openEuler

KDUMP基本原理、使用及案例介绍

陈洲

目录

openEuler

kdump介绍

- 生产内核

  production kernel，first kernel，primary kernel，正常工作的内核

- 捕获内核

  capture kernel，second kernel，crash kernel，用于收集崩溃转储信息的内核

- kexec

  快速重启机制，跳过bios

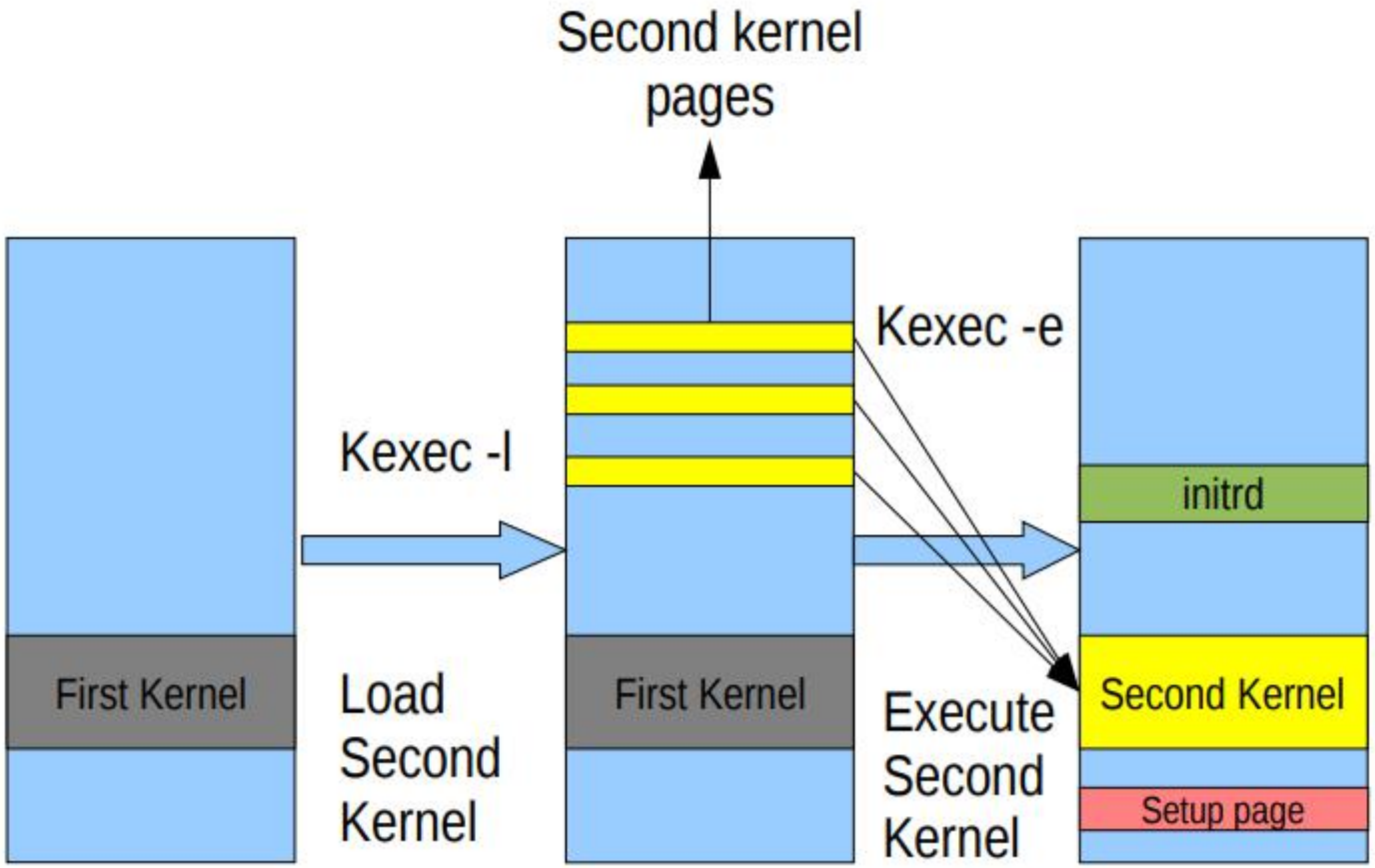
- kdump

  一种基于kexec的内核崩溃转储机制

- 架构支持

  x86, x86_64, arm, arm64, ppc, s390, sh

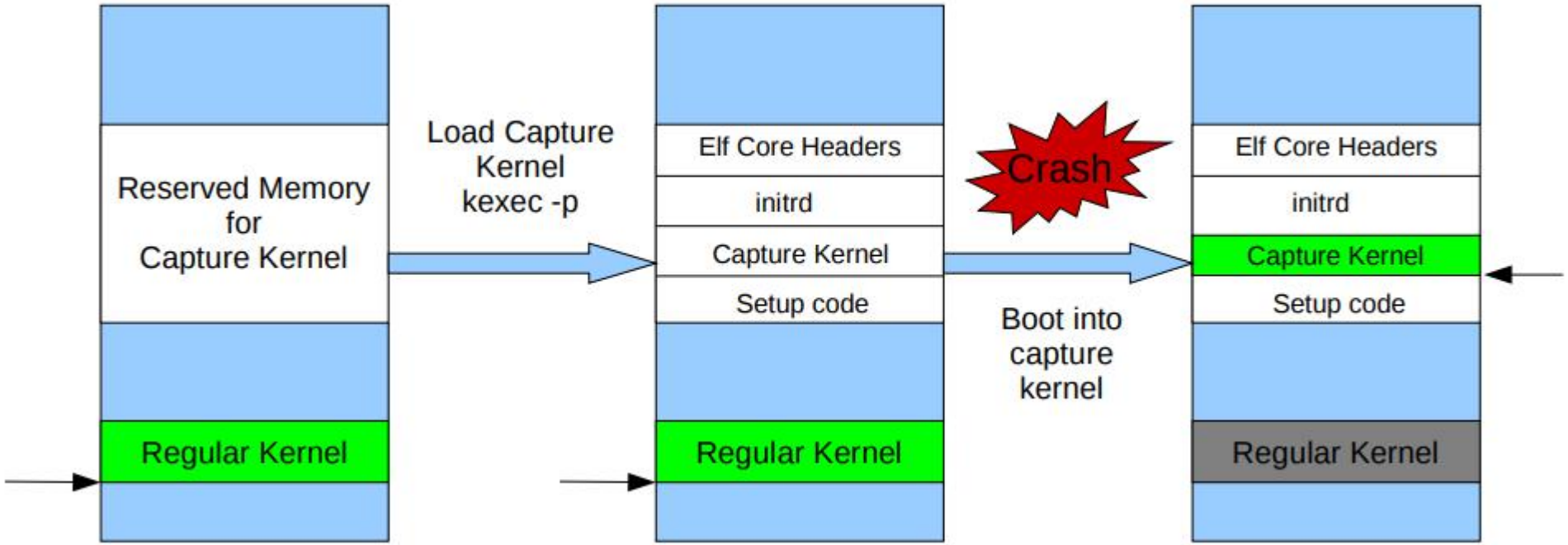
kexec原理



From Vivek Goyal

openEuler

kdump原理



From Vivek Goyal

openEuler

用户态：kexec-tools

- kexec-tools
  - 系统调用：kexec_load, reboot
  - kexec
    a. 加载第二个内核，kexec –l, kexec_load
    b. 启动到加载的内核，kexec –e, reboot
  - Kdump
    a. 加载捕获内核，kexec –p, kexec_load
    b. 系统crash，启动到捕获内核

内核态：kexec_load/reboot

- **kexec_load**

SYSCALL_DEFINE4(kexec_load, unsigned long, entry, unsigned long, nr_segments,
*struct kexec_segment __user *, segments, unsigned long, flags)*

  - **kexec_segment**
    - **kernel, initramfs, dtb, …**
  - **flags**
    - **KEXEC_ON_CRASH**
  - **arm64_relocate_new_kernel**
    - **将第二个内核放到指定位置并启动**
  - **flush reloc_code**

- **reboot**

reboot(LINUX_REBOOT_CMD_KEXEC)

openEuler

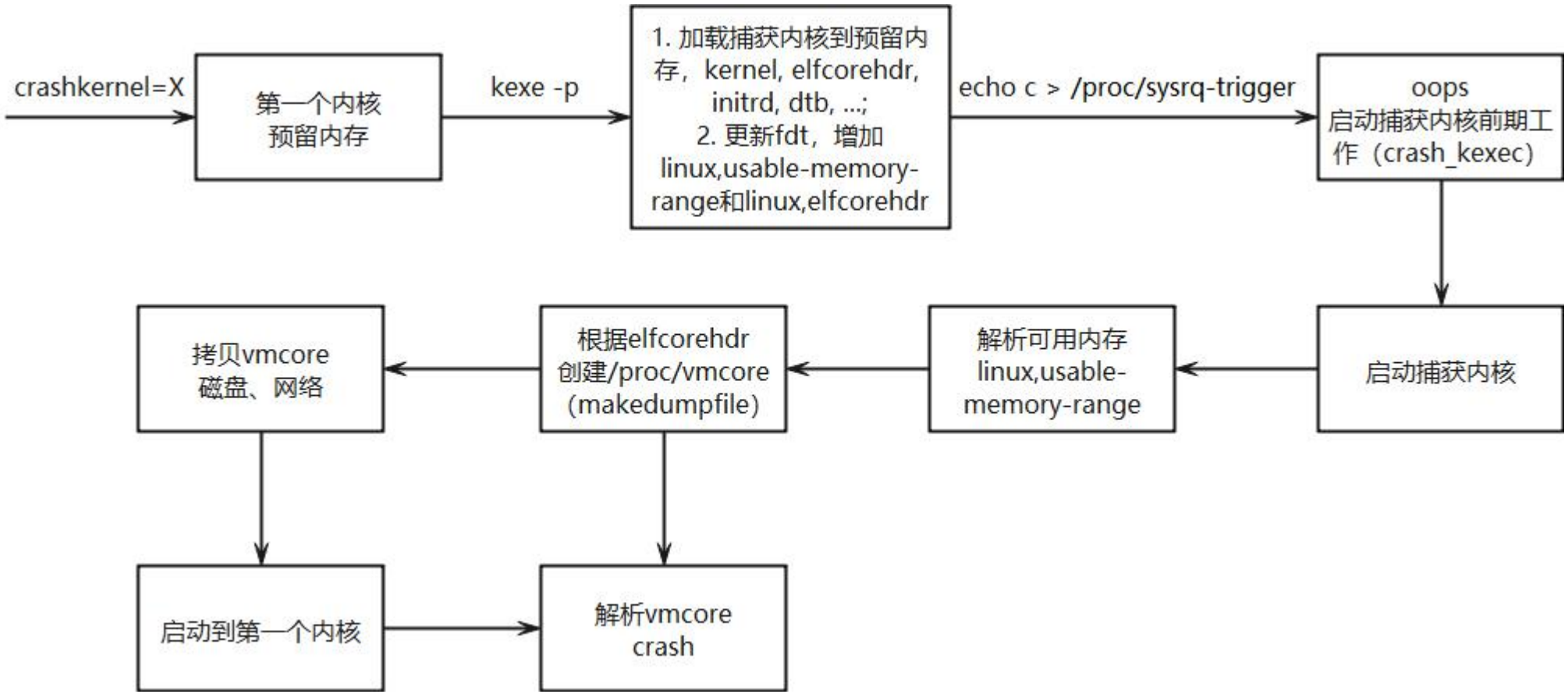
内核态：config

- **kexec**
  - CONFIG_KEXEC
  - CONFIG_KEXEC_CORE

- **kdump**
  - CONFIG_CRASH_DUMP
  - CONFIG_CRASH_CORE
  - CONFIG_PROC_VMCORE
    - /proc/vmcore
  - CONFIG_DEBUG_INFO
    - Debug info

openEuler

kdump执行流程



openEuler

预留内存

- crashkernel=size[KMG][@offset[KMG]]
  - 最常使用，offset通常不指定
- crashkernel=range1:size1[,range2:size2,...][@offset]
  - 大小取决于系统可用内存
  - crashkernel=512M-2G:64M,2G-:128M
- crashkernel=size[KMG],high
  - X86_64
  - 在4G以上预留内存
- crashkernel=size[KMG],low
  - X86_64
  - 在4G以下预留内存

openEuler

预留内存

- 分配的内存区域
  - /proc/iomem，dmesg
    - 物理内存布局
    - Crash kernel 条目

```
# cat /proc/iomem | grep -i crash
24000000-33ffffff : Crash kernel
```

- 分配内存大小查看

```
# cat /sys/kernel/kexec_crash_size
536870912
```

openEuler

预留内存

- 预留内存大小评估
  - Kernel, initrd, devices
  - 捕获内核启动cpu个数
  - 通常256M，server 512M/1G

- 最大可预留内存（4G以下）评估

  # cat /proc/iomem | grep "System RAM“
  00100000-361c3017 : System RAM -- 800M (预留内存大小 + 额外内存)

openEuler

加载捕获内核

- 使用
  - kexec -d -p -l Image --append= “... irqpoll nr_cpus=1 reset_devices" --initrd=./initrd
  - -d

    debug
  - --initrd, --reuseinitrd

    指定initramfs
  - --dtb

    指定dtb
- 捕获内核是否加载成功

  # cat /sys/kernel/kexec_crash_loaded

  1

openEuler

crash/oops

- __show_regs, dump_backtrace

- crash_kexec
  a. crash_setup_regs
     准备panic kernel regs
  b. crash_save_vmcoreinfo
     更新vmcoreinfo
  c. machine_crash_shutdown
     crash_smp_send_stop
     crash_save_cpu，保存panic cpu信息
     清理中断
  d. machine_kexec
     flush segments
     “Bye!”
     cpu_soft_restart

openEuler

/proc/vmcore

- ELF格式
- VMCOERINFO
  - 包含内核的各种信息
    - structure size
    - page size
    - symbol values
    - field offsets
    - etc
  - 用户态工具用来分析内存布局
    - makedumpfile, crash
- ELFCOREHDR

描述panic内核的布局

openEuler

makedumpfile

- **减小转储的内存镜像的体积**
  - **页面过滤**
    - 0页
    - 缓存
    - 用户数据
    - 空闲页
  - **页面压缩**

- **使用**
  - **makedumpfile -c -d 31 --message-level 31 /proc/vmcore vmcore**
    - -D，打印调试信息
    - -d，过滤级别

openEuler

kdump：以openEuler为例

- **Kdump服务**
  - **提供一些脚本**
  - **systemctl start/stop/status kdump**

- **配置文件**
  - **/etc/kdump.conf**

    path /var/crash

    keep_old_dumps -1
  - **/etc/sysconfig/kdump**

    KDUMP_COMMANDLINE_APPEND

openEuler

kdump使用案例

- 触发系统oops

  # echo c > /proc/sysrq-trigger

- 解析

  # crash vmlinux vmcore

  crash> help

  crash> ?

```
crash> bt
PID: 19888  TASK: ffff967f9e43bd80  CPU: 39  COMMAND: "bas
 #0 [ffffb7f11ba77b78] machine_kexec at ffffffff8745a49e
 #1 [ffffb7f11ba77bd0] __crash_kexec at ffffffff875592f1
 #2 [ffffb7f11ba77c90] panic at ffffffff874b2e4b
 #3 [ffffb7f11ba77d18] oops_end at ffffffff87421d0c
 #4 [ffffb7f11ba77d48] console_unlock at ffffffff87519490
 #5 [ffffb7f11ba77dc0] page_fault at ffffffff87e0122e
    [exception RIP: sysrq_handle_crash+18]
    RIP: ffffffff8791f9e2  RSP: ffffb7f11ba77e70  RFLAGS:
    RAX: ffffffff8791f9d0  RBX: 0000000000000063  RCX: 000
    RDX: 0000000000000000  RSI: ffff967fc0b56a08  RDI: 000
    RBP: ffffffff8873a120   R8: 0000000000000a79   R9: 000
    R10: 0000000000000001  R11: ffffffff88e41b2e  R12: 000
    R13: 0000000000000000  R14: 00005601baa4b1f0  R15: 000
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
 #6 [ffffb7f11ba77e70] __handle_sysrq at ffffffff879200e4
 #7 [ffffb7f11ba77e98] write_sysrq_trigger at ffffffff8792
 #8 [ffffb7f11ba77eb0] proc_reg_write at ffffffff8773f7bc
 #9 [ffffb7f11ba77ec8] vfs_write at ffffffff876c3dad
#10 [ffffb7f11ba77ef8] ksys_write at ffffffff876c4032
#11 [ffffb7f11ba77f38] do_syscall_64 at ffffffff8740419b
#12 [ffffb7f11ba77f50] entry_SYSCALL_64_after_hwframe at f
```

kdump使用案例

- 定位分析
  - crash> dis ffffffff8791f9e2

    0xffffffff8791f9e2 <sysrq_handle_crash+18>: movb $0x1,0x0

  - crash> sym ffffffff8791f9e2

    ffffffff8791f9e2 (t) sysrq_handle_crash+18 /home/chenzhou/linux/drivers/tty/sysrq.c: 147

openEuler

kdump使用问题

- **没有生成vmcore**
  - 按照kdump执行流程，确定问题来自哪个阶段
- **预留内存失败**
  - Bios上报的物理内存零散
  - 预留内存过大
- **加载内核失败**
  - 是否预留内存，crashkernel
  - 预留内存失败
  - 预留内存成功，kexec –d -p，查看失败具体原因

openEuler

kdump使用问题

- **第二个内核启动失败**
  - **Bye!**

    配置串口，earlycon/console
  - **oom**

    预留内存太小
  - **驱动初始化失败**

openEuler

kdump使用问题

- **makedumpfile**

  -D，debug信息

- **crash**

  vmlinux vmcore需要对应

- **用户态工具问题**

  kernel、kexec、makedumpfile、crash匹配问题

  更新到最新的用户态工具

openEuler

arm64 kdump支持

- **crashkernel=X**

  < 4G连续可用内存不足，预留失败

- **crashkernel=X,high**

  没有<4G内存可用，dma32 devices初始化失败，内核启动失败

  →

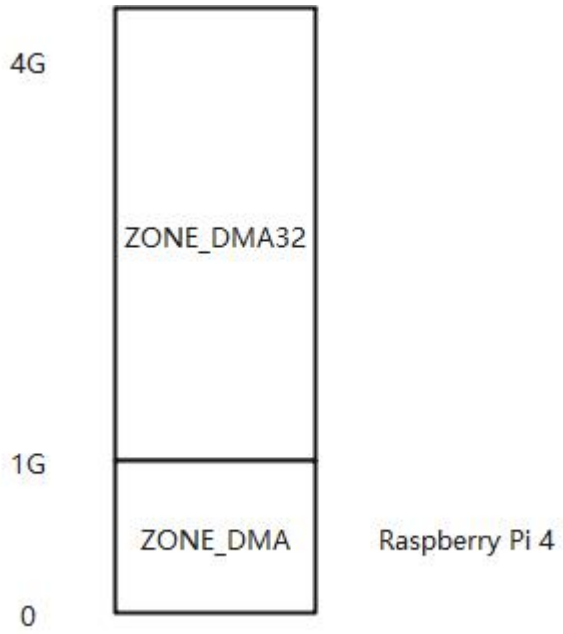
- **支持4G以上预留内存**

openEuler

arm64 kdump支持

- ZONE_DMA ZONE_DMA32共存
  - 1a8e1cef7603 ("arm64: use both ZONE_DMA and ZONE_DMA32")
    - ZONE_DMA

      低1G内存
    - ZONE_DMA32
  - 预留内存位于ZONE_DMA32，启动失败



openEuler

支持4G以上预留内存

- **预留多段crash kernel**
  - **crashkernel=X,high**

    Crash kernel (low)，兼容现有的kexec-tools

    Crash kernel
  - **crashkernel=X**
  - **crashkernel=X,high crashkernel=Y,low**
- **kexec-tools**
  - **扫描Crash kernel , Crash kernel (low)**
  - **修改dtb字段**

    linux,usable-memory-range = <BASE1 SIZE1 [BASE2 SIZE2]>
    - 兼容老版本内核，现存用户态工具
- **捕获内核解析可用内存**

[PATCH v14 00/11] support reserving crashkernel above 4G on arm64 kdump
https://lkml.org/lkml/2021/1/30/53

openEuler

ZONE_DMA ZONE_DMA32共存

➢ Linux 主线

- ➢ **patchset "arm64: Default to 32-bit wide ZONE_DMA", v5.11-rc1**
  - **动态ZONE_DMA**
    - 默认4G
    - 对于Raspberry Pi 4，覆盖1G
- ➢ **d78050ee3544 arm64: Remove arm64_dma32_phys_limit and its uses v5.11-rc4**
  - **KDUMP正常工作**

➢ openEuler 5.10

- ➢ **CONFIG_ZONE_DMA=y**

  在 ZONE_DMA区域预留
- ➢ **CONFIG_ZONE_DMA=n**

  在ZONE_DMA32区域预留

openEuler

邮件列表/文档/reference

- **邮件列表**
  - kexec@lists.infradead.org
- **git仓库**
  - https://github.com/horms/kexec-tools.git
  - https://github.com/bhupesh-sharma/makedumpfile/
  - https://github.com/crash-utility/crash.git
- **ref**
  - http://people.redhat.com/vgoyal/papers-presentations/redhat-summit-2008/vivek-redhat-summit-2008-kdump-presentation.pdf

openEuler

openEuler

openEuler 是一个开放的社区，鼓励、支持和期待使用者、爱好者、开发者及技术专家的参与。openEuler 不是做商用发行版，目标更不是替换各个公司的 OS 系统，而是让各行各业信任 openEuler，基于 openEuler 及其衍生版本构建、发布自己的 OS 系统。

openEuler 的愿景是：走进千行百业，千家万户，做信息产业的基石。

理念：
共建、共享
开源、开放
创新、引领

内核是操作系统的基础，终有一天，openEuler Kernel 将支撑着各行各业运行的 openEuler 系统及 openEuler 的衍生系统。期待您的参与能让 openEuler 运行的更好，让各行各业受益。

管理员	公众号
	

Thank you

openEuler