bp <函数名>|<地址>

- 目前工作的很稳定
- 对NT目标不稳定的主要原因是Patch Guard

硬件断点

- Ba r/w<宽度> <变量地址 | 代码地址>
- 目前只有设置断点的CPU再访问时命中
- > taskset 1 echo /proc/ndb


背景
调试模型
加载模块列表
加载符号
使用断点
其它

列进程

- !ps

- 工作的很稳定
- 我们会对其增强，显示更多属性，并支持筛选和定制

```
task_struct:0xffff888161971600 pid: 329 comm:loop1
state 1 flags:0x308040 stack:0xffffc90000a40000

task_struct:0xffff888161970000 pid: 343 comm:loop2
state 1 flags:0x308040 stack:0xffffc90000b3c000

task_struct:0xffff8881619c8000 pid: 354 comm:loop3
state 1 flags:0x308040 stack:0xffffc90000a20000

task_struct:0xffff8881619cd800 pid: 356 comm:loop4
state 1 flags:0x308040 stack:0xffffc90000bf4000

task_struct:0xffff8881619cc200 pid: 359 comm:loop5
state 1 flags:0x308040 stack:0xffffc90000d64000

task_struct:0xffff888160959600 pid: 362 comm:loop6
state 1 flags:0x308040 stack:0xffffc90000b84000

task_struct:0xffff888160869600 pid: 367 comm:loop7
state 1 flags:0x308040 stack:0xffffc90000bac000

task_struct:0xffff8881618ec200 pid: 375 comm:loop8
state 1 flags:0x308040 stack:0xffffc900009a8000

task_struct:0xffff8881618ed800 pid: 377 comm:loop9
state 1 flags:0x308040 stack:0xffffc90000b24000
```

千头万绪，终归一理

宋·朱熹 (1130-1200)

所谓致知在格物者，言欲致吾之知，在**即物**而穷其理也。盖人心之灵莫不有知，而天下之物莫不有理，惟于理有未穷，故其知有不尽也。是以《大学》始教，必使学者即凡天下之物，莫不因其已知之理而益穷之，以求至乎其极。至于用力之久，而一旦豁然**贯通**焉，则众物之表里精粗无不到，而吾心之全体大用无不明矣。此谓物格，此谓知之至也。

问答

http://www.nanocode.cn

http://advdbg.org/gdk