Gentoo Linux support

CONFIG_GENTOO_LINUX

选"Y"后,将会自动选中那些在 Gentoo 环境中必须开启的内核选项,以避免用户遗漏某些必要的选项,减轻一些用户配置内核的难度.建议选"Y".

Linux dynamic and persistent device naming (userspace devfs) support

CONFIG_GENTOO_LINUX_UDEV

目前此项的作用仅是开启 CONFIG_TMPFS 和 CONFIG_DEVTMPFS 及其所依赖的选项.CONFIG_TMPFS 是为了在"/dev/shm","/run","/sys/fs/cgroup"三个目录中挂载 tmpfs 文件系统,CONFIG_DEVTMPFS 是为了在"/dev"目录挂载 devtmpfs 文件系统.建议选"Y".

Select options required by Portage features

CONFIG_GENTOO_LINUX_PORTAGE

自动选中各种 Portage 特性(FEATURES="cgroup ipc-sandbox network-sandbox")所依赖的内核选项.由于这些 FEATURES 即将变为默认开启,所以建议选"Y".

Support for init systems, system and service managers

"init"系统(系统与服务管理器)."init"是内核启动的第一个用户空间程序(PID=1),也是所有用户态进程的"大总管"([提示]所有内核态进程的大总管是 PID=2 的[kthreadd]).

OpenRC, runit and other script based systems and managers

CONFIG_GENTOO_LINUX_INIT_SCRIPT

OpenRC 是 Gentoo 传统的"init"系统,使用基于 SysVinit 的传统启动脚本技术.选"Y"后,所有 OpenRC 所必需的内核选项(目前仅有 CONFIG_BINFMT_SCRIPT)都会被自动选中.不确定的选"Y".

systemd

CONFIG_GENTOO_LINUX_INIT_SYSTEMD

尽管倍受争议,但是 systemd 确实是目前风头最劲的"init"系统,大有在 Linux 世界一统江湖的霸气和潜力.仅在你打算从 OpenRC 迁移到 systemd 时选"Y".选"Y"后,内核中所有被 systemd 依赖的选项都将被自动选中,包括 systemd 建议的(而非必须的)选项,总计约 20 项.systemd 官方的 README 文件也列出了必须/推荐/可选的内核选项.

64-bit kernel

CONFIG_64BIT

编译 64 位内核.本文仅讲述 x86_64(AMD64)平台的内核编译,所以这个是必选项.

General setup

常规设置

Cross-compiler tool prefix

CONFIG_CROSS_COMPILE

交叉编译工具前缀(比如"arm-linux-"相当于使用"make CROSS_COMPILE=arm-linux-"进行编译).除非你想配置后默认自动进行交叉编译,否则不要使用此选项.

Compile also drivers which will not load

CONFIG_COMPILE_TEST

显示专属于其他平台(非 x86 平台)的驱动选项(需要交叉编译),仅供驱动开发者使用,普通的发行版制作者应该选"N".

Local version - append to kernel release

CONFIG_LOCALVERSION

在内核版本后面加上自定义的版本字符串(最大 64 字符),可以用"uname -a"命令看到

Automatically append version information to the version string

CONFIG_LOCALVERSION_AUTO

自动在版本字符串(CONFIG_LOCALVERSION)后面添加版本信息(类似"-gxxxxxxxx"格式),需要有 perl 以及 git 仓库支持

Kernel compression mode

内核镜像的压缩格式,可选 Gzip/Bzip2/LZMA/XZ/LZO 格式之一,推荐使用 XZ 格式.你的系统中需要有相应的压缩工具.

Default hostname

CONFIG_DEFAULT_HOSTNAME

设置默认主机名,默认值是"(none)".用户可以随后使用系统调用 sethostname()来修改主机名.

Support for paging of anonymous memory (swap)

CONFIG_SWAP

使用交换分区或者交换文件来做为虚拟内存

System V IPC

CONFIG_SYSVIPC

System V 进程间通信(IPC)支持,用于进程间同步和交换数据,许多程序需要这个功能.选"Y",除非你确实知道自己在做什么

POSIX Message Queues

CONFIG_POSIX_MQUEUE

POSIX 消息队列是 POSIX IPC 的一部分,如果你想编译和运行那些使用"mq_*"系统调用的程序(比如为 Solaris 开发的程序),或者需要使用 Docker 容器,就必须开启此选项.POSIX 消息队列

可以作为"mqueue"文件系统挂载以方便用户对队列进行操作.不确定的选"Y".

open by fhandle syscalls

CONFIG_FHANDLE

用户程序可以使用句柄(而非文件名)来追踪文件(使用open_by_handle_at(2)/name_to_handle_at(2)系统调用),即使某文件被重命名,用户程序依然可定位那个文件.此特性有助于实现用户空间文件服务器(userspace file server).建议选"Y",因为 systemd 和 udev 依赖于它.

uselib syscall

CONFIG_USELIB

启用老旧的 uselib()系统接口支持,仅在你需要使用基于libc5 的古董级程序时才需要,不确定的选"N".

Auditing support

CONFIG_AUDIT

内核审计(跟踪每个进程的活动情况)支持,某些安全相关的内核子系统(例如 SELinux)需要它.但是它会与 systemd 冲突,所以在使用 systemd 的系统上必须关闭.

Enable system-call auditing support

CONFIG_AUDITSYSCALL

对系统调用进行审计.既可独立使用,也可被其他内核子系统(例如 SELinux)使用.

Make audit loginuid immutable

CONFIG_AUDIT_LOGINUID_IMMUTABLE

审计时使用固定的 loginuid.在使用 systemd 之类的系统上应该开启(login 服务由 init 进程负责重启),在使用 SysVinit 或 Upstart 之类的系统上应该关闭(login 服务由系统管理员手动重启).OpenRC 就是一个基于 SysVinit 的系统.

IRQ subsystem

IRQ(中断请求)子系统

Expose hardware/virtual IRQ mapping via debugfs

CONFIG_IRQ_DOMAIN_DEBUG

通过 debugfs 中的 irq_domain_mapping 文件向用户显示硬件 IRQ 号/Linux IRQ 号之间的对应关系.仅用于开发调试.

Support sparse irq numbering

CONFIG_SPARSE_IRQ

稀疏 IRQ 号支持.它允许在小型设备上(例如嵌入式设备)定义一个很高的 CONFIG_NR_CPUS 值,但仍然不希望占用太多内核"memory footprint"(一段可以被操作或被管理的内存区域)的场合.稀疏 IRQ 也更适合 NUMA 平台,因为它以一种对 NUMA 更友好的方式分发中断描述符.不确定的选"N".

Timers subsystem

Linux 内核时钟子系统

Timer tick handling

内核时钟滴答处理程序,更多信息可以参考内核源码树下的"Documentation/timers/NO_HZ.txt"文件

Periodic timer ticks (constant rate, no dynticks)

CONFIG_HZ_PERIODIC

无论 CPU 是否需要,都强制按照固定频率不断触发时钟中断.这是最耗电的方式,不推荐使用

Idle dynticks system (tickless idle)

CONFIG_NO_HZ_IDLE

CPU 在空闲状态时不产生不必要的时钟中断,以使处理器能够在较低能耗状态下运行以节约电力,适合于大多数场合

Full dynticks system (tickless)

CONFIG_NO_HZ_FULL

完全无滴嗒:即使 CPU 在忙碌状态也尽可能关闭所有时钟中断,适用于 CPU 在同一时间仅运行一个任务,或者用户空间程序极少与内核交互的场合.即使开启此选项,也需要额外设置"nohz_full=?"内核命令行参数才能真正生效.

Full dynticks system on all CPUs by default

CONFIG_NO_HZ_FULL_ALL

即使没有设置"nohz_full"引导参数,也默认对所有 CPU(boot CPU 除外)开启完全无滴答特性.

Old Idle dynticks config

CONFIG_NO_HZ

等价于 CONFIG_NO_HZ_IDLE,临时用来兼容老版本内核选项,未来会被删除.

High Resolution Timer Support

CONFIG_HIGH_RES_TIMERS

高精度定时器(hrtimer)是从 2.6.16 开始引入,用于取代传统 timer wheel(基于 jiffies 定时器)的时钟子系统.可以降低与内核其他模块的耦合性,还可以提供比 1 毫秒更高的精度(因为它可以读取 HPET/TSC 等新型硬件时钟源),可以更好的支持音视频等对时间精度要求较高的应用.建议选"Y".[提示]这里说的"定时器"是指"软件定时器",而不是主板或 CPU 上集成的硬件时钟发生器(ACPI PM Timer/HPET Timer/TSC Timer).

CPU/Task time and stats accounting

CPU/进程的时间及状态统计

Cputime accounting

CPU 时间统计方式

Simple tick based cputime accounting

CONFIG_TICK_CPU_ACCOUNTING

简单的基于滴答的统计,适用于大多数场合

Deterministic task and CPU time accounting

CONFIG_VIRT_CPU_ACCOUNTING_NATIVE

通过读取 CPU 计数器进行统计,可以提供更精确的统计,但是对性能有一些不利影响.

Full dynticks CPU time accounting

CONFIG_VIRT_CPU_ACCOUNTING_GEN

利用上下文跟踪子系统,通过观察每一个内核与用户空间的边界进行统计.该选项对性能有显著的不良影响,目前仅用于完全无滴答子系统(CONFIG_NO_HZ_FULL)的调试

Fine granularity task level IRQ time accounting

CONFIG_IRQ_TIME_ACCOUNTING

通过读取 TSC 时间戳进行统计,这是统计进程 IRQ 时间的更细粒度的统计方式,但对性能有些不良影响(特别是在 RDTSC 指令速度较慢的 CPU 上).

BSD Process Accounting

CONFIG_BSD_PROCESS_ACCT

BSD 进程记账支持.用户空间程序可以要求内核将进程的统计信息写入一个指定的文件,主要包括进程的创建时间/创建者/内存占用等信息.不确定的选"N".

BSD Process Accounting version 3 file format

CONFIG_BSD_PROCESS_ACCT_V3

使用新的 v3 版文件格式,可以包含每个进程的 PID 和其父进程的 PID,但是不兼容老版本的文件格式.比如 GNU Accounting Utilities 这样的工具可以识别 v3 格式

Export task/process statistics through netlink

CONFIG_TASKSTATS

通过 netlink 接口向用户空间导出进程的统计信息,与 BSD Process Accounting 的不同之处在于这些统计信息在整个进程生存期都是可用的.

Enable per-task delay accounting

CONFIG_TASK_DELAY_ACCT

在统计信息中包含进程等候系统资源(cpu,IO 同步,内存交换等)所花费的时间

Enable extended accounting over taskstats

CONFIG_TASK_XACCT

在统计信息中包含进程的更多扩展信息.不确定的选"N".

Enable per-task storage I/O accounting

CONFIG_TASK_IO_ACCOUNTING

在统计信息中包含进程在存储设备上的 I/O 字节数.

RCU Subsystem

RCU(Read-Copy Update)子系统.它允许程序查看到正在被修改/更新的文件.在读多写少的情况下,这是一个高性能的锁机制,对于被 RCU 保护的共享数据结构,读者不需要获得任何锁就可以访问它(速度非常快),但写者在访问它时首先拷贝一个副本,然后对副本进行修改,最后使用一个回调机制在适当的时机把指向原来数据的指针重新指向新的被修改的数据,速度非常慢.RCU 只适用于读多写少的情况:如网络路由表的查询更新,设备状态表的维护,数据结构的延迟释放以及多径 I/O 设备的维护等.

RCU Implementation

RCU 的实现方式

Tree-based hierarchical RCU

CONFIG_TREE_RCU

基于树型分层结构的实现.最适用于多 CPU 的非实时系统.

Preemptible tree-based hierarchical RCU

CONFIG_TREE_PREEMPT_RCU

抢占式基于树型分层结构的实现.最适用于那些要求快速响应的多 CPU 实时系统.

UP-only small-memory-footprint RCU

CONFIG_TINY_RCU

最简单的实现,能够大幅降低 RCU 系统的内存占用.最适用于单 CPU 的非实时系统.

Preemptible UP-only small-memory-footprint RCU

CONFIG_TINY_PREEMPT_RCU

抢占式简单实现,能够大幅降低 RCU 系统的内存占用.最适用于那些要求快速响应的单 CPU 实时系统.

Consider userspace as in RCU extended quiescent state

CONFIG_RCU_USER_QS

在内核和用户边界设置钩子函数,将运行在用户态的 CPU 从全局 RCU 状态机制中移除,这样就不会在 RCU 系统中维护此 CPU 的时钟滴答.除非你想要帮助开发 CONFIG_NO_HZ_FULL 模块,否则不要打开此选项,而且它还会对性能有不利影响.

Force context tracking

CONFIG_CONTEXT_TRACKING_FORCE

默认在内核和用户边界进行探测(上下文跟踪),以便测试依赖于此特性的各种功能(比如用户空间的 RCU extended quiescent state),这个特性目前仅用于调试目的,未来也许会用于为 CONFIG_NO_HZ_FULL 模块提供支持

Tree-based hierarchical RCU fanout value

CONFIG_RCU_FANOUT

这个选项控制着树形 RCU 层次结构的端点数(fanout),以允许 RCU 子系统在拥有海量 CPU 的系统上高效工作.这个值必须至少等于 CONFIG_NR_CPUS 的 1/4 次方(4 次根号).生产系统上应该使用默认值(64).仅在你想调试 RCU 子系统时才需要减小此值.

Tree-based hierarchical RCU leaf-level fanout value

CONFIG_RCU_FANOUT_LEAF

这个选项控制着树形 RCU 层次结构的叶子层的端点数(leaf-level fanout).对于期望拥有更高能耗比(更节能)的系统,请保持其默认值(16).对于拥有成千上万个 CPU 的系统来说,应该考虑将其设为最大值(CONFIG_RCU_FANOUT).

Disable tree-based hierarchical RCU auto-balancing

CONFIG_RCU_FANOUT_EXACT

强制按照 CONFIG_RCU_FANOUT_LEAF 的值,而不是使用自动平衡树结构来实现 RCU 子系统.目前仅用于调试目的.未来也许会用于增强 NUMA 系统的性能.

Accelerate last non-dyntick-idle CPU's grace periods

CONFIG_RCU_FAST_NO_HZ

即使 CPU 还在忙碌,也允许进入 dynticks-idle 状态,并且阻止 RCU 每 4 个滴答就唤醒一次该 CPU,这样能够更有效的使用电力,同时也拉长了 RCU grace period 的时间,造成性能降低.如果能耗比对你而言非常重要(你想节省每一分电力),并且你不在乎系统性能的降低(CPU 唤醒时间增加),可以开启此选项.台式机和服务器建议关闭此选项.

Enable RCU priority boosting

CONFIG_RCU_BOOST

允许提升 RCU 子系统的实时优先级(包括读操作与写操作),以避免 RCU 操作被阻塞太长时间.如果系统的 CPU 负载经常很重,或者你需要快速的实时响应系统,那么就选"Y",否则应该选"N".

Real-time priority to boost RCU readers to

CONFIG_RCU_BOOST_PRIO

允许提升被长时间抢占(阻塞)的 RCU 读操作的实时优先级到什么程度.取值范围是[1,99].默认值"1"适用于实时应用程序中不包含 CPU 密集型(CPU-bound)线程的常规场合(例如大多数桌面系统).但是如果你的实时应用程序拥有一个或多个 CPU 密集型线程,那么可能需要增加这个值,具体可以参考内核帮助的说明.仅在你确实理解了的情况下再改变默认值.

Milliseconds to delay boosting after RCU grace-

period start

CONFIG_RCU_BOOST_DELAY

在提升 RCU 读操作的优先级之前,允许有多长时间潜伏期(阻塞),取值范围是[0,3000],单位是毫秒,默认值是"500".不确定的请使用默认值.

Offload RCU callback processing from boot-selected CPUs

CONFIG_RCU_NOCB_CPU

如果你想帮助调试内核可以开启,否则请关闭.

Build-forced no-CBs CPUs

在开启 CONFIG_RCU_NOCB_CPU 选项的情况下,指定哪些 CPU 是 No-CB CPU,相当于预先设置"rcu_nocbs="内核引导参数.

Kernel .config support

CONFIG_IKCONFIG

把内核的配置信息编译进内核中,以后可以通过 scripts/extract-ikconfig 脚本从内核镜像中提取这些信息

Enable access to .config through /proc/config.gz

CONFIG_IKCONFIG_PROC

允许通过 /proc/config.gz 文件访问内核的配置信息

Kernel log buffer size

CONFIG_LOG_BUF_SHIFT

设置内核日志缓冲区的最小尺寸(合理的设置应该等于 CONFIG_LOG_CPU_MAX_BUF_SHIFT*最大 CPU 数量): 12(最小

值)=4KB,...,16=64KB,17=128KB,18=256KB,...,25(最大值)

CPU kernel log buffer size contribution

CONFIG_LOG_CPU_MAX_BUF_SHIFT

每个 CPU 的内核日志缓存大小(通常只有几行文字,但在报告故障时可能会产生大量文字).例如在最大 CPU 数量(包含热插拔 CPU)为 64 的系统上,如果 CONFIG_LOG_BUF_SHIFT=18,那么该值应该设为 12

Memory placement aware NUMA scheduler

CONFIG_NUMA_BALANCING

允许自动根据 NUMA 系统的节点分布状况进行进程/内存均衡(方法很原始,就是简单的内存移动).这个选项对 UMA 系统无效.[提示]UMA 系统的例子:(1)只有一颗物理 CPU(即使是多核)的电脑,(2)不支持"虚拟 NUMA",或"虚拟 NUMA"被禁用的虚拟机(即使所在的物理机是 NUMA 系统)

Automatically enable NUMA aware memory/task placement

CONFIG_NUMA_BALANCING_DEFAULT_ENABLED

在 NUMA(Non-Uniform Memory Access Architecture)系统上自动启用进程/内存均衡,也就是自动开启 CONFIG_NUMA_BALANCING 特性.

Control Group support

CONFIG_CGROUPS

Cgroup(Control Group)是一种进程管理机制,可以针对一组进程进行系统资源的分配和管理,可用于 Cpusets,CFS(完全公平调度器),内存管理等子系统.此外,systemd 与 Docker/LXC 等容器也依赖于它.更多细节可以参考内核的"Documentation/cgroups/cgroups.txt"文件

Example debug cgroup subsystem

CONFIG_CGROUP_DEBUG

导出 cgroups 框架的调试信息,仅用于调试目的.

Freezer cgroup subsystem

CONFIG_CGROUP_FREEZER

允许冻结/解冻 cgroup 内所有进程.Docker 依赖于它.

PIDs cgroup subsystem

CONFIG_CGROUP_PIDS

允许限制同一 cgroup 内所有进程的数量,超出限制后将无法 fork()出新进程.

Device controller for cgroups

CONFIG_CGROUP_DEVICE

允许为 cgroup 建立设备白名单,这样 cgroup 内的进程将仅允许对白名单中的设备进行 mknod/open 操作.Docker 依赖于它.

Cpuset support

CONFIG_CPUSETS

CPUSET 支持:允许将 CPU 和内存进行分组,并指定某些进程只能运行于特定的分组.Docker 依赖于它.这里有一篇 CPUSET 的用法

Include legacy /proc/<pid>/cpuset file

CONFIG_PROC_PID_CPUSET

提供过时的 /proc/<pid>/cpuset 文件接口

Simple CPU accounting cgroup subsystem

CONFIG_CGROUP_CPUACCT

提供一个简单的资源控制器(Resource Controller,用于实现一组任务间的资源共享),以监控 cgroup 内所有进程的总 CPU 使用量.Docker 依赖于它.

Resource counters

CONFIG_RESOURCE_COUNTERS

为 cgroup 提供独立于 controller 资源计数器

Memory Resource Controller for Control Groups

CONFIG_MEMCG

为 cgroup 添加内存资源控制器,包含匿名内存和页面缓存(Documentation/cgroups/memory.txt).开启此选项后,将会增加关联到每个内存页 fixed memory 大小,具体在 64 位系统

上是 40bytes/PAGE_SIZE.仅在你确实明白什么是 memory resource controller 并且确实需要的情况下才开启此选项.此功能可以通过命令行选项"cgroup_disable=memory"进行关闭.Docker 依赖于它.

Memory Resource Controller Swap Extension

CONFIG_MEMCG_SWAP

给 Memory Resource Controller 添加对 swap 的管理功能.这样就可以针对每个 cgroup 限定其使用的 mem+swap 总量.如果关闭此选项, memory resource controller 将仅能限制 mem 的使用量,而无法对 swap 进行控制(进程有可能耗尽 swap).开启此功能会对性能有不利影响,并且为了追踪 swap 的使用也会消耗更多的内存(如果 swap 的页面大小是 4KB,那么每 1GB 的 swap 需要额外消耗 512KB 内存),所以在内存较小的系统上不建议开启.

Memory Resource Controller Swap Extension enabled by default

CONFIG_MEMCG_SWAP_ENABLED

如果开启此选项,那么将默认开启 CONFIG_MEMCG_SWAP 特性,否则将默认关闭.即使默认开启也可以通过内核引导参数"swapaccount=0"禁止此特性.

Memory Resource Controller Kernel Memory accounting

CONFIG_MEMCG_KMEM

为 Memory Resource Controller 添加对内核对象所占用内存的管理功能.和标准的 Memory Resource Controller 对内存的控制不一样之处在于:这些内核对象所占用的内存是基于每个内存页的,并且可以被 swap 到硬盘.使用这个功能可以确保 cgroup 中的进程不会单独耗尽所有内核资源.

HugeTLB Resource Controller for Control Groups

CONFIG_CGROUP_HUGETLB

为 cgroup 添加对 HugeTLB 页的资源控制功能.开启此选项之后,你就可以针对每个 cgroup 限定其对 HugeTLB 的使用.Docker 依赖于它.

Enable perf_event per-cpu per-container group (cgroup) monitoring

CONFIG_CGROUP_PERF

将 per-cpu 模式进行扩展,使其可以监控属于特定 cgroup 并运行于特定 CPU 上的线程.

Group CPU scheduler

CONFIG_CGROUP_SCHED

让 CPU 调度程序可以在不同的 cgroup 之间分配 CPU 的带宽.Docker 依赖于它.systemd 资源控制单元(resource control unit)的 CPUShares 功能依赖于它.

Group scheduling for SCHED_OTHER

CONFIG_FAIR_GROUP_SCHED

公平 CPU 调度策略,也就是在多个 cgroup 之间平均分配 CPU 带宽."鸡血补丁"CONFIG_SCHED_AUTOGROUP(自动分组调度功能)依赖于它.Docker 依赖于它.systemd 资源控制单元(resource control unit)的 CPUShares 功能也依赖于它.

CPU bandwidth provisioning for FAIR_GROUP_SCHED

CONFIG_CFS_BANDWIDTH

允许用户为运行在 CONFIG_FAIR_GROUP_SCHED 中的进程定义 CPU 带宽限制.对于没有定义 CPU 带宽限制的 cgroup 而言,可以无限制的使用 CPU 带宽.详情参见 Documentation/scheduler/sched-bwc.txt 文件.systemd 资源控制单元(resource control unit)的 CPUQuota 功能也依赖于它.

Group scheduling for SCHED_RR/FIFO

CONFIG_RT_GROUP_SCHED

允许用户为 cgroup 分配实时 CPU 带宽,还可以对非特权用户的实时进程组进行调度.详情参

见 Documentation/scheduler/sched-rt-group.txt 文档.

使用 systemd 的系统应该选"N".

Block IO controller

CONFIG_BLK_CGROUP

通用的块 IO 控制器接口,可以用于实现各种不同的控制策略.目前,IOSCHED_CFQ 用它来在不同的 cgroup 之间分配磁盘 IO 带宽(需要额外开启 CONFIG_CFQ_GROUP_IOSCHED),block io throttle 也会用它来针对特定块设备限制 IO 速率上限(需要额外开启 CONFIG_BLK_DEV_THROTTLING).更多信息可以参考"Documentation/cgroups/blkio-controller.txt"文件.

Enable Block IO controller debugging

CONFIG_DEBUG_BLK_CGROUP

仅用于调试 Block IO controller 目的.

Checkpoint/restore support

CONFIG_CHECKPOINT_RESTORE

在内核中添加"检查点/恢复"支持.也就是添加一些辅助的代码用于设置进程的 text, data, heap 段,并且在 /proc 文件系统中添加一些额外的条目.用于检测两个进程是否共享同一个内核资源的 kcmp()系统调用依赖于它.使用 systemd 的建议开启此项.

Namespaces support

CONFIG_NAMESPACES

命名空间支持.主要用于支持基于容器的轻量级虚拟化技术(比如 LXC 和 Linux-VServer 以及 Docker).

UTS namespace

CONFIG_UTS_NS

uname()系统调用的命名空间支持

IPC namespace

CONFIG_IPC_NS

进程间通信对象 ID 的命名空间支持

User namespace

CONFIG_USER_NS

允许容器使用 user 命名空间.如果开启此项,建议同时开启 CONFIG_MEMCG 和 CONFIG_MEMCG_KMEM 选项,以允许用户空间使用"memory cgroup"限制非特权用户的内存使用量.systemd 服务单元(service unit)中的"PrivateUsers="指令依赖于它.如果你打算构建一个 VPS 服务器也必须选"Y".

PID Namespaces

CONFIG_PID_NS

进程 PID 命名空间支持

Network namespace

CONFIG_NET_NS

网络协议栈的命名空间支持.systemd 服务单元(service unit)中的"PrivateNetwork=/PrivateDevices="指令依赖于它.

Require conversions between uid/gids and their internal representation

CONFIG_UIDGID_STRICT_TYPE_CHECKS

强制将 uid/gid 转换为内部表示形式,以让那些未对 uid/gid 进行转换的内核子系统代码也能正常编译.不确定的选"N".

Automatic process group scheduling

CONFIG_SCHED_AUTOGROUP

每个 TTY 动态地创建任务分组(cgroup),这样就可以降低高负载情况下的桌面延迟.也就是传说中的桌面"鸡血补丁",桌面用户建议开启.但服务器建议关闭.

Enable deprecated sysfs features to support old userspace tools

CONFIG_SYSFS_DEPRECATED

为了兼容旧版本的应用程序而保留过时的 sysfs 特性.仅当在使用 2008 年以前的发行版时才需要开启,2009 年之后的发行版中必须关闭.此外,使用 udev 或 systemd 的系统也必须关闭.

Enable deprecated sysfs features by default

CONFIG_SYSFS_DEPRECATED_V2

默认开启上述特性

Kernel->user space relay support (formerly relayfs)

CONFIG_RELAY

在某些文件系统(比如 debugfs)中提供中继(relay)支持(从内核空间向用户空间传递大批量数据).主要用于调试内核.

Initial RAM filesystem and RAM disk (initramfs/initrd) support

CONFIG_BLK_DEV_INITRD

初始内存文件系统(initramfs,2.6 以上内核的新机制,使用 cpio 格式,占据的内存随数据的增减自动增减)与初始内存盘(initrd,2.4 以前内核遗留的老机制,使用 loop 设备,占据一块固定的内存,需要额外开启 CONFIG_BLK_DEV_RAM 选项才生效)支持,一般通过 lilo/grub 的 initrd 指令加载.更多细节可以参考"Documentation/initrd.txt"文件,关于 initrd 到 initramfs 的进化(墙内镜像),可以参考 IBM 上的两篇文章:Linux2.6 内核的 Initrd 机制解析和 Linux 初始 RAM 磁盘（initrd）概述.

Initramfs source file(s)

CONFIG_INITRAMFS_SOURCE

如果你想将 initramfs 镜像直接嵌入内核(比如嵌入式环境或者想使用 EFI stub kernel),而不是通过 lilo/grub 这样的引导管理器加载,可以使用此选项,否则请保持空白.这个选项指明用来制作 initramfs 镜像的原料,可以是一个.cpio 文件,或一个 Initramfs 虚根目录(其下包含"bin,dev,etc,lib,proc,sys"等子目录),或一个描述文件.细节可以参考"Documentation/early-userspace/README"文档.[注意]内核帮助文档说可以指定多个目录或文件是错误的,实际只能接受单一的目录或文件

User ID to map to 0 (user root)

INITRAMFS_ROOT_UID

此选项仅在 CONFIG_INITRAMFS_SOURCE 中包含目录时才有效,将此值设为非零(例如"37"),那么所有 UID=37 的文件在打包到 initramfs 镜像内时,其 UID 都将被设为"0".

Group ID to map to 0 (group root)

INITRAMFS_ROOT_GID

此选项仅在 CONFIG_INITRAMFS_SOURCE 中包含目录时才有效,将此值设为非零(例如"37"),那么所有 GID=37 的文件在打包到 initramfs 镜像内时,其 GID 都将被设为"0".

Support initial ramdisks compressed using gzip

CONFIG_RD_GZIP

支持经过 gzip 压缩的 ramdisk 或 cpio 镜像

Support initial ramdisks compressed using bzip2

CONFIG_RD_BZIP2

支持经过 bzip2 压缩的 ramdisk 或 cpio 镜像

Support initial ramdisks compressed using LZMA

CONFIG_RD_LZMA

支持经过 LZMA 压缩的 ramdisk 或 cpio 镜像

Support initial ramdisks compressed using XZ

CONFIG_RD_XZ

支持经过 XZ 压缩的 ramdisk 或 cpio 镜像

Support initial ramdisks compressed using LZO

CONFIG_RD_LZO

支持经过 LZO 压缩的 ramdisk 或 cpio 镜像

Built-in initramfs compression mode

选择 initramfs 镜像的压缩格式."gzip"是兼容性最好的格式,但是压缩率却最低."XZ"是目前渐渐流行的格式,压缩率高,解压速度也不慢.

Optimize for size

CONFIG_CC_OPTIMIZE_FOR_SIZE

编译时优化内核尺寸(使用 GCC 的"-Os"而不是"-O2"参数编译),这会得到更小的内核,但是运行速度可能会更慢.主要用于嵌入式环境.

Configure standard kernel features (expert users)

CONFIG_EXPERT

配置标准的内核特性(仅供专家使用).这个选项允许你改变内核的"标准"特性(比如用于需要"非标准"内核的特定环境中),仅在你确实明白自己在干什么的时候才开启.

Enable 16-bit UID system calls

CONFIG_UID16

允许对 UID 系统调用进行过时的 16-bit 包装,建议关闭

Multiple users, groups and capabilities support

CONFIG_MULTIUSER

多用户(组)支持.若选"N",则所有进程都将以"UID=0,GID=0"运行(也就是禁止存在非 root 用户).选"Y",除非你确实知道自己在干什么.

sgetmask/ssetmask syscalls support

CONFIG_SGETMASK_SYSCALL

是否开启已被反对使用的 sys_sgetmask/sys_ssetmask 系统调用(已不再被 libc 支持).建议选"N".

Sysfs syscall support

CONFIG_SYSFS_SYSCALL

是否开启已被反对使用的 sys_sysfs 系统调用(已不再被 libc 支持).建议选"N".

Sysctl syscall support

CONFIG_SYSCTL_SYSCALL

二进制 sysctl 接口支持.由于现在流行直接通过/proc/sys 以 ASCII 明码方式修改内核参数(需要开启 CONFIG_PROC_SYSCTL 选项),所以已经不需要再通过二进制接口去控制内核参数,建议关闭它以减小内核尺寸.

Load all symbols for debugging/ksymoops

CONFIG_KALLSYMS

装载所有的调试符号表信息,会增大内核体积,仅供调试时选择

Include all symbols in kallsyms

CONFIG_KALLSYMS_ALL

在/proc/kallsyms 中包含内核知道的所有符号,内核将会增大 300K,仅在你确实需要的时候再开启

Enable support for printk

CONFIG_PRINTK

允许内核向终端打印字符信息.任何由 printk 显示的字符串通常记录在/var/log/messages 文件里.如果关闭,内核在初始化过程中将不会输出字符信息,这会导致很难诊断系统故障,并且"dmesg"命令也会失效.仅在你确实不想看到任何内核信息时选"N".否则请选"Y".

BUG() support

CONFIG_BUG

显示故障和失败条件(BUG 和 WARN),禁用它将可能导致隐含的错误被忽略.建议仅在嵌入式设备或者无法显示故障信息的系统上关闭

Enable ELF core dumps

CONFIG_ELF_CORE

内存转储支持,可以帮助调试 ELF 格式的程序,用于调试和开发用户态程序

Enable PC-Speaker support

CONFIG_PCSPKR_PLATFORM

主板上的蜂鸣器支持.主板上的蜂鸣器只能发出或长或短的"滴"或"嘟嘟"声,一般用于系统报警.不要和能够播放音乐的扬声器混淆.如果你的主板上没有就关闭,有的话(开机自检完成后一般能听到"滴"的一声)还是建议开启.

Enable full-sized data structures for core

CONFIG_BASE_FULL

在内核中使用全尺寸的数据结构.禁用它将使得某些内核的数据结构减小以节约内存,但是将会降低性能

Enable futex support

CONFIG_FUTEX

快速用户空间互斥(fast userspace mutexes)可以使线程串行化以避免竞态条件,也提高了响应速度.禁用它将导致内核不能正确的运行基于 glibc 的程序

Enable eventpoll support

CONFIG_EPOLL

Epoll 系列系统调用(epoll_*)支持,这是当前在 Linux 下开发大规模并发网络程序(比如 Nginx)的热门人选,设计目的是取代既有 POSIX select(2)与 poll(2)系统接口,systemd 依赖于它.建议开启.

Enable signalfd() system call

CONFIG_SIGNALFD

signalfd()系统调用支持,建议开启.传统的处理信号的方式是注册信号处理函数,由于信号是异步发生的,要解决数据的并发访问和可重入问题.signalfd 可以将信号抽象为一个文件描述

符,当有信号发生时可以对其 read,这样可以将信号的监听放到 select/poll/epoll 监听队列中.systemd 依赖于它.

Enable timerfd() system call

CONFIG_TIMERFD

timerfd()系统调用支持,建议开启.timerfd 可以实现定时器功能,将定时器抽象为文件描述符,当定时器到期时可以对其 read,这样也可以放到 select/poll/epoll 监听队列中.更多信息可以参考 linux 新的 API signalfd、timerfd、eventfd 使用说明.systemd 依赖于它.

Enable eventfd() system call

CONFIG_EVENTFD

eventfd()系统调用支持,建议开启.eventfd 实现了线程之间事件通知的方式,eventfd 的缓冲区大小是 sizeof(uint64_t),向其 write 可以递增这个计数器,read 操作可以读取,并进行清零.eventfd 也可以放到 select/poll/epoll 监听队列中.当计数器不是 0 时,有可读事件发生,可以进行读取.

Enable bpf() system call

CONFIG_BPF_SYSCALL

开启内核的 bpf()系统调用支持(从 3.15 版本开始引入),以支持 eBPF 功能.可用于内核调试与网络包过滤(tcpdump,libpcap,iptables).不确定的选"N".

Use full shmem filesystem

CONFIG_SHMEM

完全使用 shmem 来代替 ramfs.shmem 是基于共享内存的文件系统(可以使用 swap),在启用 CONFIG_TMPFS 后可以挂载为 tmpfs 供用户空间使用,它比简单的 ramfs 先进许多.仅在微型嵌入式环境中且没有 swap 的情况下才可能会需要使用原始的 ramfs.

Enable AIO support

CONFIG_AIO

开启 POSIX 异步 IO 支持.它常常被高性能的多线程程序使用,建议开启

Enable madvise/fadvise syscalls

CONFIG_ADVISE_SYSCALLS

开启内核的 madvise()/fadvise()系统调用支持(2.6.16 版本开始引入).以允许应用程序预先提示内核,它将如何使用特定的内存与文件.这种措施有助于提升应用程序的性能.建议选"Y".

Enable userfaultfd() system call

CONFIG_USERFAULTFD

开启内核的 userfaultfd()系统调用支持(从 4.3 版本开始引入).该特性可以被诸如 QEMU/KVM 之类的虚拟化技术用来提高 GuestOS 热迁移性能.

Enable PCI quirk workarounds

CONFIG_PCI_QUIRKS

开启针对多种 PCI 芯片组的错误规避功能,仅在确定你的 PCI 芯片组确实没有没有任何 bug 时才关闭此功能.至于究竟哪些芯片组有 bug,你可以直接打开"drivers/pci/quirks.c"文件查看.不确定的选"Y".

Enable membarrier() system call

CONFIG_MEMBARRIER

开启内核的 membarrier()系统调用支持(与 Memory Barrier 相关).有助于提升多 CPU 场景下的并行计算性能.建议选"Y".

Embedded system

CONFIG_EMBEDDED

如果你是为嵌入式系统编译内核,可以开启此选项,这样一些高级选项就会显示出来.单独选中此项本身对内核并无任何改变.

Kernel Performance Events And Counters

CONFIG_PERF_EVENTS

性能相关的事件和计数器支持(既有硬件的支持也有软件的支持).大多数现代 CPU 都会通过性能计数寄存器对特定类型的硬

件事件(指令执行,缓存未命中,分支预测失败)进行计数,同时又丝毫不会减慢内核和应用程序的运行速度.这些寄存器还会在某些事件计数到达特定的阈值时触发中断,从而可以对代码进行性能分析. Linux Performance Event 子系统对上述特性进行了抽象,提供了针对每个进程和每个 CPU 的计数器,并可以被 tools/perf/ 目录中的"perf"工具使用.

Debug: use vmalloc to back perf mmap() buffers

CONFIG_DEBUG_PERF_USE_VMALLOC

主要用于调试 vmalloc 代码.

Enable VM event counters for /proc/vmstat

CONFIG_VM_EVENT_COUNTERS

"/proc/vmstat"中包含了从内核导出的虚拟内存的各种统计信息.开启此项后可以显示较详细的信息(包含各种事件计数器),关闭此项则仅仅显示内存页计数.主要用于调试和统计.

Enable SLUB debugging support

CONFIG_SLUB_DEBUG

SLUB 调试支持,禁用后可显著降低内核大小,同时 /sys/kernel/slab 也将不复存在.

Disable heap randomization

CONFIG_COMPAT_BRK

禁用堆随机化(heap randomization)功能.堆随机化可以让针对堆溢出的攻击变得困难,但是不兼容那些古董级的二进制程序(2000 年以前).如果你不需要使用这些古董程序,那么选"N".

Choose SLAB allocator

选择内存分配管理器

SLAB

CONFIG_SLAB

久经考验的 slab 内存分配器,在大多数情况下都具有良好的适应性.

SLUB (Unqueued Allocator)

CONFIG_SLUB

SLUB 与 SLAB 兼容,但通过取消大量的队列和相关开销,简化了 slab 的结构.特别是在多核时拥有比 slab 更好的性能和更好的系统可伸缩性.

SLOB (Simple Allocator)

CONFIG_SLOB

SLOB 针对小型系统设计,做了非常激进的简化,以适用于内存非常有限(小于 64M)的嵌入式环境.

SLUB per cpu partial cache

CONFIG_SLUB_CPU_PARTIAL

让 SLUB 内存分配器使用基于每个 CPU 的局部缓存,这样可以加速分配和释放属于此 CPU 范围内的对象,但这样做的代价是增加对象释放延迟的不确定性.因为当这些局部缓存因为溢出而要被清除时,需要使用锁,从而导致延迟尖峰.对于需要快速响应的实时系统,应该选"N",服务器则可以选"Y".

Provide system-wide ring of trusted keys

CONFIG_SYSTEM_TRUSTED_KEYRING

在内核中创建一个密钥环,从而允许向密钥环上添加受信任的密钥,主要用于内核模块的签名.如果你开启了 CONFIG_MODULE_SIG,此项将被自动选中.不需要使用内核模块签名检查功能的应该选"N"

Profiling support

CONFIG_PROFILING

添加扩展的性能分析支持,可以被 OProfile 之类的工具使用.仅用于调试目的.

OProfile system profiling

CONFIG_OPROFILE

OProfile 性能分析工具支持,仅用于调试目的.

OProfile multiplexing support

CONFIG_OPROFILE_EVENT_MULTIPLEX

OProfile multiplexing 技术支持

Kprobes

CONFIG_KPROBES

Kprobes 是一个轻量级的内核调试工具,能在内核运行的几乎任意时间点进行暂停/读取/修改等操作的调试工具.仅供调试使用.

Optimize very unlikely/likely branches

CONFIG_JUMP_LABEL

针对内核中某些"几乎总是为真"或者"几乎总是为假"的条件分支判断使用"asm goto"进行优化(在分支预测失败时会浪费很多时间在回退上,但是这种情况极少发生).很多内核子系统都支持进行这种优化.建议开启.

Static key selftest

CONFIG_STATIC_KEYS_SELFTEST

在内核启动时对上述分支优化补丁进行一次自我检查.

Transparent user-space probes

CONFIG_UPROBES

Uprobes 与 Kprobes 类似,但主要用于用户空间的调试.

Stack Protector buffer overflow detection

GCC 的"stack-protector"功能可以在函数开始执行时,在函数的返回地址末端设置一个敏感值,当函数执行完成要返回时,检查这个敏感值,看看是否存在溢出.如果有溢出则表明可能受到

了堆栈溢出攻击,内核将通过 panic 来阻止可能的攻击.选项中的"None"表示关闭此功能,"Regular"表示启用此功能但是仅提供较弱的保护(需要 GCC-4.2 及以上版本),"Strong"则表示提供较强的保护(需要 GCC-4.9 及以上版本)

Enable GCOV-based kernel profiling

CONFIG_GCOV_KERNEL

基于 GCC 的 gcov(代码覆盖率测试工具)的代码分析支持,仅用于调试

Profile entire Kernel

CONFIG_GCOV_PROFILE_ALL

支持对整个内核进行分析.内核体积将会显著增大,并且运行速度显著减慢.

Enable loadable module support

可加载模块支持

Enable loadable module support

CONFIG_MODULES

打开可加载模块支持,可以通过"make modules_install"把内核模块安装在/lib/modules/中.然后可以使用 modprobe, lsmod, modinfo, insmod, rmmod 等工具进行各种模块操作.

Forced module loading

CONFIG_MODULE_FORCE_LOAD

允许使用"modprobe --force"在不校验版本信息的情况下强制加载模块,这绝对是个坏主意!建议关闭.

Module unloading

CONFIG_MODULE_UNLOAD

允许卸载已经加载的模块.如果将模块静态编译进内核中,那么内核的执行效率会更好.如果代码作为动态模块加载,那么不使用时可以减少内核的内存使用并减少启动的时间,然而内核和模块在内存上相互独立又会影响内核的执行性能.

Forced module unloading

CONFIG_MODULE_FORCE_UNLOAD

允许强制卸载正在使用中的模块(rmmod -f),即使可能会造成系统崩溃.这又是一个坏主意!建议关闭.

Module versioning support

CONFIG_MODVERSIONS

允许使用为其他内核版本编译的模块,可会造成系统崩溃.这同样是个坏主意!建议关闭.

Source checksum for all modules

CONFIG_MODULE_SRCVERSION_ALL

为模块添加"srcversion"字段,以帮助模块维护者准确的知道编译此模块所需要的源文件,从而可以校验源文件的变动.仅内核模块开发者需要它.

Module signature verification

CONFIG_MODULE_SIG

在加载模块时检查模块签名,详情参见"Documentation/module-signing.txt"文件.[!!警告!!]开启此选项后,必须确保模块签名后没有被 strip(包括 rpmbuild 之类的打包工具).

Require modules to be validly signed

CONFIG_MODULE_SIG_FORCE

仅加载已签名并且密钥正确的模块,拒绝加载未签名或者签名密钥不正确的模块

Automatically sign all modules

CONFIG_MODULE_SIG_ALL

在执行"make modules_install"安装模块的时候,自动进行签名.否则你必须手动使用 scripts/sign-file 工具进行签名.

Which hash algorithm should modules be signed with?

选择对模块签名时使用的散列函数.建议使用强度最高的"SHA-512"算法.注意:所依赖的散列算法必须被静态编译进内核.对于"SHA-512"来说,就是 CONFIG_CRYPTO_SHA512 和 CONFIG_CRYPTO_SHA512_SSSE3(如果你的 CPU 支持 SSSE3 指令集的话).

Compress modules on installation

CONFIG_MODULE_COMPRESS

在'make modules_install'时对内核模块进行压缩.传统的 module-init-tools 工具可能支持 gzip 压缩,而新式的 kmod 可能支持 gzip 与 xz 压缩.使用 Kbuild 在内核树之外编译的模块也会同样在安装时被压缩.不确定的选"N".

Enable the block layer

块设备支持

Enable the block layer

CONFIG_BLOCK

块设备支持,使用 SSD/硬盘/U 盘/SCSI/SAS 设备者必选.除非你是某些特殊的嵌入式系统,否则没有理由不使用块设备.

Block layer SG support v4

CONFIG_BLK_DEV_BSG

为块设备启用第四版 SG(SCSI generic)支持.v4 相比 v3 能够支持更复杂的 SCSI 指令(可变长度的命令描述块,双向数据传输,通用请求/应答协议),而且 UDEV 也要用它来获取设备的序列号.对于使用 systemd 的系统来说,建议选"Y".对于不使用 systemd 的系统,如果你需要通过/dev/bsg/*访问块设备,也建议开启此选项,否则(通过/dev/{sd*,st*,sr*})可以关闭.

Block layer SG support v4 helper lib

CONFIG_BLK_DEV_BSGLIB

你不需要手动开启此选项,如果有其他模块需要使用,会被自动开启.

Block layer data integrity support

CONFIG_BLK_DEV_INTEGRITY

某些块设备可以通过存储/读取额外的信息来保障端到端的数据完整性,这个选项为文件系统提供了相应的钩子函数来使用这个特性.如果你的设备支持 T10/SCSI Data Integrity Field 或者 T13/ATA External Path Protection 特性,那么可以开启此选项,否则建议关闭.

Block layer bio throttling support

CONFIG_BLK_DEV_THROTTLING

Bio Throttling 支持,也就是允许限制每个 cgroup 对特定设备的 IO 速率.细节可以参考"Documentation/cgroups/blkio-controller.txt".

Block device command line partition parser

CONFIG_BLK_CMDLINE_PARSER

允许通过内核引导参数设定块设备的分区信息(Documentation/block/cmdline-partition.txt).仅对某些嵌入式设备有意义.

Advanced partition selection

CONFIG_PARTITION_ADVANCED

如果你想支持各种不同的磁盘分区格式(特别是与 UEFI 配合使用的 GPT 格式),务必选中此项.

Acorn partition support

CONFIG_ACORN_PARTITION

Acorn 操作系统使用的分区格式,请根据实际情况选择子项,这里省略

Alpha OSF partition support

CONFIG_OSF_PARTITION

Alpha 平台上使用的分区格式

Amiga partition table support

CONFIG_AMIGA_PARTITION

AmigaOS 使用的分区格式

Atari partition table support

CONFIG_ATARI_PARTITION

Atari OS 使用的分区格式

Macintosh partition map support

CONFIG_MAC_PARTITION

苹果的 Macintosh 平台使用的分区格式

PC BIOS (MSDOS partition tables) support

CONFIG_MSDOS_PARTITION

渐成历史垃圾,但目前依然最常见的 DOS 分区格式.除非你确信不使用此格式,否则必选.其下的子项根据实际情况选择.

Windows Logical Disk Manager (Dynamic Disk) support

CONFIG_LDM_PARTITION

使用 Windows Logical Disk Manager 创建的分区格式.参见 "Documentation/ldm.txt"

SGI partition support

CONFIG_SGI_PARTITION

SGI 平台上使用的分区格式

Ultrix partition table support

CONFIG_ULTRIX_PARTITION

DEC/Compaq Ultrix 平台上使用的分区格式

Sun partition tables support

CONFIG_SUN_PARTITION

SunOS 平台上使用的分区格式

Karma Partition support

CONFIG_KARMA_PARTITION

Rio Karma MP3 player 使用的分区格式

EFI GUID Partition support

CONFIG_EFI_PARTITION

代表未来趋势,眼下正大红大紫的 EFI GPT(GUID Partition Table)分区格式.建议开启.如果你在 UEFI 平台上安装则必须开启.

SYSV68 partition table support

CONFIG_SYSV68_PARTITION

Motorola Delta 机器上使用的分区格式

IO Schedulers

IO 调度器(另一篇文章)

Deadline I/O scheduler

CONFIG_IOSCHED_DEADLINE

deadline 调度器.简洁小巧(只有 400+行代码),提供了最小的读取延迟,非常适合同一时间只有少数个别进程进行 IO 请求的情况.如果你希望尽快读取磁盘,而不介意写入延迟,那它是最佳选择.通常对于数据库工作负载有最佳的表现.

CFQ I/O scheduler

CONFIG_IOSCHED_CFQ

cfq(Complete Fair Queuing)调度器.努力在各内核线程间公平分配 IO 资源,适用于系统中存在着大量内核线程同时进行 IO

请求的情况.但对于只有少数内核线程进行密集 IO 请求的情况,则会出现明显的性能下降.

CFQ Group Scheduling support

CONFIG_CFQ_GROUP_IOSCHED

允许将 CFQ 和 cgroup 组合使用,也就是将每个 cgroup 看成一个整体,在各 cgroup 之间进行 IO 资源的分配.参见"Documentation/cgroups/blkio-controller.txt"文件.还可以参考一下《Linux 内核精髓》中的"使用 Block I/O 控制器"一章.

BFQ I/O scheduler

CONFIG_IOSCHED_BFQ

bfq(Budget Fair Queueing)调度器.这是一个基于 CFQ 调度器的改进版本,更适合于对交互性要求比较高的场合,比如桌面系统和实时系统.如果静态编译进内核,还支持和 cgroup 配合,实现分层调度(hierarchical scheduling).

BFQ hierarchical scheduling support

CONFIG_CGROUP_BFQIO

通过 cgroup 文件系统接口,允许将 BFQ 分层使用(类似 CONFIG_CFQ_GROUP_IOSCHED),这个子系统的名字是"bfqio".

Default I/O scheduler

默认 IO 调度器.如果上述调度器都是模块,那么将使用最简单的内置 NOOP 调度器.NOOP(No Operation)调度器只是一个简单的 FIFO 队列,不对 IO 请求做任何重新排序处理(但还是会做一定程度的归并),适合于 SSD/U 盘/内存/虚拟机硬盘/SAN(Storage Area Networks)等*无需寻道*的存储设备,重点是可以节约 CPU 资源,但不适用于普通硬盘这样的需要依靠磁头来定位的设备.另外,有人说拥有 TCQ/NCQ 技术(能够自动重新排序)的硬盘也适合用 NOOP 调度器,这个说法其实并不那么合理,但笔者在此不敢断言,希望读者在严谨的测试之后再做定夺.

Processor type and features

中央处理器(CPU)类型及特性

DMA memory allocation support

CONFIG_ZONE_DMA

允许为寻址宽度不足 32 位的设备(也就是 ISA 和 LPC 总线设备)在物理内存的前 16MB 范围内(也就是传统上 x86_32 架构的 ZONE_DMA 区域)分配内存.不确定的选"N",内核中若有其它驱动(主要是某些老旧的声卡)需要它会自动选中此项.[提示]LPC 总线通常和主板上的南桥物理相连,通常连接了一系列的传统设备:BIOS,PS/2 键盘,PS/2 鼠标,软盘,并口设备,串口设备,某些集成声卡,TPM(可信平台模块),等等.[题外话]x86_64 已经没有 ZONE_HIGHMEM 了

Symmetric multi-processing support

CONFIG_SMP

SMP(对称多处理器)支持,如果你有多个 CPU 或者使用的是多核 CPU 就选上.

Processor feature human-readable names

CONFIG_X86_FEATURE_NAMES

让 /proc/cpuinfo 中的 CPU 特性标记更具可读性.选"Y".

Support x2apic

CONFIG_X86_X2APIC

x2apic 支持.具有这个特性的 CPU 可以使用 32 位的 APIC ID(可以支持海量的 CPU),并且可以使用 MSR 而不是 mmio 去访问 local APIC (更加高效).可以通过"grep x2apic /proc/cpuinfo"命令检查你的 CPU 是否支持这个特性.注意:有时候还需要在 BIOS 中也开启此特性才真正生效.[提示]在虚拟机中,还需要 VMM 的支持(例如 qemu-kvm).

Enable MPS table

CONFIG_X86_MPPARSE

如果是不支持 acpi 特性的古董级 SMP 系统就选上.但现今的 64 位系统早都已经支持 acpi 了,所以可以安全的关闭.

Support for extended (non-PC) x86 platforms

CONFIG_X86_EXTENDED_PLATFORM

支持非标准的 PC 平台：Numascale NumaChip，ScaleMP vSMP，SGI Ultraviolet。绝大多数人都遇不见这些平台。

Numascale NumaChip

CONFIG_X86_NUMACHIP

Numascale NumaChip 平台支持

ScaleMP vSMP

CONFIG_X86_VSMP

ScaleMP vSMP 平台支持

SGI Ultraviolet

CONFIG_X86_UV

SGI Ultraviolet 平台支持

Intel Low Power Subsystem Support

CONFIG_X86_INTEL_LPSS

为 Intel Lynx Point PCH 或更高级别芯片组中的 Intel Low Power Subsystem 技术提供支持。Lynx Point PCH 芯片组主要是为采用 LGA1150 的 Haswell 处理器提供支持。

AMD ACPI2Platform devices support

CONFIG_X86_AMD_PLATFORM_DEVICE

为 AMD Carrizo 以及后继架构的 I2C,UART,GPIO 提供支持。

Intel SoC IOSF Sideband support for SoC platforms

CONFIG_IOSF_MBI

为主打低功耗的 Intel SoC 平台 CPU 开启"sideband"寄存器访问支持.这些 CPU 包括:BayTrail,Braswell,Quark

Single-depth WCHAN output

CONFIG_SCHED_OMIT_FRAME_POINTER

使用简化的 /proc/<PID>/wchan 值,禁用此选项会使用更加精确的 wchan 值(可以在"ps -l"结果的 WCHAN 域看到),但会轻微增加调度器消耗.

Linux guest support

CONFIG_HYPERVISOR_GUEST

如果这个内核将在虚拟机里面运行就开启,否则就关闭.

Enable paravirtualization code

CONFIG_PARAVIRT

半虚拟化(paravirtualization)支持.

paravirt-ops debugging

CONFIG_PARAVIRT_DEBUG

仅供调试.paravirt-ops 是内核通用的半虚拟化接口.

Paravirtualization layer for spinlocks

CONFIG_PARAVIRT_SPINLOCKS

半虚拟化的自旋锁支持.开启之后运行在虚拟机里的内核速度会加快,但是运行在物理 CPU 上的宿主内核运行效率会降低(最多可能会降低 5%).请根据实际情况选择.

Xen guest support

CONFIG_XEN

Xen 半虚拟化技术支持

Enable Xen debug and tuning parameters in debugfs

CONFIG_XEN_DEBUG_FS

为 Xen 在 debugfs 中输出各种统计信息和调整选项.对性能有严重影响.仅供调试.

KVM Guest support (including kvmclock)

CONFIG_KVM_GUEST

KVM 客户机支持(包括 kvmclock).

Paravirtual steal time accounting

CONFIG_PARAVIRT_TIME_ACCOUNTING

允许进行更细粒度的 task steal time 统计.会造成性能的略微降低.仅在你确实需要的时候才开启.

Memtest

CONFIG_MEMTEST

为内核添加内存测试功能,也就是添加"memtest"内核引导参数以支持对内存进行"体检".仅在你确实知道这是什么东西并且确实需要的时候再开启.否则请关闭.

Processor family

处理器系列,请按照你实际使用的 CPU 选择."Generic-x86-64"表示通用于所有 x86-64 平台,而不是针对特定类型的 CPU 进行优化.

Supported processor vendors

CONFIG_PROCESSOR_SELECT

支持的 CPU 厂商,按实际情况选择.

Enable DMI scanning

CONFIG_DMI

允许扫描 DMI(Desktop Management Interface)/SMBIOS(System Management BIOS)以获得机器的硬件配置,从而对已知的 bug bios 进行规避.具体涉及到哪些机器可参见"drivers/acpi/blacklist.c"文件.除非确定你的机器没有 bug,否则请开启此项.

GART IOMMU support

CONFIG_GART_IOMMU

为较旧的 AMD Athlon64/Opteron/Turion/Sempron CPU 提供 GART IOMMU 支持.图形地址重映射表(Graphics Address Remapping Table)可以将物理地址不连续的系统内存映射成看上去连续的图形内存交给 GPU 使用,是一种挖 CPU 内存补 GPU 内存机制,这种机制也可以被认为是一种"伪 IOMMU"(缺乏地址空间隔离和访问控制).开启此选项以后,在内存大于 3G 的系统上,传统的 32 位总线(PCI/AGP)的设备将可以使用完全 DMA 的方式

直接访问原本超出 32 位寻址范围之外的系统内存区域.具体方法是:通过编程让设备在受 GART 控制的显存区域工作,然后使用 GART 将这个地址映射为真实的物理地址(4GB 以上)来实现的.USB/声卡/IDE/SATA 之类的设备常常需要它.开启此选项之后,除非同时开启了 CONFIG_IOMMU_DEBUG 选项或者使用了"iommu=force"内核引导参数,否则此特性仅在条件满足的情况下(内存足够大且确有支持 GART 的设备)激活.由于较新的 AMD CPU 都已配备了 AMD IOMMU(应该使用 CONFIG_AMD_IOMMU),故而仅建议在内存大于 3G 的老式 AMD 系统上选"Y".

IBM Calgary IOMMU support

CONFIG_CALGARY_IOMMU

IBM xSeries/pSeries 系列服务器的 Calgary IOMMU 支持.

Should Calgary be enabled by default?

CONFIG_CALGARY_IOMMU_ENABLED_BY_DEFAULT

开启此选项表示默认启用 Calgary 特性,关闭此选项表示默认禁用 Calgary 特性(可以使用"iommu=calgary"内核引导参数开启).

Enable Maximum number of SMP Processors and NUMA Nodes

CONFIG_MAXSMP

让内核支持 x86_64 平台所能支持的最大 SMP 处理器数量和最大 NUMA 节点数量.主要用于调试目的.

Maximum number of CPUs

CONFIG_NR_CPUS

支持的最大 CPU 数量,每个 CPU 要占 8KB 的内核镜像,最小有效值是"2",最大有效值是"512".注意:这里的"CPU 数量"是指"逻辑 CPU 数量".例如,对于一颗带有超线程技术的 4 核 8 线程 CPU 来说,相当于拥有 8 个 CPU.

SMT (Hyperthreading) scheduler support

CONFIG_SCHED_SMT

Intel 超线程技术(HyperThreading)支持.

Multi-core scheduler support

CONFIG_SCHED_MC

针对多核 CPU 进行调度策略优化

Preemption Model

内核抢占模式

No Forced Preemption (Server)

CONFIG_PREEMPT_NONE

禁止内核抢占,这是 Linux 的传统模式,可以得到最大的吞吐量,适合服务器和科学计算环境

Voluntary Kernel Preemption (Desktop)

CONFIG_PREEMPT_VOLUNTARY

自愿内核抢占,通过在内核中设置明确的抢占点以允许明确的内核抢占,可以提高响应速度,但是对吞吐量有不利影响.适合普通桌面环境的

Preemptible Kernel (Low-Latency Desktop)

CONFIG_PREEMPT

主动内核抢占,允许抢占所有内核代码,对吞吐量有更大影响,适合需要运行实时程序的场合或者追求最快响应速度的桌面环境.

Reroute for broken boot IRQs

CONFIG_X86_REROUTE_FOR_BROKEN_BOOT_IRQS

这是一个对某些芯片组 bug(在某些情况下会发送多余的"boot IRQ")的修复功能.开启此选项之后,仅对有此 bug 的芯片组生效.要检查哪些芯片组有此 bug 可以查看"drivers/pci/quirks.c"文件中的"quirk_reroute_to_boot_interrupts_intel"函数.

Machine Check / overheating reporting

CONFIG_X86_MCE

MCE(Machine Check Exception)支持.让 CPU 检测到硬件故障(过热/数据错误)时通知内核,以便内核采取相应的措施(如显示一条提示信息或关机等).更多信息可以"man mcelog"看看.可以通过"grep mce /proc/cpuinfo"检查 CPU 是否支持此特性,若支持建议选中,否则请关闭.当然,如果你对自己的硬件质量很放心,又是桌面系统的话,不选也无所谓.

Intel MCE features

CONFIG_X86_MCE_INTEL

Intel CPU 支持

AMD MCE features

CONFIG_X86_MCE_AMD

AMD CPU 支持

Machine check injector support

CONFIG_X86_MCE_INJECT

MCE 注入支持,仅用于调试

Enable vsyscall emulation

CONFIG_X86_VSYSCALL_EMULATION

对过时的 vsyscall 页提供仿真支持.禁用此项大致相当于使用"vsyscall=none"内核引导参数(差别在于当应用程序使用 vsyscall 时将直接崩溃(segfault)而不会产生警告消息).许多 2013 年之前编译的程序(也可能包括某些新近编译的程序)需要使用此特性.

Enable support for 16-bit segments

CONFIG_X86_16BIT

如果你需要使用 Wine 运行那些古董级的 16 位保护模式程序,就选"Y",否则选"N"

Dell laptop support

CONFIG_I8K

Dell Inspiron 8000 笔记本的 System Management Mode 驱动(i8k).该驱动可以读取 CPU 温度和风扇转速,进而帮助上层工具控制风扇转速.该驱动仅针对 Dell Inspiron 8000 笔记本进行过测试,所以不保证一定能适用于其他型号的 Dell 笔记本.

CPU microcode loading support

CONFIG_MICROCODE

CPU 的微代码更新支持,建议选中.CPU 的微代码更新就像是给 CPU 打补丁,用于纠正 CPU 的行为.更新微代码的常规方法是升级 BIOS,但是也可以在 Linux 启动后更新.比如在 Gentoo 下,可以使用"emerge microcode-ctl"安装 microcode-ctl 服务,再把这个服务加入 boot 运行级即可在每次开机时自动更新 CPU 微代码.其他 Linux 系统可以参考这个帖子.

Intel microcode loading support

CONFIG_MICROCODE_INTEL

Intel CPU 微代码支持

AMD microcode loading support

CONFIG_MICROCODE_AMD

AMD CPU 微代码支持

Early load microcode

CONFIG_MICROCODE_INTEL_EARLY

支持从 initrd 镜像首部加载微代码,以便尽可能早的更新 CPU 微代码.即使在 initrd 首部并未嵌入微代码也不会造成问题,所以"Y"是安全的.不过你真的需要吗?笔者认为你一般并不需要:)

/dev/cpu/*/msr - Model-specific register support

CONFIG_X86_MSR

允许用户空间的特权进程(使用 rdmsr 与 wrmsr 指令)访问 x86 的 MSR 寄存器(Model-Specific Register)以访问 CPU 的很多重要的参数.MSR 是非标准寄存器,主要用于读取 CPU 的工作状态(频率/电压/功耗/温度/性能等),以及设置 CPU 的工作参数(触发特定的 CPU 特性,依 CPU 的不同而不同).msrtool 工具可以转储出 MSR 的内容.不确定的可以选"M".

/dev/cpu/*/cpuid - CPU information support

CONFIG_X86_CPUID

允许用户空间的特权进程使用 CPUID 指令获得详细的 CPU 信息(CPUID):CPU 类型,型号,制造商信息,商标信息,序列号,缓存等.不确定的可以选"M".

Enable 1GB pages for kernel pagetables

CONFIG_DIRECT_GBPAGES

允许内核页表使用大小为 1GB 的 Hugepages 并进行直线映射(linear mapping),需要高端 CPU 的支持(可以用"grep

pdpe1gb /proc/cpuinfo"命令检查).这可以减小页表缓存(Translation Lookaside Buffer)的压力,从而提升系统的性能,这对于拥有海量内存并且运行某些特定应用(PosgreSQL,MySQL,Java,Memcached,KVM,Xen...)的系统来说比较有意义.如果你的 CPU 支持,可以选"Y".

Numa Memory Allocation and Scheduler Support

CONFIG_NUMA

开启 NUMA(Non Uniform Memory Access) 支持.虽然说集成了内存控制器的 CPU 都属于 NUMA 架构.但事实上,对于大多数只有一颗物理 CPU 的个人电脑而言,即使支持 NUMA 架构,也没必要开启此特性.可以参考 SMP/NUMA/MPP 体系结构对比.此外,对于不支持"虚拟 NUMA",或"虚拟 NUMA"被禁用的虚拟机(即使所在的物理机是 NUMA 系统),也应该关闭此项.

Old style AMD Opteron NUMA detection

CONFIG_AMD_NUMA

因为 AMD 使用一种旧式的方法读取 NUMA 配置信息(新式方法是 CONFIG_X86_64_ACPI_NUMA),所以如果你使用的是 AMD 多核 CPU,建议开启.不过,即使开启此选项,内核也会优先尝试 CONFIG_X86_64_ACPI_NUMA 方法,仅在失败后才会使用此方法,所以即使你不能确定 CPU 的类型也可以安全的选中此项.

ACPI NUMA detection

CONFIG_X86_64_ACPI_NUMA

使用基于 ACPI SRAT(System Resource Affinity Table)技术的 NUMA 节点探测方法.这也是检测 NUMA 节点信息的首选方法,建议选中.

NUMA emulation

CONFIG_NUMA_EMU

仅供开发调试使用

Maximum NUMA Nodes (as a power of 2)

CONFIG_NODES_SHIFT

允许的最大 NUMA 节点数.需要注意其计算方法:最大允许节点数=$2^{CONFIG\_NODES\_SHIFT}$.也就是说这里设置的值会被当做 2 的指数使用.取值范围是[1,10],也就最多允许 1024 个节点.

Memory model

内存模式."Sparse Memory"主要用来支持内存热插拔,相比其他两个旧有的内存模式,代码复杂性也比较低,而且还拥有一些性能上的优势,对某些架构而言是唯一的可选项.其他两个旧有的内存模式是:"Discontiguous Memory"和"Flat Memory".

Sparse Memory virtual memmap

CONFIG_SPARSEMEM_VMEMMAP

对于 64 位 CPU 而言,开启此选项可以简化 pfn_to_page/page_to_pfn 的操作,从而提高内核的运行效率.但是在 32 位平台则建议关闭.更多细节可以参考这个帖子.

Enable to assign a node which has only movable memory

CONFIG_MOVABLE_NODE

允许对一个完整的 NUMA 节点(CPU 和对应的内存)进行热插拔.

一般的服务器和个人电脑不需要这么高级的特性.

Allow for memory hot-add

CONFIG_MEMORY_HOTPLUG

支持向运行中的系统添加内存.也就是内存热插支持.

Allow for memory hot remove

CONFIG_MEMORY_HOTREMOVE

支持从运行中的系统移除内存.也就是内存热拔支持.

Allow for balloon memory compaction/migration

CONFIG_BALLOON_COMPACTION

允许压缩/合并气球内存(balloon memory).内存气球技术是指虚拟机在运行时动态地调整它所占用的宿主机内存资源,该技术在节约内存和灵活分配内存方面有明显的优势,目前所有主流虚拟化方案都支持这项技术(前提是客户机操作系统中必须安装有相应的 balloon 驱动).由于内存的动态增加和减少会导致内存过度碎片化,特别是对于 2M 尺寸的连续大内存页来说更加严重,从而严重降低内存性能.允许 balloon 内存压缩和合并可以很好的解决在客户机中使用大内存页时内存过度碎片化问题.如果你打算在虚拟机中使用大内存页(huge page),那么建议开启,否则建议关闭.

Allow for memory compaction

CONFIG_COMPACTION

允许对大内存页(huge pages)进行压缩.主要是为了解决大内存页的碎片问题.建议在使用大内存页的情况下开启此项,否则建议关闭.

Page migration

CONFIG_MIGRATION

允许在保持虚拟内存页地址不变的情况下移动其所对应的物理内存页的位置.这主要是为了解决两个问题:(1)在 NUMA 系统上,将物理内存转移到相应的节点上,以加快 CPU 与内存之间的访问速度.(2)在分配大内存页的时候,可以避免碎片问题.

Enable bounce buffers

CONFIG_BOUNCE

为那些不能直接访问所有内存范围的驱动程序开启 bounce buffer 支持.当 CONFIG_ZONE_DMA 被开启后,这个选项会被默认开启(当然,你也可以在这里手动关闭).这主要是为了那些不具备 IOMMU 功能的 PCI/ISA 设备而设,但它对性能有些不利影响.在支持 IOMMU 的设备上,应该关闭它而是用 IOMMU 来代替.

Enable KSM for page merging

CONFIG_KSM

KSM(Kernel Samepage Merging)支持:周期性的扫描那些被应用程序标记为"可合并"的地址空间,一旦发现有内容完全相同的

页面,就将它们合并为同一个页面,这样就可以节约内存的使用,但对性能有不利影响.推荐和内核虚拟机KVM(Documentation/vm/ksm.txt)或者其他支持"MADV_MERGEABLE"特性的应用程序一起使用.KSM 并不默认开启,仅在应用程序设置了"MADV_MERGEABLE"标记,并且/sys/kernel/mm/ksm/run 被设为"1"的情况下才会生效.

Low address space to protect from user allocation

CONFIG_DEFAULT_MMAP_MIN_ADDR

2009 年,内核曾经爆过一个严重的 NULL 指针漏洞,由于其根源是将 NULL 指针映射到地址"0"所致,所以从 2.6.32 版本以后,为了防止此类漏洞再次造成严重后果,特别设置了此选项,用于指定受保护的内存低端地址范围(可以在系统运行时通过 /proc/sys/vm/mmap_min_addr 进行调整),这个范围内的地址禁止任何用户态程序的写入,以从根本上堵死此类漏洞可能对系统造成的损害.但内核这种强加的限制,对于需要使用 vm86 系统调用(用于在保护模式的进程中模拟 8086 的实模式)或者需要映射此低端地址空间的程序(bitbake,dosemu,qemu,wine,...)来说,则会造成不兼容,不过目前这些程序的新版本都进行了改进,以适应内核的这种保护.一般情况下,"65536"是个明智的选择.

Enable recovery from hardware memory errors

CONFIG_MEMORY_FAILURE

在具备MCA(Machine Check Architecture)恢复机制的系统上,允许内核在物理内存中的发生数据错误的情况下,依然坚强的纠正错误并恢复正常运行.这需要有相应的硬件(通常是 ECC 内存)支持.有 ECC 内存的选,没有的就别选了.

HWPoison pages injector

CONFIG_HWPOISON_INJECT

仅用于调试.

Transparent Hugepage Support

CONFIG_TRANSPARENT_HUGEPAGE

大多数现代计算机体系结构都支持多种不同的内存页面大小(比如 x86_64 支持 4K 和 2M 以及 1G[需要 cpu-flags 中含有 "pdpe1gb"]).大于 4K 的内存页被称为"大页"(Hugepage).TLB(页表缓存)是位于 CPU 内部的分页表(虚拟地址到物理地址的映射表)缓冲区,既高速又很宝贵(尺寸很小).如果系统内存很大(大于 4G)又使用 4K 的内存页,那么分页表将会变得很大而难以在 CPU 内缓存,从而导致较高的 TLB 不命中概率,进而降低系统的运行效率.开启大内存页支持之后,就可以使用大页(2M 或 1G),从而大大缩小分页表的尺寸以大幅提高 TLB 的命中率,进而优化系统性能.传统上使用大内存页的方法是通过 Hugetlbfs 虚拟文件系统(CONFIG_HUGETLBFS),但是 hugetlbfs 需要专门进行配置以及应用程序的特别支持.所以从 2.6.38 版本开始引入了 THP(Transparent Hugepages),目标

是替代先前的 Hugetlbfs 虚拟文件系统(CONFIG_HUGETLBFS).THP 允许内核在可能的条件下,透明的(对应用程序来说)使用大页(huge pages)与 HugeTLB,THP 不像 hugetlbfs 那样需要专门进行配置以及应用程序的特别支持.THP 将这一切都交给操作系统来完成,也不再需要额外的配置,对于应用程序完全透明,因而可用于更广泛的应用程序.这对于数据库/KVM 等需要使用大量内存的应用来说,可以提升其效能,但对于内存较小(4G 或更少)的个人 PC 来说就没啥必要了.详见"Documentation/vm/transhuge.txt"文档.

Transparent Hugepage Support sysfs defaults

设置 /sys/kernel/mm/transparent_hugepage/enabled 文件的默认值."always"表示总是对所有应用程序启用透明大内存页支持,"madvise"表示仅对明确要求该特性的程序启用.建议选"always".

Cross Memory Support[Enable process_vm_readv/writev syscalls]

CONFIG_CROSS_MEMORY_ATTACH

交叉内存支持,也就是 process_vm_readv()和 process_vm_writev()系统调用支持.从而允许有权限的进程直接读取/写入另外一个进程的地址空间.现在它们只用于 openMPI 快速进程通信,也可以用于调试程序.未来也许还会有其他用途.

Enable cleancache driver to cache clean pages if tmem is present

CONFIG_CLEANCACHE

Cleancache 是内核 VFS 层新增的特性,可以被看作是内存页的"Victim Cache"(受害者缓存),当回收内存页时,先不把它清空,而是把其加入到内核不能直接访问的"transcendent memory"中,这样支持 Cleancache 的文件系统再次访问这个页时可以直接从"transcendent memory"加载它,从而减少磁盘 IO 的损耗.目前只有 zcache 和 XEN 支持"transcendent memory",不过将来会有越来越多的应用支持.开启此项后即使此特性不能得到利用,也仅对性能有微小的影响,所以建议开启.更多细节请参考"Documentation/vm/cleancache.txt"文件.

Enable frontswap to cache swap pages if tmem is present

CONFIG_FRONTSWAP

Frontswap 是和 Cleancache 非常类似的东西,在传统的 swap 前加一道内存缓冲(同样位于"transcendent memory"中).目的也是减少 swap 时的磁盘读写.CONFIG_ZSWAP 依赖于它,建议开启.

Contiguous Memory Allocator

CONFIG_CMA

这是一个分配连续物理内存页面的分配器.一些比较低端的 DMA 设备只能访问连续的物理内存,同时透明大内存页也需要连续的

物理内存.传统的解决办法是在系统启动时,在内存还很充足的时候,先预留一部分连续物理内存页面,留作后用,但这部分内存就无法被挪作他用了,为了可能的分配需求,预留这么一大块内存,并不是一个明智的方法.而连续内存分配器(Contiguous Memory Allocator)可以做到允许这部分预留的内存被正常使用,仅在确实需要的时候才将大块的连续物理内存分配给相应的驱动程序.这个机制对于那些不支持 I/O map 和 scatter-gather 的设备很有作用.详情参见"include/linux/dma-contiguous.h"文件.此选项仅对嵌入式系统有意义,不确定的选"N".

Track memory changes

CONFIG_MEM_SOFT_DIRTY

在内核页表的 PTE(Page Table Entry)数据结构上添加一个"soft-dirty"位以追踪内存页内容的变化.此特性基本上专用于 CRIU(Checkpoint/Restore In Userspace)项目(可以帮助容器进行热迁移).不确定的选"N".

Compressed cache for swap pages

CONFIG_ZSWAP

ZSWAP 是一个放置在 swap 前面的压缩缓存,它可以将需要换出的页压缩存放在内存中的压缩池里,这样在压缩池没有满的时候,可以避免使用真正的 swap 设备.当压缩池满的时候,则把最老的页解压后写入 swap 设备.压缩池默认是内存总量的

20%(/sys/module/zswap/parameters/max_pool_percent).

ZSWAP 不仅提升了 swap 的整体性能,也变相的增加了 swap 空间.选中此项后,可以通过"zswap.enabled=1"内核引导参数开启此功能.

Common API for compressed memory storage

CONFIG_ZPOOL

通用的内存压缩 API,主要用于给 zbud(zswap)或 zsmalloc 提供支持.不确定的选"N",如果内核有其他选项依赖于它会自动选中.

Low density storage for compressed pages

CONFIG_ZBUD

专用于 zswap 内部的低密度内存压缩 API,最多允许将两个物理内存页压缩为一个压缩内存页,这既有优势(简单的空间收集及空闲空间复用)也有劣势(潜在的低内存利用率).此种算法还能确保压缩后的内存页不会比最初未压缩页数多.不确定的选"N".

Memory allocator for compressed pages

CONFIG_ZSMALLOC

zsmalloc 压缩内存分配器主要用于给 zram 提供支持,建议与 CONFIG_ZRAM 同开关.参考:3 种内存压缩方案对比.

Use page table mapping to access object in zsmalloc

CONFIG_PGTABLE_MAPPING

zsmalloc 默认使用基于内存复制的对象映射方法来访问跨越不同页面的区域,但如果某些架构(例如 ARM)执行虚拟内存映射的速度快于内存复制,那么应该将此项选"Y",这将导致 zsmalloc 使用页表映射而不是内存复制来进行对象的映射.你可以在你的系统上使用 "https://github.com/spartacus06/zsmapbench"脚本来测试这两种方法的速度差异.在 x86_64 平台上,Debian8 与 Fedora22 与 openSUSE13 此项默认为"N",而 Ubuntu15 此项默认为"Y",作者本人未测试过哪个更合理.

Enable idle page tracking

CONFIG_IDLE_PAGE_TRACKING

此特性跟踪哪些用户页面需要被工作负载使用,哪些用户页面处于闲置状态.此信息(/sys/kernel/mm/page_idle)可用于确定工作负载需要的用户内存大小.从而帮助调优内存 cgroup 限制以及决定将此任务放置到集群中的那台机器上.参见 Documentation/vm/idle_page_tracking.txt 文档.不确定的选"N".

Support non-standard NVDIMMs and ADR protected memory

CONFIG_X86_PMEM_LEGACY

支持 Intel Sandy Bridge-EP 处理器使用的不符合 NVDIMM 规范的非易失内存(以电容做后备电力且掉电后不会丢失数据的内存).仅有某些高端服务器才会使用这种外带电容供电的内存.

Check for low memory corruption

CONFIG_X86_CHECK_BIOS_CORRUPTION

低位内存脏数据检查,即使开启此选项,默认也不会开启此功能(需要明确使用"memory_corruption_check=1"内核引导选项).这些脏数据通常被认为是有 bug 的 BIOS 引起的,默认每 60 秒(可以通过 memory_corruption_check_period 内核参数进行调整)扫描一次 0-64k(可以通过 memory_corruption_check_size 内核参数进行调整)之间的区域.这种检查所占用的开销非常小,基本可以忽略不计.如果始终检查到错误,则可以通过"memmap="内核引导参数来避免使用这段内存.一般没必要选中,如果你对 BIOS 不放心,带着它试运行一段时间,确认没问题之后再去掉.

Set the default setting of

memory_corruption_check

CONFIG_X86_BOOTPARAM_MEMORY_CORRUPTION_CHECK

设置 memory_corruption_check 的默认值,选中表示默认开启(相当于使用"memory_corruption_check=1"内核引导选项),不选中表示默认关闭.

Amount of low memory, in kilobytes, to reserve for the BIOS

CONFIG_X86_RESERVE_LOW

为 BIOS 设置保留的低端地址(默认是 64K).内存的第一页(4K)存放的必定是 BIOS 数据,内核不能使用,所以必须要保留.但是有许多 BIOS 还会在 suspend/resume/热插拔等事件发生的时候使用更多的页(一般在 0-64K 范围),所以默认保留 0-64K 范围.如果你确定自己的 BIOS 不会越界使用内存的话,可以设为"4",否则请保持默认值.但是也有一些很奇葩的 BIOS 会使用更多的低位内存,这种情况下可以考虑设为"640"以保留所有 640K 的低位内存区域.

MTRR (Memory Type Range Register) support

CONFIG_MTRR

MTRR(Memory type range registers)是 CPU 内的一组 MSR(Model-specific registers),其作用是告诉 CPU 以哪种模式(write-back/uncachable/...)存取各内存区段效率最高.这对于 AGP/PCI 显卡意义重大,因为 write-combining 技术可以将若干个总线写传输捆绑成一次较大的写传输操作,可以将图像写操作的性能提高 2.5 倍或者更多.这段代码有着通用的接口,其他 CPU 的寄存器同样能够使用该功能.简而言之,开启此选项是个明智的选择.

MTRR cleanup support

CONFIG_MTRR_SANITIZER

MTRR cleanup 的功能是将 MTRR 中的内存布局由连续布局转化为离散布局,这样 X 驱动就可以在其中添加 writeback 项(也就是一个内存段),算是一种优化措施.建议开启.可以使用"mtrr_chunk_size"来限制每段内存的最大尺寸.

MTRR cleanup enable value (0-1)

CONFIG_MTRR_SANITIZER_ENABLE_DEFAULT

"1"表示默认开启 CONFIG_MTRR_SANITIZER 特性,相当于使用"enable_mtrr_cleanup","0"表示默认关闭 CONFIG_MTRR_SANITIZER 特性,相当于使用"disable_mtrr_cleanup".建议图形界面用户设为"1".仅在开启后导致无法正常启动或者显卡驱动不能正常工作的情况下才需要关闭.

MTRR cleanup spare reg num (0-7)

CONFIG_MTRR_SANITIZER_SPARE_REG_NR_DEFAULT

这里设定的值等价于使用内核引导参数"mtrr_spare_reg_nr=N"中的"N".也就是告诉内核有多少个内存段(reg)可以被清理或改写(参见"/proc/mtrr"文件).在多数情况下默认值是"1",其含义是最多允许使用 1 个空闲的"reg".一般保持其默认值即可.修改此项的值通常是为了解决某些 MTRR 故障.

x86 PAT support

CONFIG_X86_PAT

PAT(Page Attribute Table)是对 MTRR 的补充,且比 MTRR 更灵活.如果你的 CPU 支持 PAT(grep pat /proc/cpuinfo),那么建议开启.仅在开启后导致无法正常启动或者显卡驱动不能正常工作的情况下才需要关闭.

x86 architectural random number generator

CONFIG_ARCH_RANDOM

Intel 从 Ivy Bridge 微架构开始(对于 Atom 来说是从 Silvermont 开始),在 CPU 中集成了一个高效的硬件随机数生成器(称为"Bull Mountain"技术),并引入了一个新的 x86 指令"RDRAND",可以非常高效的产生随机数.此选项就是对此特性的支持.

Supervisor Mode Access Prevention

CONFIG_X86_SMAP

SMAP(Supervisor Mode Access Prevention)是 Intel 从 Haswell 微架构开始引入的一种新特征,它在 CR4 寄存器上引入一个新标志位 SMAP,如果这个标志为 1,内核访问用户进程的地址空间时就会触发一个页错误,目的是为了防止内核因为自身错误意外访问用户空间,这样就可以避免一些内核漏洞所导致的安全问题.但是由于内核在有些时候仍然需要访问用户空间,因此 intel 提供了两条指令 STAC 和 CLAC 用于临时打开/关闭这个功

能,反复使用 STAC 和 CLAC 会带来一些轻微的性能损失,但考虑到增加的安全性,还是建议开启.

Intel MPX (Memory Protection Extensions)

CONFIG_X86_INTEL_MPX

Intel MPX(内存保护扩展)是一种用于检测缓冲区溢出 bug 的硬件特性.此选项并非用于保护内核自身,而是用于允许应用程序利用 MPX 特性.可以通过"grep mpx /proc/cpuinfo"检查你的 CPU 是否支持 MPX 特性.详见 Documentation/x86/intel_mpx.txt 文档.不确定的选"N".

EFI runtime service support

CONFIG_EFI

EFI/UEFI 支持.如果你打算在 UEFI/EFI 平台上安装 Linux(2010 年之后的机器基本都已经是 UEFI 规格了),那么就必须开启此项(开启后也依然可以在传统的 BIOS 机器上启动).UEFI 启动流程与传统的 BIOS 相差很大.虽然 Linux 受到了所谓"安全启动"问题的阻挠(已经解决),但是 UEFI 依然将迅速一统江湖.[提示]在 UEFI 平台上安装 Linux 的关键之一是首先要用一个支持 UEFI 启动的 LiveCD 以 UEFI 模式启动机器.

EFI stub support

CONFIG_EFI_STUB

EFI stub 支持.如果开启此项,就可以不通过 GRUB2 之类的引导程序来加载内核,而直接由 EFI 固件进行加载,这样就可以不

必安装引导程序了.不过这是一个看上去很美的特性,由于 EFI 固件灵活性比 GRUB2 差许多,所以缺点有三:(1)不能在传统的 BIOS 机器上启动.(2)给内核传递引导参数很麻烦(需要使用 "efibootmgr -u").(3)不能使用 intrd[待修正,可使用 "initrd="].不过,针对后两点的解决办法是:使用 CONFIG_CMDLINE 和 CONFIG_INITRAMFS_SOURCE.更多细节可参考"Documentation/x86/efi-stub.txt"文档.

EFI mixed-mode support

CONFIG_EFI_MIXED

允许在 32 位固件上启动 64 位内核.选"N".

Enable seccomp to safely compute untrusted bytecode

CONFIG_SECCOMP

允许使用 SECCOMP 技术安全地运算非信任代码.通过使用管道或其他进程可用的通信方式作为文件描述符(支持读/写调用),就可以利用 SECCOMP 把这些应用程序隔离在它们自己的地址空间.这是一种有效的安全沙盒技术.systemd 也强烈建议开启它.除非你是嵌入式系统,否则不要关闭.

Enable -fstack-protector buffer overflow detection

CONFIG_CC_STACKPROTECTOR

开启 GCC 的"-fstack-protector"命令行选项,以使用 GCC 中的编译器堆栈保护技术.这样可以有效的防御以堆栈溢出为代表的缓冲区溢出攻击,不过系统的运行速度也会受到一些影响.服

务器之类强调安全的场合建议开启,个人 PC 之类的就不是很有必要了.

Timer frequency

内核时钟频率.对于要求快速响应的场合,比如桌面环境,建议使用 1000Hz,而对于不需要快速响应的 SMP/NUMA 服务器,建议使用 250Hz 或 100Hz 或 300Hz(主要处理多媒体数据).

kexec system call

CONFIG_KEXEC

提供 kexec 系统调用,可以不必重启而切换到另一个内核(不一定必须是 Linux 内核),不过这个特性并不总是那么可靠.如果你不确定是否需要它,那么就是不需要.

kernel crash dumps

CONFIG_CRASH_DUMP

当内核崩溃时自动导出运行时信息的功能,主要用于调试目的.更多信息请参考"Documentation/kdump/kdump.txt"文件.

kexec jump

CONFIG_KEXEC_JUMP

kexec jump 支持.这是对 CONFIG_KEXEC 的增强功能,仅在你确实明白这是干啥的情况下再开启,否则请关闭.

Physical address where the kernel is loaded

CONFIG_PHYSICAL_START

加载内核的物理地址.如果内核不是可重定位的(CONFIG_RELOCATABLE=n),那么 bzImage 会将自己解压到该物理地址并从此地址开始运行,否则,bzImage 将忽略此处设置的值,而从引导装载程序将其装入的物理地址开始运行.仅在你确实知道自己是在干什么的情况下才可以改变该值,否则请保持默认.

Build a relocatable kernel

CONFIG_RELOCATABLE

使内核可以在浮动的物理内存位置加载,主要用于调试目的.仅在你确实知道为什么需要的时候再开启,否则请关闭.

Support for hot-pluggable CPUs

CONFIG_HOTPLUG_CPU

热插拔 CPU 支持(通过 /sys/devices/system/cpu 进行控制).

Set default setting of cpu0_hotpluggable

CONFIG_BOOTPARAM_HOTPLUG_CPU0

开启/关闭此项的意思是设置"cpu0_hotpluggable"的默认值为"on/off".开启此项表示默认将 CPU0 设置为允许热插拔.

Debug CPU0 hotplug

CONFIG_DEBUG_HOTPLUG_CPU0

仅用于调试目的.

Compat VDSO support

CONFIG_COMPAT_VDSO

是否将 VDSO(Virtual Dynamic Shared Object)映射到旧式的确定性地址.如果 Glibc 版本大于等于 2.3.3 选"N",否则就选"Y".

vsyscall table for legacy applications

设置内核引导参数"vsyscall=[native|emulate|none]"的值.对于使用 Glibc-2.14 以上版本的系统来说,如果不需要使用特别老旧的静态二进制程序,应该将此项设为"None"以提升性能与安全性.

Built-in kernel command line

CONFIG_CMDLINE_BOOL

将内核引导参数直接编进来.在无法向内核传递引导参数的情况下(比如在嵌入式系统上,或者想使用 EFI stub kernel),这就是唯一的救命稻草了.如果你使用 grub 之类的引导管理器,那么就可以不需要此特性.

Built-in kernel command string

CONFIG_CMDLINE

将要编译进内核的引导参数字符串.

Built-in command line overrides boot loader arguments

CONFIG_CMDLINE_OVERRIDE

开启此项表示完全忽略引导加载器传递过来的参数,并仅仅只使用 CONFIG_CMDLINE 所指定的参数.通常情况下建议关闭此项,除非你确定引导加载器在传递内核引导参数的时候不能正常工作.

Enable the LDT (local descriptor table)

CONFIG_MODIFY_LDT_SYSCALL

Linux 允许用户空间的应用程序使用 modify_ldt(2)系统调用针对每个 CPU 安装 Local Descriptor Table (LDT).某些老旧的程序或者运行在 DOSEMU/Wine 中的程序需要使用此接口.不确定的选"N"(尤其是嵌入式系统与服务器).

Power management and ACPI options

电源管理和 ACPI 选项

Suspend to RAM and standby

CONFIG_SUSPEND

"休眠到内存"(ACPI S3)支持.也就是系统休眠后,除了内存之外,其他所有部件都停止工作,重开机之后可以直接从内存中恢复运行状态.要使用此功能,你需要执行"echo mem >

/sys/power/state"命令,还需要在 BIOS 中开启 S3 支持,否则可能会有问题.

Enable freezer for suspend to RAM/standby

CONFIG_SUSPEND_FREEZER

选"Y".除非你知道自己在做什么

Hibernation (aka 'suspend to disk')

CONFIG_HIBERNATION

"休眠到硬盘"(ACPI S4)支持.也就是将内存的内容保存到硬盘(hibernation),所有部件全都停止工作.要使用此功能,你首先需要使用内核引导参数"resume=/dev/swappartition",然后执行"echo disk > /sys/power/state"命令.如果你不想从先前的休眠状态中恢复,那么可以使用"noresume"内核引导参数.更多信息,可以参考"Documentation/power/swsusp.txt"文件.

Default resume partition

CONFIG_PM_STD_PARTITION

默认的休眠分区.这个分区必须是 swap 分区.不过这里设置的值会被明确的内核引导参数中的值覆盖.

Opportunistic sleep

CONFIG_PM_AUTOSLEEP

这是一种从安卓借鉴过来的休眠方式.这个特性在安卓系统上被称为"suspend blockers"或"wakelocks".这是一种更激进的电源管理模式,以尽可能节约电力为目的.系统默认就处于休眠状态,仅为内存和少数唤醒系统所必须的设备供电,当有任务(唤醒源)需要运行的时候才唤醒相关组件工作,工作完成后又立即进入休眠状态.不过这些特性需要相应的设备驱动程序的支持.目前除了安卓设备,在 PC 和服务器领域,能够利用此特性的驱动还比较少,不过这是一项非常有前途的电源技术,喜欢尝鲜的可以考虑开启.

User space wakeup sources interface

CONFIG_PM_WAKELOCKS

允许用户空间的程序通过 sys 文件系统接口,创建/激活/撤销系统的"唤醒源".需要与 CONFIG_PM_AUTOSLEEP 配合使用.

Maximum number of user space wakeup sources (0 = no limit)

CONFIG_PM_WAKELOCKS_LIMIT

用户空间程序允许使用的"唤醒源"数量,"0"表示无限,最大值是"100000".

Garbage collector for user space wakeup sources

CONFIG_PM_WAKELOCKS_GC

对"唤醒源"对象使用垃圾回收.主要用于调试目的和 Android 环境.

Run-time PM core functionality[Device power management core functionality]

CONFIG_PM_RUNTIME

CONFIG_PM

允许 IO 设备(比如硬盘/网卡/声卡)在系统运行时进入省电模式,并可在收到(硬件或驱动产生的)唤醒信号后恢复正常.此功能通常需要硬件的支持.建议在笔记本/嵌入式等需要节约电力的设备上选"Y".

Power Management Debug Support

CONFIG_PM_DEBUG

仅供调试使用

Enable workqueue power-efficient mode by default

CONFIG_WQ_POWER_EFFICIENT_DEFAULT

因为"per-cpu workqueue"的缓存更靠近对应的 CPU,所以它比"unbound workqueue"拥有更好的性能,但另一方面"per-cpu workqueue"通常又比"unbound workqueue"需要消耗更多的电能.选中此项表示默认开启"workqueue.power_efficient"内核引导参数,以使用"unbound workqueue"而不是"per-cpu workqueue"以降低功耗,但是性能会有微小的损失.

ACPI (Advanced Configuration and Power Interface) Support

CONFIG_ACPI

高级配置与电源接口(Advanced Configuration and Power Interface)包括了软件和硬件方面的规范,目前已被软硬件厂商广泛支持,并且取代了许多过去的配置与电源管理接口,包括 PnP BIOS (Plug-and-Play BIOS), MPS(CONFIG_X86_MPPARSE), APM(Advanced Power Management) 等.总之,ACPI 已经成为 x86 平台必不可少的组件,如果你没有特别的理由,务必选中此项.

AML debugger interface (EXPERIMENTAL)

CONFIG_ACPI_DEBUGGER

仅供调试使用.

Deprecated /proc/acpi files

CONFIG_ACPI_PROCFS

过时的 /proc/acpi 接口支持,建议关闭.

Deprecated power /proc/acpi directories

CONFIG_ACPI_PROCFS_POWER

过时的 /proc/acpi 接口支持,建议关闭.

Allow supported ACPI revision to be overriden

CONFIG_ACPI_REV_OVERRIDE_POSSIBLE

某些笔记本固件会根据操作系统支持的 ACPI 版本决定硬件的工作模式.例如 Dell XPS 13 (2015) 期望操作系统支持"ACPI

v5"规范,但 Linux 实际上只支持"ACPI v4"规范,此时固件会将声卡的工作模式从 HDA 模式(Linux 支持此模式,且为首选模式)转换成 I2S 模式(次选模式).选中此项后,将强制 Linux 内核哄骗固件说它支持"ACPI v5"规范,相当于使用了"acpi_rev_override"内核引导参数.

EC read/write access through /sys/kernel/debug/ec

CONFIG_ACPI_EC_DEBUGFS

仅供调试使用.

Deprecated /proc/acpi/event support

CONFIG_ACPI_PROC_EVENT

过时的 /proc/acpi/event 接口支持,建议关闭.

AC Adapter

CONFIG_ACPI_AC

允许在外接交流电源和内置电池之间进行切换.

Battery

CONFIG_ACPI_BATTERY

允许通过 /proc/acpi/battery 接口查看电池信息.

Button

CONFIG_ACPI_BUTTON

允许守护进程通过 /proc/acpi/event 接口捕获 power/sleep/lid(合上笔记本)按钮事件,并执行相应的动作,软关机(poweroff)也需要它的支持.

Video

CONFIG_ACPI_VIDEO

对主板上的集成显卡提供 ACPI 支持.注意:仅支持集成显卡.

Fan

CONFIG_ACPI_FAN

允许用户层的程序对风扇进行控制(开/关/查询状态)

Dock

CONFIG_ACPI_DOCK

支持兼容 ACPI 规范的扩展坞(比如 IBM Ultrabay 和 Dell Module Bay)支持.

Processor

CONFIG_ACPI_PROCESSOR

在支持 ACPI C2/C3 的 CPU 上,将 ACPI 安装为 idle 处理程序.有几种 CPU 频率调节驱动依赖于它.而且目前的 CPU 都已经支持 ACPI 规范,建议开启此项.

IPMI

CONFIG_ACPI_IPMI

允许 ACPI 使用 IPMI(智能平台管理接口)的请求/应答消息访问 BMC(主板管理控制器).IPMI 通常出现在服务器中,以允许通过诸如 ipmitool 这样的工具监视服务器的物理健康特征(温度/电压/风扇状态/电源状态).

Processor Aggregator

CONFIG_ACPI_PROCESSOR_AGGREGATOR

支持 ACPI 4.0 加入的处理器聚合器(processor Aggregator)功能,以允许操作系统对系统中所有的 CPU 进行统一的配置和控制.目前只支持逻辑处理器(也就是利用 Intel 超线程技术虚拟出来的 CPU)idling 功能,其目标是降低耗电量.不确定的应该选"N".在某些服务器上此驱动(acpi_pad)可能与 BIOS 中的节能功能冲突

Thermal Zone

CONFIG_ACPI_THERMAL

ACPI thermal zone 支持.系统温度过高时可以及时调整风扇的工作状态以避免你的 CPU 被烧毁.目前所有 CPU 都支持此特性.务必开启.参见 CONFIG_THERMAL 选项.

NUMA support

CONFIG_ACPI_NUMA

通过读取系统固件中的 ACPI 表,获得 NUMA 系统的 CPU 及物理内存分布信息.NUMA 系统必选.

Custom DSDT Table file to include

CONFIG_ACPI_CUSTOM_DSDT_FILE

允许将一个定制过的 DSDT 编译进内核.详情参见"Documentation/acpi/dsdt-override.txt"文档.看不懂的请保持空白.

ACPI tables override via initrd

CONFIG_ACPI_INITRD_TABLE_OVERRIDE

允许 initrd 更改 ACPI tables 中的任意内容. ACPI tables 是 BIOS 提供给 OS 的硬件配置数据,包括系统硬件的电源管理和配置管理.详情参见"Documentation/acpi/initrd_table_override.txt"文件.

Debug Statements

CONFIG_ACPI_DEBUG

详细的 ACPI 调试信息,不搞开发就别选.

PCI slot detection driver

CONFIG_ACPI_PCI_SLOT

将每个 PCI 插槽都作为一个单独的条目列在 /sys/bus/pci/slots/ 目录中,有助于将设备的物理插槽位置与逻辑的 PCI 总线地址进行对应.不确定的选"No".

Power Management Timer Support
CONFIG_X86_PM_TIMER

ACPI PM Timer,简称"ACPI Timer",是一种集成在主板上的硬件时钟发生器,提供 3.579545MHz 固定频率.这是比较传统的硬件时钟发生器(HPET 则是比较新型的硬件时钟发生器),目前所有的主板都支持,而且是 ACPI 规范不可分割的部分.除非你确定不需要,否则必选.

Container and Module Devices
CONFIG_ACPI_CONTAINER

支持 NUMA 节点/CPU/内存 的热插拔. Device ID: ACPI0004, PNP0A05, PNP0A06 (find /sys/devices/ -name "PNP0A0[56]*" -or -name "ACPI0004*")

Memory Hotplug

CONFIG_ACPI_HOTPLUG_MEMORY

内存热插拔支持. Device ID: PNP0C80 (find /sys/devices/ -name "PNP0C80*")

Smart Battery System

CONFIG_ACPI_SBS

智能电池系统(Smart Battery System)可以让笔记型电脑显示及管理详细精确的电池状态信息.使用锂电池的笔记本电脑必备利器.但遗憾的是并不是所有笔记本都支持这项特性.

Hardware Error Device

CONFIG_ACPI_HED

Hardware Error Device (Device ID: PNP0C33) 能够通过 SCI 报告一些硬件错误(通常是已经被纠正的错误).如果你的系

统中有设备 ID 为"PNP0C33"的设备(find /sys/devices/ -name "PNP0C33*"),那么就选上.

Allow ACPI methods to be inserted/replaced at run time

CONFIG_ACPI_CUSTOM_METHOD

允许在不断电的情况下直接对 ACPI 的功能进行删改,包含一定危险性,它允许 root 任意修改内存中内核空间的内容.仅用于调试.

Boottime Graphics

Resou

rce

Table

suppo

rt

CONFI

G_ACP

I_BGR

T

在 /sys/firmware/acpi/bgrt/ 中显示 ACPI Boottime Graphics Resource Table ,以允许操作系统获取固件中的启动画面(splash).

Hard

ware

-

redu

ced

ACPI

supp

ort

only

CONF

IG_A

CPI_

REDU

CED_

HARD

WARE

_ONL

Y

以"reduced hardware"模式编译内核的 ACPI 代码,从而获得体积更小的内核但仅能运行在 ACPI "reduced hardware"模式的硬件上.不确定的选"N".

AC

PI

NV

DI

MM

Fi

rm

wa

re

In

te

rf
ac
e
Ta
bl
e
(N
FI
T)
CO
NF
IG
_A
CP
I_
NF
IT

非易失性内存(NVDIMM)支持.此种内存使用超级电容作为后备电力,并且使用非挥发性的 flash 存储介质来保存数据,以使数据能够在掉电之后依然保存.这是一种很有前途的技术,但是目前笔记本与普通服务器并不使用这种内存.

ACPIPlatformErrorInterr

face(APEI)CONFIG_ACPI_

APEI

高级平台错误接口(ACPI Platform Error Interface)是RAS(Reliability, Availability and Serviceability)的一部分,是定义在 ACPI 4.0 规范中的一个面向硬件错误管理的接口,主要是为了统一 firmware/BIOS 和 OS 之间的错误交互机制,使用标准的错误接口进行管理,同时也扩展了错误接口的内容以便实现更加灵活丰富的功能.

APEI Generic Hardware Error Source

CONFIG_ACPI_APEI_GHES

"Firmware First Mode"支持.由于 BIOS/FIRMWARE 是平台相关的,因此 BIOS/FIRMWARE 比 OS 更清楚硬件平台的配置情况,甚至包含各种必须的修正/定制/优化.这样,在"Firmware First"模式下,BIOS/FIRMWARE 利用这一优势,可以有针对性的对发生的硬件错误进行分析/处理/分发,也可以更准确的记录错误的现场信息.这样,不但对硬件错误可以做出更准确,更复杂的处理,而且可以降低 OS 的复杂性和冗余度.建议开启.

APEI PCIe AER logging/recovering support

CONFIG_ACPI_APEI_PCIEAER

让 PCIe AER errors 首先通过 APEI firmware 进行报告.

APEI memory error recovering support

CONFIG_ACPI_APEI_MEMORY_FAILURE

让 Memory errors 首先通过 APEI firmware 进行报告.

APEI Error INJection (EINJ)

CONFIG_ACPI_APEI_EINJ

仅供调试使用.

APEI Error Record Serialization Table (ERST) Debug Support

CONFIG_ACPI_APEI_ERST_DEBUG

仅供调试使用

Extended Error

rLogsupportCONFIG_ACPI

服务器 CPU 一般都会在非核心寄存器中记录比 CONFIG_X86_MCE 故障更详细的额外信息,诸如 PFA(Predictive Failure Analysis)之类的故障预警系统需要收集这些信息.但由于这些非核心寄存器的位置差别很大没有统一标准,系统软件难以直接读取这些扩展的错误信息.此驱动可以在 MCE 或 CMCI 机制之外,将系统固件提供的这些额外扩展错误信息导出到用户空间.不确定的选"N".

e

r

M

a

n

a

g

e

m

e

n

t

I

n

t

e

g

r

a

t

e

d

Circuit) operation regio

nsupportCONFIG_PMIC_OP

R

E

G

I

O

N

电源管理芯片(PMIC)支持.此种芯片常用于以电池作为电源的嵌入式装置.

SFI (Simple Firmware Interface) Support

CONFIG_SFI

简单固件接口规范(Simple Firmware Interface)使用一种轻量级的简单方法(通过内存中的一张静态表格)从 firmware 向操作系统传递信息.目前这个规范仅用于第二代 Intel Atom 平台,其核心名称是"Moorestown".

CPU Frequency scaling

CONFIG_CPU_FREQ

CPUfreq 子系统允许动态改变 CPU 主频,达到省电和降温的目的.现如今的 CPU 都已经支持动态频率调整,建议开启.不过,如果你是为虚拟机编译内核,就没有必要开启了,由宿主机内核去控制就 OK 了.

CPU frequency translation statistics

CONFIG_CPU_FREQ_STAT

通过 sysfs 文件系统输出 CPU 频率变化的统计信息

CPU frequency translation statistics details

CONFIG_CPU_FREQ_STAT_DETAILS

输出更详细的 CPU 频率变化统计信息

Default CPUFreq governor

默认的 CPU 频率调节策略.不同策略拥有不同的调节效果.

'performance' governor

CONFIG_CPU_FREQ_GOV_PERFORMANCE

'性能'优先,静态的将频率设置为 cpu 支持的最高频率.最耗电,发热量最大,性能/效率比最低.不建议使用.

'powersave' governor

CONFIG_CPU_FREQ_GOV_POWERSAVE

'节能'优先,静态的将频率设置为 cpu 支持的最低频率,严重影响性能.[注意]此调节器实际上并不能真正节省电能,因为系统需要花更长的时间才能进入空闲状态(C1E,C3,C6).但对于 CONFIG_X86_INTEL_PSTATE 驱动来说,这是效果最佳的调节器.

'userspace' governor for userspace frequency scaling

CONFIG_CPU_FREQ_GOV_USERSPACE

既允许手动调整 cpu 频率,也允许用户空间程序动态调整 cpu 频率(需要额外的调频软件).比较麻烦,不建议使用.

'ondemand' cpufreq policy governor

CONFIG_CPU_FREQ_GOV_ONDEMAND

'随需应变',内核周期性的考察 CPU 负载,当 CPU 负载超过/低于设定的百分比阈值(/sys/devices/system/cpu/cpufreq/ondemand/up_threshold)时,就自动将 cpu 频率设为最高/最低值(也就是仅在最高和最低频率间切换),比较适合台式机.[优化建议]将"up_threshold"设为 95 左右,可以获得更高的"性能/瓦特"比.

'conservative' cpufreq governor

CONFIG_CPU_FREQ_GOV_CONSERVATIVE

'保守',和'ondemand'相似,内核同样周期性的考察 CPU 负载,但是频率的升降是渐变式的(通常只在相邻的两档频率间切换,但具体取决于"/sys/devices/system/cpu/cpufreq/conservative/freq_step"的百分比设置,设为"100"则等价于仅允许在最高和最低频率间切换):当 CPU 负载超过百分比上限(/sys/devices/system/cpu/cpufreq/conservative/up_threshold)时,就自动提升一档 CPU 频率;当 CPU 负载低于百分比下限(/sys/devices/system/cpu/cpufreq/conservative/down_threshold)时,就自动降低一档 CPU 频率.更适合用于笔记本/PDA/x86_64 环境.[优化建议]'conservative'在默认设置下

的"性能/瓦特"比通常不如'ondemand'优秀,但是优化设置之后情况则可能反转.例如,在"down_threshold=93,up_threshold=97"的情况下,可以比"up_threshold=95"的'ondemand'略有优势.

x86 CPU frequency scaling drivers

CPU 频率调节器驱动

Intel P state control

CONFIG_X86_INTEL_PSTATE

此驱动是专用于 Intel 的"Sandy Bridge"/"Ivy Bridge"/"Haswell"/"Broadwell"/"SkyLake"或更新 CPU 微架构的首选驱动,可以更好的支持"Turbo Boost 2.0"技术.[注意]此驱动仅支持"performance"与"powersave"(首选)两种频率调节策略(但两者都支持动态频率调整),且"性能/瓦特"比都优于传统的'ondemand'.[提示]可以通过"echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo"关闭睿频加速,进一步降低 CPU 温度与性能.

Processor Clocking Control interface driver

CONFIG_X86_PCC_CPUFREQ

PCC(Processor Clocking Control)接口支持.此种接口仅对某些 HP Proliant 系列服务器有意义.更多细节可以参考"Documentation/cpu-freq/pcc-cpufreq.txt"文件.

ACPI Processor P-States driver

CONFIG_X86_ACPI_CPUFREQ

此驱动同时支持 Intel 和 AMD 的 CPU,这是较老的 intel cpu 与非 intel cpu 首选的驱动(除非你的 CPU 是古董级别).[注意]对于可以使用 P-state 驱动的 Intel CPU 来说,应该选"N".

Legacy cpb sysfs knob support for AMD CPUs

CONFIG_X86_ACPI_CPUFREQ_CPB

为了兼容旧的用户空间程序而设置,建议关闭.

AMD Opteron/Athlon64 PowerNow!

CONFIG_X86_POWERNOW_K8

过时的驱动,仅为老旧的 K8 核心的 AMD 处理器提供支持.K10 以及更新的 CPU 应该使用 CONFIG_X86_ACPI_CPUFREQ 驱动.

AMD frequency sensitivity feedback powersave bias

CONFIG_X86_AMD_FREQ_SENSITIVITY

如果你使用 AMD Family 16h 或者更高级别的处理器,同时又使用"ondemand"频率调节器,开启此项可以更有效的进行频率调节(在保证性能的前提下更节能).

Intel Enhanced SpeedStep (deprecated)

CONFIG_X86_SPEEDSTEP_CENTRINO

已被时代抛弃的驱动,仅对老旧的迅驰平台 Intel Pentium M / XEON 处理器有意义.

Intel Pentium 4 clock modulation

CONFIG_X86_P4_CLOCKMOD

已被时代抛弃的驱动,仅对支持老旧的 Speedstep 技术的 Intel Pentium 4 / XEON 处理器有意义.而且即便是在这样的 CPU 上,因为种种兼容性问题可能导致的不稳定,也不建议开启.

CPU idle PM support

CONFIG_CPU_IDLE

CPU idle 指令支持,该指令可以让 CPU 在空闲时"打盹"以节约电力和减少发热.只要是支持 ACPI 的 CPU 就应该开启.由于所有 64 位 CPU 都已支持 ACPI,所以不必犹豫,开启![提示]为虚拟机编译的内核就没有必要开启了,由宿主机内核去控制就 OK 了.

Support multiple cpuidle drivers

CONFIG_CPU_IDLE_MULTIPLE_DRIVERS

允许 CONFIG_CPU_IDLE 为每个不同的 CPU 使用不同的驱动.仅在你的系统由多个不同型号的 CPU 组成,并且具有不同的唤醒潜伏时间和状态的时候才需要开启.

Cpuidle Driver for Intel Processors

CONFIG_INTEL_IDLE

专用于 Intel CPU 的 cpuidle 驱动.而 CONFIG_CPU_IDLE 则可用于非 Intel 的 CPU.

Memory power savings

内存节能

Intel chipset idle memory power saving driver

CONFIG_I7300_IDLE

在某些具备内存节能特性的 intel 服务器芯片组上,让内存也可以在空闲时通过 idle 指令"打盹".这些芯片组必须具备 I/O AT 支持(例如 Intel 7300).同时内存也需要支持此特性.

Bus options (PCI etc.)

总线选项

PCI support

CONFIG_PCI

PCI 是最重要的内部总线,不但 PCI 与 PCI Express 设备依赖于它,而且 USB/IDE/SATA/SCSI/火线(IEEE 1394)/PCMCIA/CardBus 等各种内部和外部总线也都依赖于它.所以必须选"Y",除非你知道自己在干什么.

Support mmconfig PCI config space access

CONFIG_PCI_MMCONFIG

允许通过 mmconfig 方式访问 PCI config space,这种访问方式比传统的 IO 方式速度更快.建议开启.MMCONFIG 的意思是

"Memory-Mapped config",它是 PCI Express 引入的新总线枚举方式.背景知识:PCI 设备都有一组叫做'Configuration Space'的寄存器,PCI-E 设备在 PCI 的基础上又增加了一组叫做'Extended Configuration Space'的寄存器.这些寄存器都被映射到了内存中(Memory-Mapped),操作系统理应提供相应的 API 供设备驱动和诊断程序访问这些'Configuration Space'.但如果操作系统没有提供 Memory-Mapped 方式的 API 的话,这些驱动程序和诊断程序就必须自己根据操作系统的底层规则(IO 方式)去访问,这显然就增加了开发难度.这个选项的目的就是提供 Memory-Mapped 方式的 API.

Read CNB20LE Host Bridge Windows

CONFIG_PCI_CNB20LE_QUIRK

CNB20LE 芯片组 PCI 热插拔支持.除非你非常明确的知道你需要它,否则请关闭此项.

PCI Express support

CONFIG_PCIEPORTBUS

PCI Express 是 PCI 的升级版并在软件层与 PCI 兼容,其目标是统一电脑内部总线.基本上只要不是古董机,都早已支持 PCI-E 了.选"Y".

PCI Express Hotplug driver

CONFIG_HOTPLUG_PCI_PCIE

如果你的主板和设备都支持 PCI Express 热插拔就可以选上.

Root Port Advanced Error Reporting support

CONFIG_PCIEAER

PCI Express Root Port Advanced Error Reporting (AER) 驱动支持.这样,发送到 Root Port 的 Error reporting messages 就会由 PCI Express AER 处理.建议开启.背景知识:PCI Express 定义了两种错误报告范例:(1)baseline,所有 PCI-E 组件都必须要支持,功能也比较基础.(2)AER(Advanced Error Reporting),功能比较高级,也更可靠,但并不要求所有组件都支持.

PCI Express ECRC settings control

CONFIG_PCIE_ECRC

允许覆写 firmware/bios 设置的 PCI Express ECRC(端对端循环冗余校验).建议关闭,除非你确实知道为什么要开启.

PCIe AER error injector support

CONFIG_PCIEAER_INJECT

允许 PCI-E AER 注入,仅用于测试目的.

PCI Express ASPM control

CONFIG_PCIEASPM

PCI Express ASPM(Active State Power Management) 和 Clock Power Management 支持.这是 PCI-E 规范制定的一种电源管理方案,可以在设备空闲时采用节电模式.建议开启.ASPM 可以在运行时通过

/sys/module/pcie_aspm/parameters/policy 进行开启或关闭.

Debug PCI Express ASPM

CONFIG_PCIEASPM_DEBUG

仅供调试.

Default ASPM policy

默认的 ASPM 电源管理策略.下面的三个选项:"BIOS default"表示使用 BIOS 中的设置作为默认."Powersave"表示在可能的情况下,默认使用"L0s"和"L1",以尽可能节约电力."Performance"表示禁止使用"L0s"和"L1"(即使 BIOS 开启也同样禁止),以保证最高性能.

Message Signaled Interrupts (MSI and MSI-X)

CONFIG_PCI_MSI

PCI/PCI-E 支持三类中断:(1)INTx 使用传统的 IRQ 中断,可以与现行的驱动程序和操作系统兼容.(2)MSI 是 PCI2.2 规范中新增的,通过写入特殊的内存地址来触发和发送中断,该种方式脱离了中断引脚带来的数目限制,并且延迟小/效率高.不过 MSI 方式将中断全部落在单个 CPU 上,对多核 CPU 利用不佳.(3)MSI-X 是在 PCI3.0 规范中新增的,在 MSI 的基础上,支持更多的消息数量以及独立的消息地址,可以自动在多个 CPU 上分担中断,更适合多 CPU 系统.建议开启.开启后,也可以使用"pci=nomsi"内核引导参数关闭 MSI 特性.

PCI Debugging

CONFIG_PCI_DEBUG

将 PCI 调试信息输出到系统日志里.如果你想诊断 PCI 设备的故障,可以开启,否则应该关闭.

Enable PCI resource re-allocation detection

CONFIG_PCI_REALLOC_ENABLE_AUTO

让内核自动检测"是否需要重新分配 PCI 资源".即使此项已开启,你依然可以用"pci=realloc=[on|off]"来覆盖它.此项仅在已开启 CONFIG_PCI_IOV 的情况下才有意义.此时,如果 BIOS 没有为 SR-IOV(Single-Root I/O Virtualization) BAR(基地址寄存器)分配资源,那么内核将会自动对 PCI 资源进行重新分配.不确定的选"N".

PCI Stub driver

CONFIG_PCI_STUB

PCI 设备穿透(PCI Stub)的作用是将属主机的 PCI 设备跟目前绑定的驱动分离,暂时由其接管,最后再交给虚拟机内的客户操作系统自己去驱动这个 PCI 设备(例如网卡穿透/显卡穿透),以获得高性能.由于 USB 等所有外围设备实际上也都是连接在 PCI 总线上的,所以此功能同样适合各种外围设备,例如 U 盘加密狗之类.

Xen PCI Frontend

CONFIG_XEN_PCIDEV_FRONTEND

如果你使用 XEN 的半虚拟化技术,并且你的硬件支持 IOMMU,那么可以开启此项,否则应该关闭.

Interrupts on hypertransport devices

CONFIG_HT_IRQ

允许本地的 HyperTransport 设备使用中断.这个只可用于 AMD 平台,Intel 平台不支持这个.

PCI IOV support

CONFIG_PCI_IOV

PCI I/O Virtualization 支持.这需要硬件支持 IOMMU 技术(AMD-Vi,Intel VT-d).

PCI PRI support

CONFIG_PCI_PRI

PCI Page Request Interface 支持.它允许 IOMMU 之后的设备能够从页错误中恢复过来.这需要硬件支持 IOMMU 技术(AMD-Vi,Intel VT-d).

PCI PASID support

CONFIG_PCI_PASID

PASID(Process Address Space Identifiers)可以被 PCI 设备用来同时访问多个 IO 地址空间.这需要硬件 IOMMU 技术(AMD-Vi,Intel VT-d)支持 PASID 特性.不确定的选"N".

PCI IO-APIC hotplug support

CONFIG_PCI_IOAPIC

PCI IO-APIC 热插拔支持.

ISA-style DMA support

CONFIG_ISA_DMA_API

ISA-style DMA 控制器支持.目前基本只有 LPC 总线设备需要使用,最常见的是串口,并口,PS/2 键盘,Super I/O 芯片(可以使用 Superiotool 和 sensors-detect 工具检测).不确定的选"Y".[说明]这是一个历史遗留问题,对于 ISA 架构,DMA 操作是由一个专用的"DMA 控制器"(最常见的是 Intel 8237)来执行的,但是到了 PCI 架构,由于每一个 PCI 设备都可以控制 PCI 总线(成为"bus master")并直接读写系统内存,所以"DMA 控制器"又消失了.此选项只是为那些需要"ISA-DMA 控制器"的设备提供了兼容性接口(API)而已.

PCCard (PCMCIA/CardBus) support

CONFIG_PCCARD

PCCard(PCMCIA/CardBus/ExpressCard)接口通常出现在笔记本电脑上,这些接口卡通常大小与信用卡差不多,厚度大约 3-5

毫米.注意:必须要配合 pcmciautils 工具才能正常使用 PCMCIA 设备.

16-bit PCMCIA support

CONFIG_PCMCIA

老旧的 16-bit PCMCIA 卡支持

Load CIS updates from userspace

CONFIG_PCMCIA_LOAD_CIS

有些 PCMCIA 卡需要从用户空间更新 CIS(Card Information Structure)之后才能正常工作.开启此项后,内核将可以使用内置的固件加载器和热插拔子系统自动加载 CIS,而不再需要用户空间工具的辅助.建议选"Yes".

32-bit CardBus support

CONFIG_CARDBUS

常见的 PCMCIA 卡基本上都是 32 位的 CardBus 与 ExpressCard 设备.如果你有这样的卡,就选"Yes".由于绝大多数的卡都是"yenta-compatible"的,所以一般你还需要选中 CONFIG_YENTA 项.

CardBus yenta-compatible bridge support

CONFIG_YENTA

使用 PCMCIA 卡的基本上都需要选择这一项,子项是一些拥有自己特定扩展的硬件,请按实际情况选择.

{省略的部分请按照自己实际使用的 PCMCIA 卡选择}

Support for PCI Hotplug

CONFIG_HOTPLUG_PCI

PCI 热插拔不仅仅针对 PCI 和 PCI-E 设备,也包括 CardBus 与 ExpressCard 设备.请按需选择.

{省略的部分请按照自己实际使用 PCI 控制器进行选择}

RapidIO support

CONFIG_RAPIDIO

RapidIO 总线支持.这种总线主要用于嵌入式系统.

Discovery timeout duration (seconds)

CONFIG_RAPIDIO_DISC_TIMEOUT

等待主机完成枚举(也就是初始化)的超时秒数.

Enable RapidIO Input/Output Ports

CONFIG_RAPIDIO_ENABLE_RX_TX_PORTS

开启所有 RapidIO Input/Output 端口.

DMA Engine support for RapidIO

CONFIG_RAPIDIO_DMA_ENGINE

使用 DMA 引擎(CONFIG_DMADEVICES)进行 RapidIO 数据传输

RapidIO subsystem debug messages

CONFIG_RAPIDIO_DEBUG

将 RapidIO 调试信息输出到系统日志里.如果你想诊断 RapidIO 设备的故障,可以开启,否则应该关闭.

{省略的部分请按照自己实际使用的控制器进行选择}

Mark VGA/VBE/EFI FB as generic system framebuffer

CONFIG_X86_SYSFB

此选项的主要是为 simplefb(可作为 VGA/VBE/EFI FB 的单一替代品通用于 BIOS 和 UEFI 平台)提供支持,仅在你确实需要开启 CONFIG_FB_SIMPLE 选项时才需要选"Y",否则请选"N".

Executable file formats / Emulations

可执行文件格式/仿真

Kernel support for ELF binaries

CONFIG_BINFMT_ELF

ELF 是最常用的跨平台二进制文件格式,支持动态连接,支持不同的硬件平台,支持不同的操作系统.必选,除非你知道自己在做什么.

Write ELF core dumps with partial segments

CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS

如果你打算在此 Linux 上开发应用程序或者帮助别人调试 bug,那么就选"Y",否则选"N".注意这里的调试和开发不是指内核调试和开发,是应用程序的调试和开发.

Kernel support for scripts starting with #!

CONFIG_BINFMT_SCRIPT

支持以"#!/path/to/interpreter"行开头的脚本.务必"Y",不要"M"或"N",除非你知道自己在做什么.

Kernel support for MISC binaries

CONFIG_BINFMT_MISC

允许插入二进制封装层到内核中,直接运行 Java,.NET(Mono-based),Python,Emacs-Lisp 等语言编译的二进制程序时需要它,DOSEMU 也需要它.想要更方便的使用此特性(指定特定类型的文件用特定的程序打开),你还需要使用"mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc"挂载 binfmt_misc 伪文件系统.具体详情可以参考"Documentation/binfmt_misc.txt"文档.

Enable core dump support

CONFIG_COREDUMP

核心转储(core dump)支持.如果你打算在此 Linux 上开发应用程序或者帮助别人调试 bug,那么就选"Y",否则选"N".注意这里的调试和开发不是指内核调试和开发,是应用程序的调试和开发.

IA32 Emulation

CONFIG_IA32_EMULATION

允许在 64 位内核中运行 32 位代码.除非你打算使用纯 64 位环境,否则请开启此项.提示:GRUB2 支持引导纯 64 位内核,但是 GRUB 不支持.

IA32 a.out support

CONFIG_IA32_AOUT

早期 UNIX 系统的可执行文件格式(32 位),目前已经被 ELF 格式取代.除非你需要使用古董级的二进制程序.否则请关闭.

x32 ABI for 64-bit mode

CONFIG_X86_X32

允许 32 位程序(32 位指针)使用完整的 64 位寄存器,以减小内存占用(memory footprint).初衷是为了提高 32 位程序的运行性能,但实际在嵌入式 X86 环境中并无存在感,而对于非嵌入式 X86 环境又没有存在的意义,并且此特性未来会被删除.仅在你确实知道必须使用此特性时选"Y",否则就选"N".

Networking support

网络支持

Networking options

CONFIG_NET

网络选项.systemd 依赖于它

Packet socket

CONFIG_PACKET

链路层 PF_PACKET 套接字支持.可以让应用程序(比如:抓包工具 tcpdump,DHCP 客户端 dhclient,WiFi 设置工具 wpa_supplicant)直接与网络设备通讯,而无需使用内核中的其它中介协议.不确定的选"Y"或"M".

Packet: sockets monitoring interface

CONFIG_PACKET_DIAG

PF_PACKET 套接字监控接口,ss 这样的诊断工具需要它.

Unix domain sockets

CONFIG_UNIX

Unix domain sockets 支持.许多程序都使用它在操作系统内部进行进程间通信(IPC),比如: X Window, syslog, udev 等等.选"Y",除非你确实知道自己在做什么.

UNIX: socket monitoring interface

CONFIG_UNIX_DIAG

UNIX 套接字监控接口,ss 这样的工具需要它.

Transformation user configuration interface

CONFIG_XFRM_USER

为 IPsec 相关的工具提供 Transformation(XFRM)用户配置接口

Transformation sub policy support

CONFIG_XFRM_SUB_POLICY

XFRM 子策略支持,不确定的选"N".

Transformation migrate database

CONFIG_XFRM_MIGRATE

用于动态的更新 IPsec SA(security association) 的定位器(locator).这个特性对于手机这类移动设备来讲至关重要,因为它需要在不同的基站之间迁移.不确定的选"N".

Transformation statistics

CONFIG_XFRM_STATISTICS

转换统计,这不是 SNMP/MIB 规范的内容.用于调试目的.不确定的选"N".

PF_KEY sockets

CONFIG_NET_KEY

PF_KEYv2 套接字支持(与 KAME 兼容).PF_KEY 协议族主要用来处理 SA(安全关联),对 SADB(SA 数据库)进行管理,主要用在 IPsec 协议中.PF_KEY v2 的编程 API 在 RFC2367 中定义.

PF_KEY MIGRATE

CONFIG_NET_KEY_MIGRATE

向 PF_KEYv2 套接字中添加一个 PF_KEY MIGRATE 消息.

PF_KEY MIGRATE 消息可用于动态的更新 IPsec SA(security association) 的定位器(locator).这个特性对于手机这类移

动设备来讲至关重要,因为它需要在不同的基站之间迁移.不确定的选"N".

TCP/IP networking

CONFIG_INET

TCP/IP 协议,必选!

IP: multicasting

CONFIG_IP_MULTICAST

IP 多播(IP multicasting)支持.指的是一个发送者向一组特定的接收者发送数据,但只需发送一份数据副本.实际应用的场合很少,MBONE 算是其中之一,与 RTP 等音视频协议相结合也算一种.不确定的选"N".

IP: advanced router

CONFIG_IP_ADVANCED_ROUTER

高级路由支持,需要开启内核的 IP 转发功能(echo 1 > /proc/sys/net/ipv4/ip_forward)才能正常工作.如果这个 Linux 系统用作专业的路由器就选上,选上之后还需要按需选择其下的子项.一般的主机不需要这个.

FIB TRIE statistics

CONFIG_IP_FIB_TRIE_STATS

主要用于测试 TRIE 性能

IP: policy routing

CONFIG_IP_MULTIPLE_TABLES

策略路由

IP: equal cost multipath

CONFIG_IP_ROUTE_MULTIPATH

用于基于目的地址的负载均衡

IP: verbose route monitoring

CONFIG_IP_ROUTE_VERBOSE

显示冗余的路由监控信息

IP: kernel level autoconfiguration

CONFIG_IP_PNP

在内核启动时自动配置网卡的 ip 地址/路由表,配置信息来自于以下途径:内核引导参数,自举协议(BOOTP),反向地址转换协议(RARP),动态主机配置协议(DHCP).通常,需要从网络启动的无盘工作站才需要这个东西(此时还需要开启 CONFIG_ROOT_NFS),一般的发行版都通过启动脚本(dhcpcd/dhclient/ifconfig)配置网络.不确定的选"N".

IP: DHCP support

CONFIG_IP_PNP_DHCP

DHCP 协议支持

IP: BOOTP support

CONFIG_IP_PNP_BOOTP

BOOTP 协议支持

IP: RARP support

CONFIG_IP_PNP_RARP

RARP 协议支持

IP: tunneling

CONFIG_NET_IPIP

IP 隧道,主要目的是为了在 TCP/IP 网络中传输其他协议的数据包,当然也包括 IP 数据包(例如用于实现 VPN).

IP: GRE demultiplexer

CONFIG_NET_IPGRE_DEMUX

GRE demultiplexer 支持.被 CONFIG_NET_IPGRE 和 CONFIG_PPTP 所依赖.

IP: GRE tunnels over IP

CONFIG_NET_IPGRE

基于 IP 的通用路由封装(Generic Routing Encapsulation)隧道支持.该驱动主要用于对端是 Cisco 路由器的场合,因为 Cisco 的路由器特别偏好 GRE 隧道(而不是 CONFIG_NET_IPIP),并且 GRE 还允许通过隧道对组播进行再分发.

IP: broadcast GRE over IP

CONFIG_NET_IPGRE_BROADCAST

GRE/IP 的一种应用是构建一个广播 WAN(Wide Area Network),而其看上去却很像一个跑在互联网上的 LAN(Local Area Network).如果你想要创建这样的网络,那么就选"Y"(还要加上 CONFIG_IP_MROUTE).

IP: multicast routing

CONFIG_IP_MROUTE

组播路由支持.实际应用的场合很少,MBONE 算是其中之一,不确定的选"N".

IP: multicast policy routing

CONFIG_IP_MROUTE_MULTIPLE_TABLES

通常,组播路由器上会运行一个单独的用户态守护进程,根据源地址和目的地址来处理数据包.开启此项后,将能同时考虑数据包所带的标记(mark)和所通过的网络接口,并可在用户空间同时运行多个守护进程,每一个进程处理一张路由表.

IP: PIM-SM version 1 support

CONFIG_IP_PIMSM_V1

Sparse Mode PIM (Protocol Independent Multicast) version 1 支持. 该协议被 Cisco 路由器广泛支持,你需要特定的软件(pimd-v1)才能使用它.

IP: PIM-SM version 2 support

CONFIG_IP_PIMSM_V2

Sparse Mode PIM (Protocol Independent Multicast) version 2 支持. 该协议的使用并不广泛,你需要特定的软件(pimd 或 gated-5)才能使用它.

IP: ARP daemon support

CONFIG_ARPD

通常情况下,内核自身会使用 ARP 协议解析本地网络中的 IP 地址与 MAC 地址的对应关系,并进行缓存.开启此项后,内核将使用用户空间的守护进程进行 ARP 解析.这主要是为了使用其他的替代解析协议(比如 mGRE 隧道中的 NHRP),或调试目的.不确定的选"N".

IP: TCP syncookie support

CONFIG_SYN_COOKIES

TCP syncookie 支持,这是抵抗 SYN flood 攻击的好东西.此特性的开关可以通过"/proc/sys/net/ipv4/tcp_syncookies"文件控制,写入"1"表示开启,写入"0"表示关闭.建议服务器环境开启此项.

Virtual (secure) IP: tunneling

CONFIG_NET_IPVTI

虚拟 IP 隧道.可以和 xfrm 隧道一起使用,以实现 IPSEC 安全隧道,并在其上使用路由协议.不确定的选"N".

IP: Foo (IP protocols) over UDP

CONFIG_NET_FOU

允许将任意 IP 层协议封装到 UDP 隧道中.不确定的选"N".

IP: AH transformation

CONFIG_INET_AH

IPsec AH 支持.IPsec 验证头(AH)可对整个数据包(IP 报头与数据)提供身份验证/完整性/抗重播保护.但是它不提供保密性,即它不对数据进行加密.由于这个原因,AH 头正在慢慢被 ESP 头取代.

IP: ESP transformation

CONFIG_INET_ESP

IPsec ESP 支持.IPsec 封装安全负载(ESP)不仅为 IP 负载提供身份验证/完整性/抗重播保护,还提供保密性,也就是还对数据进行加密.ESP 有两种使用模式:传输模式(ESP 不对整个数据包进行签名,只对 IP 负载(不含 IP 报头)进行保护)和隧道模式(将原始 IP 包封装进新的带有 ESP 头的 IP 包内,可提供完整的保护).ESP 可以独立使用,也可与 AH 组合使用(越来越少).

IP: IPComp transformation

CONFIG_INET_IPCOMP

IP 静荷载压缩协议(IP Payload Compression Protocol)(RFC3173)支持.用于支持 IPsec

IP: IPsec transport mode

CONFIG_INET_XFRM_MODE_TRANSPORT

IPsec 传输模式.常用于对等通信,用以提供内网安全.数据包经过了加密但 IP 头没有加密,因此任何标准设备或软件都可查看和使用 IP 头

IP: IPsec tunnel mode

CONFIG_INET_XFRM_MODE_TUNNEL

IPsec 隧道模式.用于提供外网安全(包括虚拟专用网络).整个数据包(数据头和负载)都已经过加密处理且分配有新的 ESP 头/IP 头/验证尾,从而能够隐藏受保护站点的拓扑结构

IP: IPsec BEET mode

CONFIG_INET_XFRM_MODE_BEET

IPsec BEET 模式.

Large Receive Offload (ipv4/tcp)

CONFIG_INET_LRO

LRO(Large Receive Offload) (ipv4/tcp) 支持.它通过将多个 TCP 数据整合在一个 skb 结构中,并在稍后的某个时刻作为一个大的数据包交付给上层的网络协议栈,以减少上层协议栈处理 skb 的开销,提高 Linux 系统接收 TCP 数据包的能力.目前,主流网卡驱动都已支持此特性.建议开启.不过,LRO 不应该在路由器上开启,因为它破坏了 end-to-end 原则,并会对路由性能造成显著的不利影响.

INET: socket monitoring interface

CONFIG_INET_DIAG

INET(TCP,DCCP,...) socket 监视接口,一些 Linux 本地工具(如:包含 ss 的 iproute2)需要使用它

UDP: socket monitoring interface

CONFIG_INET_UDP_DIAG

UDP socket 监视接口,一些 Linux 本地工具(如:包含 ss 的 iproute2)需要使用它

TCP: advanced congestion control

CONFIG_TCP_CONG_ADVANCED

高级拥塞控制,子项提供多种拥塞控制算法供选用.如果没有特殊需求就别选了,内核会自动将默认的拥塞控制设为"CUBIC"并将"new Reno"作为候补.仅在你确实知道自己需要的情况下选"Y".不确定的选"N".

TCP: MD5 Signature Option support (RFC2385)

CONFIG_TCP_MD5SIG

RFC2385 中描述了一种对 TCP 会话进行 MD5 签名的保护机制.目前仅用于保护互联网运营商骨干路由器间的 BGP 会话.一般的路由器/服务器等设备根本不需要这个.

The IPv6 protocol CONFIG_IPV6

引领未来的 IPv6 支持.

IPv6: Privacy Extensions (RFC 3041) support

CONFIG_IPV6_PRIVACY

IPv6 利用"Stateless Address Autoconfiguration"在无 DHCP 服务器的情况下,产生可用的"临时 IPv6 地址".而本选项则为这个机制增加"隐私扩展"(RFC4941)保护.默认状态下,内核并不生产"临时地址",需要"echo 2 >/proc/sys/net/ipv6/conf/all/use_tempaddr"才能开启

IPv6: Router Preference (RFC 4191) support

CONFIG_IPV6_ROUTER_PREF

主机连上 IPv6 网络后,会发出路由器邀请包(Router Solicitation),路由器则应答路由器公告包(Router

Advertisement),其中包含网关地址/IPv6 前缀/DNS 地址,这样主机就能取得 IPv6 地址,并连接到互联网上,这就是无状态地址自动分配(StateLess Address AutoConfiguration)."Router Preference"是"Router Advertisement"包的可选扩展.它可以改进主机选中路由器的能力,特别是在多归属(multi-homed)网络中.不确定的选"N".

IPv6: Route Information (RFC 4191) support

CONFIG_IPV6_ROUTE_INFO

对"Route Information"的实验性支持.

IPv6: Enable RFC 4429 Optimistic DAD

CONFIG_IPV6_OPTIMISTIC_DAD

乐观重复地址检测(Optimistic Duplicate Address Detection)的实验性支持.可以更快的进行自动地址配置.不确定的选"N".

IPv6: AH transformation

CONFIG_INET6_AH

IPsec AH 支持.不确定的选"Y"或"M".AH 头正在慢慢被 ESP 头取代.

IPv6: ESP transformation

CONFIG_INET6_ESP

IPsec ESP 支持.不确定的选"Y"或"M".

IPv6: IPComp transformation

CONFIG_INET6_IPCOMP

IPv6 静荷载压缩协议(IP Payload Compression Protocol)(RFC3173)支持.用于支持 IPsec.不确定的选"Y"或"M".

IPv6: Mobility

CONFIG_IPV6_MIP6

移动 IPv6(RFC3775)支持.主要用于移动设备.不确定的选"N".

IPv6: IPsec transport mode

CONFIG_INET6_XFRM_MODE_TRANSPORT

IPsec 传输模式.常用于对等通信,用以提供内网安全.数据包经过了加密但 IP 头没有加密,因此任何标准设备或软件都可查看和使用 IP 头.不确定的选"Y"或"M".

IPv6: IPsec tunnel mode

CONFIG_INET6_XFRM_MODE_TUNNEL

IPsec 隧道模式.用于提供外网安全(包括虚拟专用网络).整个数据包(数据头和负载)都已经过加密处理且分配有新的 ESP 头/IP 头/验证尾,从而能够隐藏受保护站点的拓扑结构.不确定的选"Y"或"M".

IPv6: IPsec BEET mode

CONFIG_INET6_XFRM_MODE_BEET

IPsec BEET 模式.不确定的选"Y"或"M".

IPv6: MIPv6 route optimization mode
CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION

移动 IPv6(Mobile IPv6)路由优化模式.主要用于移动设备.不确定的选"N".

IPv6: IPv6-in-IPv4 tunnel (SIT driver)
CONFIG_IPV6_SIT

在 IPv4 网络上建立 IPv6 隧道.如果你希望可以通过 IPv4 网络接入一个 IPv6 网络,可以选"Y"或"M",否则选"N".

IPv6: IPv6 Rapid Deployment (6RD)
CONFIG_IPV6_SIT_6RD

IPv6 快速部署(6RD)支持.不确定的选"N".

IPv6: IP-in-IPv6 tunnel (RFC2473)
CONFIG_IPV6_TUNNEL

IPv6-in-IPv6/IPv4-in-IPv6 隧道(RFC2473)支持.不确定的选"N".

IPv6: GRE tunnel

CONFIG_IPV6_GRE

基于 IPv6 的通用路由封装(Generic Routing Encapsulation)隧道支持.该驱动主要用于对端是 Cisco 路由器的场合,因为 Cisco 的路由器特别偏好 GRE 隧道(而不是 CONFIG_IPV6_TUNNEL),并且 GRE 还允许通过隧道对组播进行再分发.

IPv6: Multiple Routing Tables

CONFIG_IPV6_MULTIPLE_TABLES

多重路由表(Multiple Routing Tables)支持.不确定的选"N".

IIPv6: source address based routing

CONFIG_IPV6_SUBTREES

允许根据源地址或前缀进行路由.不确定的选"N".

IPv6: multicast routing

CONFIG_IPV6_MROUTE

测试性的 IPv6 组播路由支持.实际应用的场合很少,不确定的选"N".

IPv6: multicast policy routing

CONFIG_IPV6_MROUTE_MULTIPLE_TABLES

通常,组播路由器上会运行一个单独的用户态守护进程,根据源地址和目的地址来处理数据包.开启此项后,将能同时考虑数据包所带的标记(mark)和所通过的网络接口,并可在用户空间同时运行多个守护进程,每一个进程处理一张路由表.

IPv6: PIM-SM version 2 support

CONFIG_IPV6_PIMSM_V2

IPv6 PIM multicast routing protocol PIM-SMv2 支持.

NetLabel subsystem support

CONFIG_NETLABEL

NetLabel 子系统支持.NetLabel 子系统为诸如 CIPSO 与 RIPSO 之类能够在分组信息上添加标签的协议提供支持,看不懂就别选了.

Security Marking

CONFIG_NETWORK_SECMARK

对网络包进行安全标记,类似于 nfmark,但主要是为安全目的而设计.看不懂的就别选了

Timestamping in PHY devices

CONFIG_NETWORK_PHY_TIMESTAMPING

允许在硬件支持的前提下,为物理层(PHY)数据包打上时间戳.这会略微增加发送与接收的开销.不确定的选"N".

Network packet filtering framework (Netfilter)

CONFIG_NETFILTER

Netfilter 可以对数据包进行过滤和修改,可以作为防火墙("packet filter"或"proxy-based")或网关(NAT)或代理(proxy)或网桥使用.

Network packet filtering debugging

CONFIG_NETFILTER_DEBUG

仅供开发者调试 Netfilter 使用

Advanced netfilter configuration

CONFIG_NETFILTER_ADVANCED

选"Y"将会显示所有模块供用户选择,选"N"则会隐藏一些不常用的模块,并自动将常用模块设为"M".

Bridged IP/ARP packets filtering

CONFIG_BRIDGE_NETFILTER

如果你希望使用桥接防火墙就打开它.Docker 依赖于它.不确定的选"N".

Core Netfilter Configuration

核心 Netfilter 配置(当包流过 Chain 时如果 match 某个规则那么将由该规则的 target 来处理,否则将由同一个 Chain 中的下一个规则进行匹配,若不 match 所有规则那么最终将由该 Chain 的 policy 进行处理)

Netfilter ingress support

CONFIG_NETFILTER_INGRESS

允许将入站包进行分类.

Netfilter NFACCT over NFNETLINK interface

CONFIG_NETFILTER_NETLINK_ACCT

允许通过 NFNETLINK 接口支持 NFACCT(记账).

Netfilter NFQUEUE over NFNETLINK interface

CONFIG_NETFILTER_NETLINK_QUEUE

允许通过 NFNETLINK 接口支持 NFQUEUE(排队).

Netfilter LOG over NFNETLINK interface

CONFIG_NETFILTER_NETLINK_LOG

允许通过 NFNETLINK 接口支持"LOG"(日志).该选项废弃了 ipt_ULOG 和 ebg_ulog 机制,并打算在将来废弃基于 syslog 的 ipt_LOG 和 ip6t_LOG 模块.

Netfilter connection tracking support

CONFIG_NF_CONNTRACK

连接追踪(connection tracking)支持,连接跟踪把所有连接都保存在一个表格内,并将每个包关联到其所属的连接.可用于报文伪装或地址转换,也可用于增强包过滤能力.

Connection mark tracking support

CONFIG_NF_CONNTRACK_MARK

允许对连接进行标记,与针对单独的包进行标记的不同之处在于它是针对连接流的. CONNMARK target 和 connmark match 需要它的支持.

Connection tracking security mark support

CONFIG_NF_CONNTRACK_SECMARK

允许对连接进行安全标记,通常这些标记包(SECMARK)复制到其所属连接(CONNSECMARK),再从连接复制到其关联的包(SECMARK).

Connection tracking zones

CONFIG_NF_CONNTRACK_ZONES

"conntrack zones"支持.通常,每个连接需要一个全局唯一标示符,而"conntrack zones"允许在不同 zone 内的连接使用相同的标识符.

Supply CT list in procfs (OBSOLETE)

CONFIG_NF_CONNTRACK_PROCFS

已被废弃,选"N".

Connection tracking events

CONFIG_NF_CONNTRACK_EVENTS

连接跟踪事件支持.如果启用这个选项,连接跟踪代码将提供一个"notifier"链,它可以被其它内核代码用来获知连接跟踪状态的改变

Connection tracking timeout

CONFIG_NF_CONNTRACK_TIMEOUT

连接跟踪"timeout"扩展.这样你就可以在网络流上通过 CT target 附加超时策略.

Connection tracking timestamping

CONFIG_NF_CONNTRACK_TIMESTAMP

时间戳支持.这样你就能在连接建立和断开时打上时间戳.

DCCP protocol connection tracking support

CONFIG_NF_CT_PROTO_DCCP

DCCP 协议支持.

SCTP protocol connection tracking support

CONFIG_NF_CT_PROTO_SCTP

SCTP 协议支持.

UDP-Lite protocol connection tracking support

CONFIG_NF_CT_PROTO_UDPLITE

UDP-Lite 支持.

Amanda backup protocol support

CONFIG_NF_CONNTRACK_AMANDA

Amanda 备份协议支持.

FTP protocol support

CONFIG_NF_CONNTRACK_FTP

文件传输协议(FTP)支持.跟踪 FTP 连接需要额外的帮助程序.

H.323 protocol support

CONFIG_NF_CONNTRACK_H323

H.323 协议支持.

IRC protocol support

CONFIG_NF_CONNTRACK_IRC

IRC扩展协议DCC(Direct Client-to-Client Protocol)支持.该协议允许用户之间绕开服务器直接聊天和传输文件.

NetBIOS name service protocol support

CONFIG_NF_CONNTRACK_NETBIOS_NS

NetBIOS协议支持.

SNMP service protocol support

CONFIG_NF_CONNTRACK_SNMP

SNMP协议支持.

PPtP protocol support

CONFIG_NF_CONNTRACK_PPTP

RFC2637 点对点隧道协议(Point to Point Tunnelling Protocol) 协议支持.

SANE protocol support

CONFIG_NF_CONNTRACK_SANE

SANE 协议支持.

SIP protocol support

CONFIG_NF_CONNTRACK_SIP

SIP 协议支持.

TFTP protocol support

CONFIG_NF_CONNTRACK_TFTP

TFTP 协议支持.

Connection tracking netlink interface

CONFIG_NF_CT_NETLINK

基于 netlink 的用户接口支持.

Connection tracking timeout tuning via Netlink

CONFIG_NF_CT_NETLINK_TIMEOUT

通过 Netlink 机制支持对连接追踪超时进行细粒度的调节:允许为特定的网络流指定超时策略,而不是使用统一的全局超时策略.

Connection tracking helpers in user-space via Netlink CONFIG_NF_CT_NETLINK_HELPER

通过 Netlink 机制为用户空间的连接追踪帮助程序提供基础框架.

NFQUEUE integration with Connection Tracking CONFIG_NETFI

LTER_NETLINK

_QUEUE_CT

开启此项后,即使网络包已经在队列(NFQUEUE)中,它依然可以包含连接追踪信息.

Transparent proxying support

CONFIG_NETFILTER_TPROXY

透明代理支持,也就是可以处理非本地的 IPv4 TCP/UDP 套接字.此功能需要配合一些 iptables 规则和策略路由才能工作.详见"Documentation/networking/tproxy.txt"文档.

Netfilter Xtables support (required

for ip_tables)

CONFIG_NETFILTER_XTABLES

如果你打算使用 ip_tables, ip6_tables, arp_tables 之一就必须选上

nfmark target and match support

CONFIG_NETFILTER_XT_MARK

"nfmark"是用户给包打上的一个自定义标记.用于 match 时,允许基于"nfmark"值对包进行匹配.用于 target 时,允许在"mangle"表中创建规则以改变包的"nfmark"值.

ctmark target and match support

CONFIG_NETFILTER_XT_CONNMARK

"ctmark"是用户以连接为组,给同一连接中的所有包打上的自定义标记.用法与"nfmark"相似.

set target and match support

CONFIG_NETFILTER_XT_SET

"set"是 ipset 工具创建的 IP 地址集合.使用 match 可以对 IP 地址集合进行匹配,使用 target 可以对集合中的项进行增加和删除.

AUDIT target support

CONFIG_NETFILTER_XT_TARGET_AUDIT

为被 drop/accept 的包创建审计记录.

CHECKSUM target support

CONFIG_NETFILTER_XT_TARGET_CHECKSUM

用于"mangle"表,为缺少校验和的包添加 checksum 字段的值.主要是为了兼容一些老旧的网络程序(例如某些 dhcp 客户端).

"CLASSIFY" target support

CONFIG_NETFILTER_XT_TARGET_CLASSIFY

允许为包设置优先级,一些 qdiscs 排队规则(atm,cbq,dsmark,pfifo_fast,htb,prio)需要使用它

"CONNMARK" target support

CONFIG_NETFILTER_XT_TARGET_CONNMARK

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_CONNMARK

"CONNSECMARK" target support

CONFIG_NETFILTER_XT_TARGET_CONNSECMARK

针对链接进行安全标记,同时还会将连接上的标记还原到包上(如果链接中的包尚未进行安全标记),通常与 SECMARK target 联合使用

"CT" target support

CONFIG_NETFILTER_XT_TARGET_CT

允许为包加上连接追踪相关的参数,比如"event"和"helper".

"DSCP" and "TOS" target support

CONFIG_NETFILTER_XT_TARGET_DSCP

DSCP target 允许对 IPv4/IPv6 包头部的 DSCP(Differentiated Services Codepoint)字段(常用于 Qos)进行修改. TOS target 允许在"mangle"表创建规则以修改 IPv4 包头的 TOS(Type Of Service)字段或 IPv6 包头的 Priority 字段.

"HL" hoplimit target support

CONFIG_NETFILTER_XT_TARGET_HL

HL(IPv6)/TTL(IPv4) target 允许更改包头的 hoplimit/time-to-live 值.

"HMARK" target support

CONFIG_NETFILTER_XT_TARGET_HMARK

允许在"raw"和"mangle"表中创建规则,以根据特定范围的哈希计算结果设置"skbuff"标记.

IDLETIMER target support

CONFIG_NETFILTER_XT_TARGET_IDLETIMER

每个被匹配的包的定时器都会被强制指定为规则指定的值,当超时发生时会触发一个 sysfs 文件系统的通知.剩余时间可以通过 sysfs 读取.

"LED" target support

CONFIG_NETFILTER_XT_TARGET_LED

允许在满足特定条件的包通过的时候,触发 LED 灯闪烁.比如可以用于控制网卡的状态指示灯仅在有 SSH 活动的时候才闪烁.

LOG target support

CONFIG_NETFILTER_XT_TARGET_LOG

允许向 syslog 中记录包头信息.

"MARK" target support

CONFIG_NETFILTER_XT_TARGE

T_MARK

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_MARK

"NETMAP" target support

CONFIG_NETFILTER_XT_TAR

GET_NETMAP

NETMAP 用于实现一对一的静态 NAT(地址转换).

"NFLOG" target support

CONFIG_NETFILTER_XT_TA

RGET_NFLOG

通过 nfnetlink_log 记录日志.

"NFQUEUE" target

Support

CONFIG_NETFILTER_XT_

TARGET_NFQUEUE

用于替代老旧的 QUEUE target. 因为 NFQUEUE 能支持最多 65535 个队列,而 QUEUE 只能支持一个.

"NOTRACK" target

support

(DEPRECATED)
CONFIG_NETFILTER_X
T_TARGET_NOTRACK

已被废弃,勿选.

"RATEEST" target
support
CONFIG_NETFILTER_
XT_TARGET_RATEEST

RATEEST target 允许测量网络流的传输速率.[注: rateest match 允许根据速率进行匹配.]

REDIRECT target
support
CONFIG_NETFILTE
R_XT_TARGET_RED
IRECT

REDIRECT 是一种特别的 NAT:所有进入的连接都被映射到其入口网卡的地址,这样这些包就会"流入"本机而不是"流过"本机.这主要用于实现透明代理.

"TEE" -
packet
cloning to
alternate

destination

CONFIG_NETFIL

TER_XT_TARGET

_TEE

对包进行克隆,并将克隆的副本路由到另一个临近的路由器(Next Hop).

"TPROXY"

target

support

CONFIG_NETFI

LTER_XT_TARG

ET_TPROXY

类似于 REDIRECT,但并不依赖于连接追踪和 NAT,也只能用于"mangle"表,用于将网络流量重定向到透明代理.

"TRACE"

target

support

CONFIG_NET

FILTER_XT_

TARGET_TRA

CE

允许对包打标记,这样内核就可以记录每一个匹配到的规则.

"SECMARK"

target

support

CONFIG_NE

TFILTER_X

T_TARGET_

SECMARK

允许对包进行安全标记,用于安全子系统

"TCPMSS

"

target

support

CONFIG_

NETFILT

ER_XT_T

ARGET_T

CPMSS

允许更改 TCP SYN 包的MSS(Maximum Segment Size)值,通常=MTU-40.

"TCPO

PTSTR

IP"

target

t

suppo

rt

CONFI

G_NET

FILTE

R_XT_

TARGE

T_TCP

OPTST

RIP

允许从 TCP 包头中剥离所有 TCP 选项.

"add

rtyp

e"

addr

ess

type

matc

h

supp

ort

CONF

IG_N

ETFI

LTER

_XT_

MATC

H_AD

DRTY

PE

根据地址类型进行匹配: UNICAST, LOCAL, BROADCAST, ...

Docker 依赖于它.

"b

pf

"

ma

tc

h

su

pp

or

t

CO

NF

IG

_N

ET

FI

LT

ER

_X

T_

MA

TC

H_

BP

F

BPF(BSD Packet Filter)是一个强大的包匹配模块,用于匹配那些让过滤器返回非零值的包.

"

c

l

u

s

t
e
r
"
m
a
t
c
h
s
u
p
p
o
r
t
C
O
N
F
I
G

INETFILTEREXTIMATCHICL

U

S

T

E

R

这个模块可以用于创建网络服务器/防火墙集群,而无需借助价格昂贵的负载均衡设备.通常,在包必须被本节点处理的条件下,这个 match 返回"true".这样,所有节点都可以看到所有的包,但只有匹配的节点才需要进行处理,这样就将负载进行了分摊.而分摊算法是基于对源地址的哈希值.

"

c

o

m

m

e

n

t

"

m

a

t

c
h
s
u
p
p
o
r
t
C
O
N
F
I
G
_
N
E
T
F
I
L

TERM_XT_MATCH_COMMENT

这是一个"伪 match",目的是允许你在 iptables 规则集中加入注释

"
c
o
n
n
b
y
t
e
s
"
p
e
r
-
c
o
n
n
e
c
t

i
o
n
c
o
u
n
t
e
r
m
a
t
c
h
s
u
p
p
o
r
t

CONFIG_NETFILTER_XT_MA

MATCH_CONNBYTES

允许针对单个连接内部每个方向(进/出)匹配已经传送的字节数/包数

"connla

b
e
l
"
m
a
t
c
h
s
u
p
p
o
r
t
C
O
N
F
I
G

_NETFILTER_EXT_MATCH_CO

NNLABEL

允许向连接分配用户自定义的标签名.内核仅存储 bit 值,而名称和 bit 之间的对应关系由用户空间处理.与"connmark"的不同之处在于:可以同时为一个连接分配 32 个标志位(flag bit).

"connlimit"

matchsupportCONFIG_NET

FILTER_XT_MATCH_CONNLI

M

I

T

允许根据每一个客户端 IP 地址(或每一段客户端 IP 地址段)持有的并发连接数进行匹配.

"

c

o

n

n

m

a

r

k

"

c

o

n

n

e

c

t

i
o
n
m
a
r
k
m
a
t
c
h
s
u
p
p
o
r
t
C
O
N

FIG_NETFILTER_XT_MATCH

_CONNMARK

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_CONNMARK

"conntrack"

connectiontrackingmatc

h
s
u
p
p
o
r
t
C
O
N
F
I
G
_
N
E
T
F
I
L
T

ERROR_TEXT_MATCH_CONTRACT

通用连接跟踪匹配,是"state"的超集,它允许额外的链接跟踪信息,在需要设置一些复杂的规则(比如网关)时很有用.Docker 依赖于它.

"cpu"match cshsupportpportCO

NFIG_NETFILTER_XT_MATC

H

_

C

P

U

根据处理包所使用的 CPU 是哪个进行匹配

DCCP 是打算取代 UDP 的新传输协议,它在 UDP 的基础上增加了流控和拥塞控制机制,面向实时业务

允许根据网卡所属的"设备组"进行匹配

dscp match 允许根据 IPv4/IPv6 包头的 DSCP 字段进行匹配，

tos match 允许根据 IPv4 包头的 TOS 字段进行匹配

允许根据 IPv4 TCP 包头的 ECN 字段进行匹配

允许对 IPSec 包的 ESP 头中的 SPI(安全参数序列)范围进行匹配

此项的目的是取代"limit",它基于你选定的源/目的地址和/或端口动态创建"limit bucket"哈希表.这样你就可以迅速创建类似这样的匹配规则:(1)为给定的目的地址以每秒 10k 个包的速度进行匹配;(2)为给定的源地址以每秒 500 个包的速率进行匹配

加载特定协议的连接跟踪辅助模块,由该模块过滤所跟踪的连接类型的包,比如 ip_conntrack_ftp 模块

基于 IPv6 包头的 hoplimit 字段,或 IPv4 包头的 time-to-live 字段进行匹配

根据 IP 地址范围进行匹配,而普通的 iptables 只能根据"IP/mask"的方式进行匹配.

允许根据包的 IPVS 属性进行匹配

允许对包的长度进行匹配

允许根据包的进出速率进行规则匹配,常和"LOG target"配合使用以抵抗某些 Dos 攻击

允许根据以太网的 MAC 地址进行匹配

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_MARK

允许对 TCP 或 UDP 包同时匹配多个不连续的端口(通常情况下只能匹配单个端口或端口范围)

允许通过 nfnetlink_acct 使用扩展记账

开启 Passive OS Fingerprinting 模块,以允许通过进入的 TCP SYN 包被动匹配远程操作系统.规则和加载程序可以从这里获取:http://www.ioremap.net/projects/osf

基于创建套接字的本地进程身份(user/group)进行匹配,还可以用于检查一个套接字是否确实存在

基于 IPsec policy 进行匹配

允许对进入或离开所经过的物理网口进行匹配

允许对封包目的地址类别(广播/组播/直播)进行匹配

允许对总字节数的限额值进行匹配

根据 RATEEST target 评估的速率值进行匹配

允许根据 iptables 中的路由子系统中的 realm 值进行匹配.它与 tc 中的 CONFIG_NET_CLS_ROUTE4 非常类似.

recent match 用于创建一个或多个最近使用过的地址列表,然后又可以根据这些列表再进行匹配.

支持根据流控制传输协议(SCTP)源/目的端口和"chunk type"进行匹配.

can be used to match packets for which a TCP or UDP socket lookup finds a valid socket. It can be used in combination with the MARK target and policy routing to implement full featured non-locally bound sockets.

这是对包进行分类的有力工具,它允许利用连接跟踪信息对连接中处于特定状态的包进行匹配

允许根据一个给定的百分率对包进行周期性的或随机性的匹配

允许根据包所承载的数据中包含的特定字符串进行匹配

允许根据 TCP SYN 包头中的 MSS(最大分段长度)选项的值进行匹配

根据包的到达时刻(外面进入的包)或者离开时刻(本地生成的包)进行匹配

"u32"允许从包中提取拥有特定 mask 的最多 4 字节数据,将此数据移动(shift)特定的位数,然后测试其结果是否位于特定的集合范围内.更多细节可以直接参考内核源码(net/netfilter/xt_u32.c)

IP set support

CONFIG_IP_SET

为内核添加 IP 集(IP set)支持,然后就可以使用 CONFIG_NETFILTER_XT_SET 功能.此特性必须配合用户态工具 ipset 一起使用.

Maximum number of IP sets

CONFIG_IP_SET_MAX

默认的最大"set"数,取值范围是[2,65534].此值也可以由ip_set 模块的 max_sets 参数设置.

bitmap:ip set support

CONFIG_IP_SET_BITMAP_IP

"bitmap:ip"集合类型.根据 IP 地址范围设定集合.

bitmap:ip,mac set support

CONFIG_IP_SET_BITMAP_IPMAC

"bitmap:ip,mac"集合类型.根据 IP/MAC 地址对范围设定集合.

bitmap:port set support

CONFIG_IP_SET_BITMAP_PORT

"bitmap:port"集合类型.根据端口范围设定集合.

hash:ip set support

CONFIG_IP_SET_HASH_IP

"hash:ip"集合类型.为多个离散的 IP 地址设定集合.

hash:ip,port set support

CONFIG_IP_SET_HASH_IPPORT

"hash:ip,port"集合类型.为多个离散的 IP/MAC 地址对设定集合.

hash:ip,port,ip

CONFIG_IP_SET_HASH_IPPORTIP

"hash:ip,port,ip"集合类型.为多个离散的 IP/端口/IP 三元组设定集合.

hash:ip,port,net set support

CONFIG_IP_SET_HASH_IPPORTNET

"hash:ip,port,net"集合类型.为多个离散的 IP/端口/网段三元组设定集合.

hash:net set support

CONFIG_IP_SET_HASH_NET

"hash:net"集合类型.为多个离散的网段设定集合

hash:net,port set support

CONFIG_IP_SET_HASH_NETPORT

"hash:net,port"集合类型.为多个离散的网段/端口对设定集合

hash:net,iface set support

CONFIG_IP_SET_HASH_NETIFACE

"hash:net,iface"集合类型.为多个离散的网段/网卡接口对设定集合

list:set set support

CONFIG_IP_SET_LIST_SET

"list:set"集合类型.将多个集合组成一个更大的集合

IP virtual server support

CONFIG_IP_VS

IPVS(IP Virtual Server)支持.IPVS 可以帮助 LVS 基于多个后端真实服务器创建一个高性能的虚拟服务器.可以使用三种具体的方法实现:NAT,隧道,直接路由(使用较广).

IPv6 support for IPVS

CONFIG_IP_VS_IPV6

为 IPVS 添加 IPv6 支持

IP virtual server debugging

CONFIG_IP_VS_DEBUG

为 IPVS 添加调试支持

IPVS connection table size (the Nth power of 2)

CONFIG_IP_VS_TAB_BITS

设置 IPVS 连接哈希表的大小($2^{CONFIG\_IP\_VS\_TAB\_BITS}$),取值范围是[8,20],默认值 12 的意思是哈希表的大小是 2^{12}=4096 项.IPVS 连接哈希表使用链表来处理哈希碰撞.使用大的哈希表能够显著减少碰撞几率,特别是哈希表中有成千上万连接的时候.比较恰当的值差不多等于每秒的新建连接数乘以每个连接的平均持续秒数.太小的值会造成太多碰撞,从而导致性能大幅下降;太大的值

又会造成占用太多不必要的内存(每个表项 8 字节+每个连接 128 字节).该值也可以通过 ip_vs 模块的 conn_tab_bits 参数进行设置.

TCP load balancing support

CONFIG_IP_VS_PROTO_TCP

TCP 传输协议负载均衡支持

UDP load balancing support

CONFIG_IP_VS_PROTO_UDP

UDP 传输协议负载均衡支持

ESP load balancing support

CONFIG_IP_VS_PROTO_ESP

IPSec ESP(Encapsulation Security Payload)传输协议负载均衡支持

AH load balancing support

CONFIG_IP_VS_PROTO_AH

IPSec AH(Authentication Header)传输协议负载均衡支持.

SCTP load balancing support

CONFIG_IP_VS_PROTO_SCTP

SCTP 传输协议负载均衡支持

round-robin scheduling

CONFIG_IP_VS_RR

循环分散算法:最简单的调度算法,将连接简单的循环分散到后端服务器上

weighted round-robin scheduling

CONFIG_IP_VS_WRR

基于权重的循环分散算法:在循环分散的基础上,权重较高的后端服务器接纳较多的连接

least-connection scheduling

CONFIG_IP_VS_LC

最少连接算法:将连接优先分配到活动连接最少的后端服务器

weighted least-connection scheduling

CONFIG_IP_VS_WLC

基于权重的最少连接算法:结合考虑活动连接数与服务器权重

locality-based least-connection scheduling

CONFIG_IP_VS_LBLC

基于目的 IP 的最少连接算法(常用于缓存集群):优先根据目的 IP 地址将连接分配到特定的后端,仅在这些后端过载时(活动连接数大于其权重)才分散到其他后端.

locality-based least-connection with replication

scheduling

CONFIG_IP_VS_LBLCR

与 LBLC 类似,不同之处在于:前端负载均衡器会像 NAT 一样同时记住客户端 IP 与后端的对应关系,并在新的连接到来的时候,复用这个对应关系.

destination hashing scheduling

CONFIG_IP_VS_DH

目标地址哈希表算法:简单的根据静态设定的目标 IP 地址哈希表将连接分发到后端

source hashing scheduling

CONFIG_IP_VS_SH

源地址哈希表算法:简单的根据静态设定的源 IP 地址哈希表将连接分发到后端

shortest expected delay scheduling

CONFIG_IP_VS_SED

最小期望延迟算法:将连接分配到根据期望延迟公式((Ci+1)/Ui)算得的延迟最小的后端."i"是后端服务器编号,"Ci"是该服务器当前的连接数,"Ui"是该服务器的权重.

never queue scheduling

CONFIG_IP_VS_NQ

无排队算法:这是一个两阶段算法,如果有空闲服务器,就直接分发到空闲服务器(而不是等待速度最快的服务器),如果没有空闲服务器,就分发到期望延迟最小的服务器(SED 算法).

IPVS source hashing table size (the Nth power of 2)

CONFIG_IP_VS_SH_TAB_BITS

将源 IP 地址映射到后端服务器所使用的哈希表的大小($2^{CONFIG\_IP\_VS\_SH\_TAB\_BITS}$),取值范围是[4,20],默认值 8 的意思是哈希表的大小是 2^8=256 项.理想的大小应该是所有后端的权重乘以后端总数?

FTP protocol helper

CONFIG_IP_VS_FTP

FTP 协议连接追踪帮助

Netfilter connection tracking

CONFIG_IP_VS_NFCT

Netfilter 连接追踪支持

SIP persistence engine

CONFIG_IP_VS_PE_SIP

基于 SIP Call-ID 提供持久连接支持

IP: Netfilter Configuration

针对 IPv4 的 Netfilter 配置

IPv4 connection tracking support (required for NAT)

CONFIG_NF_CONNTRACK_IPV4

IPv4 链接跟踪.可用于包伪装或地址转换,也可用于增强包过滤能力

proc/sysctl compatibility with old connection tracking

CONFIG_NF_CONNTRACK_PROC_COMPAT

用于兼容老旧的连接追踪用户态程序

IP tables support (required for filtering/masq/NAT)

CONFIG_IP_NF_IPTABLES

要用 iptables 就肯定要选上

"ah" match support

CONFIG_IP_NF_MATCH_AH

允许对 IPSec 包头的 AH 字段进行匹配

"ecn" match support

CONFIG_IP_NF_MATCH_ECN

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_MATCH_ECN

"rpfilter" reverse path filter match support

CONFIG_IP_NF_MATCH_RPFILTER

对进出都使用同一个网络接口的包进行匹配

"ttl" match support

CONFIG_IP_NF_MATCH_TTL

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_MATCH_HL

Packet filtering

CONFIG_IP_NF_FILTER

定义 filter 表,以允许对包进行过滤.Docker 依赖于它.

REJECT target support

CONFIG_IP_NF_TARGET_REJECT

允许返回一个 ICMP 错误包而不是简单的丢弃包

ULOG target support

CONFIG_IP_NF_TARGET_ULOG

反对使用该选项,因为它已经被

CONFIG_NETFILTER_NETLINK_LOG 代替

IPv4 NAT

CONFIG_NF_NAT_IPV4

允许进行伪装/端口转发以及其它的 NAT 功能,仅在你需要使用

iptables 中的 nat 表时才需要选择.Docker 依赖于它.

MASQUERADE target support

CONFIG_IP_NF_TARGET_MASQUERADE

SNAT 是指在数据包从网卡发送出去的时候,把数据包中的源地址部分替换为指定的 IP,这样,接收方就认为数据包的来源是被替换的那个 IP 的主机.伪装(MASQUERADE)是一种特殊类型的 SNAT:MASQUERADE 是用发送数据的网卡上的 IP 来替换源 IP,用于那些 IP 不固定的场合(比如拨号或者通过 DHCP 分配).Docker 依赖于它.

NETMAP target support

CONFIG_IP_NF_TARGET_NETMAP

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_TARGET_NETMAP.

REDIRECT target support

CONFIG_IP_NF_TARGET_REDIRECT

这只是一个兼容旧配置的选项,等价于

CONFIG_NETFILTER_XT_TARGET_REDIRECT.

Basic SNMP-ALG support

CONFIG_NF_NAT_SNMP_BASIC

为 SNMP 荷载实现 ALG(Application Layer Gateway)支持

(RFC2962).

Packet mangling

CONFIG_IP_NF_MANGLE

在 iptables 中启用 mangle 表以便对包进行各种修改,常用于

改变包的路由

CLUSTERIP target support

CONFIG_IP_NF_TARGET_CLUSTERIP

CLUSTERIP target 允许你无需使用昂贵的负载均衡设备也能

创建廉价的负载均衡集群

ECN target support

CONFIG_IP_NF_TARGET_ECN

用于 mangle 表,可以去除 IPv4 包头的 ECN(Explicit

Congestion Notification)位,主要用于在保持 ECN 功能的前

提下,去除网络上的"ECN 黑洞".

"TTL" target support

CONFIG_IP_NF_TARGET_TTL

这只是一个兼容旧配置的选项,等价于 CONFIG_NETFILTER_XT_TARGET_HL.

raw table support (required for NOTRACK/TRACE)

CONFIG_IP_NF_RAW

在 iptables 中添加一个 raw 表,该表在 netfilter 框架中非常靠前,并在 PREROUTING 和 OUTPUT 链上有钩子,从而可以对收到的数据包在连接跟踪前进行处理

Security table

CONFIG_IP_NF_SECURITY

在 iptables 中添加一个 security 表,以支持强制访问控制(Mandatory Access Control)策略

ARP tables support

CONFIG_IP_NF_ARPTABLES

arptables 支持

ARP packet filtering

CONFIG_IP_NF_ARPFILTER

ARP 包过滤.对于进入和离开本地的 ARP 包定义一个 filter 表,在桥接的情况下还可以应用于被转发的 ARP 包

ARP payload mangling

CONFIG_IP_NF_ARP_MANGLE

允许对 ARP 包的荷载部分进行修改,比如修改源和目标物理地址

IPv6: Netfilter Configuration

针对 IPv6 的 Netfilter 配置.其子项内容类似于 IPv4,需要的话可以参考前面 IPv4 的 Netfilter 配置进行选择

DECnet: Netfilter

Configuration

针对 DECnet 的 Netfilter 配置

Ethernet Bridge tables

(ebtables) support

CONFIG_BRIDGE_NF_EBTABLES

针对以太网桥的 ebtables Netfilter 配置

The DCCP Protocol

CONFIG_IP_DCCP

数据报拥塞控制协议(Datagram Congestion Control Protocol)在 UDP 的基础上增加了流控和拥塞控制机制,使数据报协议能够更好地用于流媒体业务的传输

The SCTP Protocol

CONFIG_IP_SCTP

流控制传输协议(Stream Control Transmission Protocol)是一种新兴的传输层协议.TCP 协议一次只能连接一个 IP 地址而在 SCTP 协议一次可以连接多个 IP 地址且可以自动平衡网络负载,一旦某一个 IP 地址失效会自动将网络负载转移到其他 IP 地址上

The RDS Protocol

CONFIG_RDS

可靠数据报套接字(Reliable Datagram Sockets)协议支持.RDS 可以使用 Infiniband 和 iWARP 作为支持 RDMA(远程直接内存访问)的传输方式,RDMA 用于一台远程计算机访问另一台计算机的内存而无需本机计算机操作系统的辅助,这就像直接内存访问(DMA),但是这里远程代替了本地计算机.

The TIPC Protocol

CONFIG_TIPC

透明内部进程间通信协议(Transparent Inter Process Communication),以共享内存为基础实现任务和资源的调度,专门用于集群内部通信

Asynchronous Transfer Mode (ATM)

CONFIG_ATM

异步传输模式(ATM)支持.主要用于高速 LAN 和 WAN.目前已经日薄西山了.

Layer Two Tunneling Protocol (L2TP)

CONFIG_L2TP

第二层隧道协议(RFC2661)是一种对应用透明的隧道协议,VPN 经常使用它.

802.1d Ethernet Bridging

CONFIG_BRIDGE

802.1d 以太网桥(例如为 QEMU 虚拟机或 Docker 容器提供桥接网卡支持)

IGMP/MLD snooping

CONFIG_BRIDGE_IGMP_SNOOPING

选"Y"可以允许以太网桥根据 IGMP(Internet Group Management Protocol, IPv4)/MLD(Multicast Listener Discovery, IPv6)负载选择性的转发不同端口上的多播包.选"N"可以减小二进制文件的体积.确定需要使用组播的选"Y".

802.1Q/802.1ad VLAN Support CONFIG_VLAN_8021Q

802.1Q 虚拟局域网

DECnet Support CONFIG_DECNET

DECnet 协议

ANSI/IEEE 802.2 LLC type 2 Support CONFIG_LLC2

PF_LLC 类型套接字支持.也就是 IEEE 802.2 LLC 2

The IPX

proto

col

CONFI

G_IPX

IPX 协议是由 Novell 公司提出的运行于 OSI 模型第三层的协议,具有可路由的特性,IPX 的地址分为网络地址和主机地址,网络地址由管理员分配,主机地址为 MAC 地址.由于 IP 协议的广泛使用,IPX 的应用早已日薄西山.

Appl

etal

k

prot

ocol

supp

ort

CONF

IG_A

TALK

Appletalk 是苹果公司创建的一组网络协议,仅用于苹果系列计算机.

CC

IT

TX.25PacketLayer

CONFIG_X25

CCITT X.25协议集支持.

LAPBData

aLinkDriverCONFIG_LAPB

LAPB 协议支持.

PhonetprotocolsfamilY

CONFIG_PHONET

PhoNet 是 Nokia 开发的面相数据包的通信协议,仅用于 Nokia maemo/meego 产品.

IEEEStd

802.15.4 Low-Rate Wireless

s

s

P

e

r

s

o

n

a

l

A

r

e

a

N

e

t

w

o

r

k

s

s
u
p
p
o
r
t
C
O
N
F
I
G
_
I
E
E
E
8
0
2
1

5

4

IEEE Std 802.15.4 定义了一个低速率/低功耗/低复杂度的短距离个人无线网络规范.主要用于物联网中的传感器/交换器之类设备之间的互联.

6lowpan support over IEEE 802.15.4

CONFIG_IEEE802154_6LOWPAN

在 IEEE 802.15.4 上支持 IPv6 压缩.

Generic IEEE 802.15.4 Soft Networking Stack (mac802154)

CONFIG_MAC802154

为 SoftMAC 设备(仅实现了 PHY 层)实现硬件独立的 IEEE Std 802.15.4 协议栈.使用 HardMAC 设备的用户应该选"N".[注意]这里的实现既未经过认证,也未进行充分的兼容性测试.

Q

o

S

a

n

d

/

o

rfairqueueingCONFIG_NE

T
_
S
C
H
E
D

QoS(Quality of Service)支持.当内核有多个包需要通过网络发送的时候,它需要决定哪个包先发,那个包后发,哪个包丢弃.这就是包调度算法.关闭此项表示内核使用最简单的 FIFO 算法,开启此项后就可以使用多种不同的调度算法(需要配合用户层工具 iproute2+tc).QoS 还用于支持 diffserv(Differentiated Services)和 RSVP(Resource Reservation Protocol)功能.包调度的状态信息可以从"/proc/net/psched"文件中获取.仅在你确实需要的时候选"Y".

D
a
t
a
C
e

n
t
e
r
B
r
i
d
g
i
n
g
s
u
p
p
o
r
t
C
O
N

FIG_DCB

DCB(Data Center Bridging)支持.数据中心桥接是一组可增强传统以太网功能,以管理通信的功能,尤其适用于网络通信流量和传输率都很高的环境中.光纤通道可专用于承载此类型的通信.但是,如果使用专用链路来仅提供光纤通道通信,则成本可能会很高.因此,更多情况下使用以太网光纤通道.DCB 功能可满足光纤通道对遍历以太网时包丢失的敏感度要求.DCB 允许对等方基于优先级区分通信.通过区分优先级,可确保在主机之间发生拥塞时,保持较高优先级通信的包完整性.使用 DCB 交换协议,通信主机可以交换会影响高速网络通信的配置信息.然后,对等方可对公用配置进行协商,确保通信流不中断,同时防止高优先级包出现包丢失.这些功能都需要底层的网卡支持.一般网卡都是不支持的.所以不确定的可以选"N".

DNS

ResolverSupportConfig_

DNS_RESOLVER

内核 DNS 解析支持.用于支持 CONFIG_AFS_FS/CONFIG_CIFS/CONFIG_CIFS_SMB2/NFS_V4 模块.此功能需要用户态程序"/sbin/dns.resolve"和配置文件"/etc/request-key.conf"的支持.更多信息参见"Documentation/networking/dns_resolver.txt"文档,不确定的选"N".

B.A.

T.M.A.N.Advanced Meshin

gProtocolCONFIG_BATMAN

B.A.T.M.A.N.(更好的移动无线网络方案)是一种用于 multi-hop ad-hoc mesh 网络的路由协议.它是一种去中心化分布式无线 Adhoc 模式,特别适用于自然灾害等紧急情况下,创建临时的无线网络.不确定的选"N".

Open vSwitch 是一个多层虚拟交换标准.此选项提供了内核级的高速转发功能(需要配合用户态守护进程 ovs-vswitchd 来实现).

这是一个类似于 TCP/IP 的协议,用于虚拟机之间以及虚拟机与宿主之间的通信.开启此项后,还需要从子项中选择适用于特定虚拟化技术的传输协议.

VMware VMCI transport for Virtual Sockets

CONFIG_VMWARE_VMCI_VSOCKETS

适用于 VMware 虚拟化技术的 VMCI 传输协议支持.

基于内存映射机制的 netlink IO 支持.可以避免在用户空间与内存空间之间复制数据,从而提升操作速度.不确定的选"N".

NETLINK socket 监视接口.ss 这样的诊断工具需要它.

多协议标签交换(MPLS)是新一代的 IP 高速骨干网络交换标准.

不确定的选"N".

以太网 HSR(高可用性无缝冗余)规范(IEC 62439-3:2010)支持.不确定的选"N".

Cgroup 子系统支持:基于每个网络接口为每个进程分配网络使用优先级.Docker 依赖于它.

BPF(Berkeley Packet Filter)的过滤功能通常由一个解释器(interpreter)解释执行 BPF 虚拟机指令的方式工作.开启此项,内核在加载过滤指令后,会将其编译为本地指令,以加快执行

速度.网络嗅探程序(libpcap/tcpdump)可以从中受益.注意:需要"echo 1 > /proc/sys/net/core/bpf_jit_enable"之后才能生效.

网络测试,仅供调试使用

Amateur Radio support

CONFIG_HAMRADIO

业余无线电支持.供无线电爱好者进行自我训练/相互通讯/技术研究

CAN bus subsystem support

CONFIG_CAN

CAN(Controller Area Network)是一个低速串行通信协议.被广泛地应用于工业自动化/船舶/医疗设备/工业设备等嵌入式领域.更多信息参见"Documentation/networking/can.txt"文件.

IrDA (infrared) subsystem support

CONFIG_IRDA

红外线通讯技术支持,主要用于嵌入式环境,某些老旧的笔记本上也可能会有红外接口.

Bluetooth subsystem support

CONFIG_BT

蓝牙支持.蓝牙目前已经基本取代红外线,成为嵌入式设备/智能设备/笔记本的标配近距离(小于 10 米)通信设备.在 Linux 上通常使用来自 BlueZ 的 hciconfig 和 bluetoothd 工具操作蓝牙通信.

RFCOMM protocol support

CONFIG_BT_RFCOMM

虚拟串口协议(RFCOMM)是一个面向连接的流传输协议,提供 RS232 控制和状态信号,从而模拟串口的功能.它被用于支持拨号网络,OBEX(Object Exchange),以及某些蓝牙程序(例如文件传输).

RFCOMM TTY support

CONFIG_BT_RFCOMM_TTY

允许在 RFCOMM 通道上模拟 TTY 终端

BNEP protocol support

CONFIG_BT_BNEP

蓝牙网络封装协议(Bluetooth Network Encapsulation Protocol)可以在蓝牙上运行其他网络协议(TCP/IP). Bluetooth PAN(Personal Area Network)需要它的支持.

Multicast filter support

CONFIG_BT_BNEP_MC_FILTER

组播支持

Protocol filter support

CONFIG_BT_BNEP_PROTO_FILTER

协议过滤器支持

CMTP protocol support

CONFIG_BT_CMTP

CMTP(CAPI 消息传输协议)用于支持已在上世纪被淘汰的 ISDN 设备.不确定的选"N".

HIDP protocol support

CONFIG_BT_HIDP

人机接口设备协议(Human Interface Device Protocol)用于支持各种人机接口设备(比如鼠标/键盘/耳机等).

Bluetooth device drivers

各种蓝牙设备驱动

HCI USB driver

CONFIG_BT_HCIBTUSB

使用 USB 接口的蓝牙设备支持

HCI SDIO driver

CONFIG_BT_HCIBTSDIO

使用 SDIO 接口的蓝牙设备支持

HCI UART driver

CONFIG_BT_HCIUART

使用串口的蓝牙设备支持.此外,基于 UART 的蓝牙 PCMCIA 和 CF 设备也需要此模块的支持.

UART (H4) protocol support

CONFIG_BT_HCIUART_H4

大多数使用 UART 接口的蓝牙设备(包括 PCMCIA 和 CF 卡)都使用这个协议.

BCSP protocol support

CONFIG_BT_HCIUART_BCSP

基于 CSR(Cambridge Silicon Radio)公司的 BlueCore 系列芯片的蓝牙设备(包括 PCMCIA 和 CF 卡)支持

Atheros AR300x serial support

CONFIG_BT_HCIUART_ATH3K

基于 Atheros AR300x 系列芯片的蓝牙设备支持

HCILL protocol support

CONFIG_BT_HCIUART_LL

基于 Texas Instruments 公司的 BRF 芯片的蓝牙设备支持

Three-wire UART (H5) protocol support

CONFIG_BT_HCIUART_3WIRE

Three-wire UART (H5) 协议假定 UART 通信可能存在各种错误,从而使得 CTS/RTS 引脚线变得可有可无.看不懂就可以不选.

HCI VHCI (Virtual HCI device) driver

CONFIG_BT_HCIVHCI

模拟蓝牙设备支持.主要用于开发

{大多数蓝牙设备并不需要特定的独立驱动,此处省略的独立驱动仅是为了驱动那些不严格遵守蓝牙规范的芯片}

RxRPC session sockets

CONFIG_AF_RXRPC

RxRPC 会话套接字支持(仅包括传输部分,不含表示部分).CONFIG_AFS_FS 依赖于它.不确定的选"N".详情参见"Documentation/networking/rxrpc.txt"文档.

Wireless

CONFIG_WIRELESS

无线网络支持.

cfg80211 - wireless configuration API

CONFIG_CFG80211

cfg80211 是 Linux 无线局域网(802.11)配置接口,是使用 WiFi 的前提.注意:"WiFi"是一个无线网路通信技术的品牌,由 WiFi 联盟所持有.目的是改善基于 IEEE 802.11 标准的无线网路产品之间的互通性.现时一般人会把 WiFi 及 IEEE 802.11 混为一谈,甚至把 WiFi 等同于无线网路(WiFi 只是无线网络的一种).

nl80211 testmode command

CONFIG_NL80211_TESTMODE

仅供调试和特殊目的使用.

enable developer warnings

CONFIG_CFG80211_DEVELOPER_WARNINGS

仅供调试开发使用

cfg80211 regulatory debugging

CONFIG_CFG80211_REG_DEBUG

仅供调试开发使用

cfg80211 certification onus

CONFIG_CFG80211_CERTIFICATION_ONUS

仅在你确实明白此项含义的情况下,才考虑选"Y",否则请选"N".

enable powersave by default

CONFIG_CFG80211_DEFAULT_PS

若开启此项则表示默认开启省电模式(也就是默认"Soft blocked: yes").关闭此项则表示默认使用 BIOS 中的状态(通常是上一次关机时的状态).详情参见"Documentation/power/pm_qos_interface.txt"文档.

cfg80211 DebugFS entries

CONFIG_CFG80211_DEBUGFS

仅供调试

use statically compiled regulatory rules database

CONFIG_CFG80211_INTERNAL_REGDB

由于绝大多数发行版都含有 CRDA 软件包,所以绝大多数人应该选"N".如果你确实需要选"Y",那么请认真阅读"net/wireless/db.txt"文件.

cfg80211 wireless extensions compatibility

CONFIG_CFG80211_WEXT

为那些老旧的用户空间程序提供兼容性,建议关闭.

lib80211 debugging messages

CONFIG_LIB80211_DEBUG

仅供调试

Generic IEEE 802.11 Networking Stack (mac80211)

CONFIG_MAC80211

独立于硬件的通用 IEEE 802.11 协议栈模块(mac80211).它是驱动开发者用来编写 softMAC 无线设备驱动的框架,softMAC 设备允许用软件实现帧的管理(包括解析和产生 80211 无线帧),从而让系统能更好的控制硬件,现在大多数的无线网卡都是 softMAC 设备.不确定的选"Y".

PID controller based rate control algorithm

CONFIG_MAC80211_RC_PID

基于比例-积分-微分控制器(PID controller)的发送速率(TX rate)控制算法.用于 CONFIG_MAC80211 模块.不确定的选"N".

Minstrel

CONFIG_MAC80211_RC_MINSTREL

minstrel 发送速率(TX rate)控制算法.用于 CONFIG_MAC80211 模块.这是首选的算法,不确定的选"Y".

Minstrel 802.11n support

CONFIG_MAC80211_RC_MINSTREL_HT

minstrel_ht 发送速率(TX rate)控制算法.适用于 802.11n 规范.不确定的选"Y".

Default rate control algorithm

默认发送速率(TX rate)控制算法.相当于 mac80211 模块 "ieee80211_default_rc_algo"参数的值.建议选择 "Minstrel"算法.

Enable mac80211 mesh networking (pre-802.11s) support

CONFIG_MAC80211_MESH

802.11s 草案是无线网状网络(Mesh Networking)的延伸与增补标准(amendment).它扩展了 IEEE 802.11 MAC(介质访问控制)标准,定义了利用自我组态的多点跳跃拓朴(multi-hop topologies),进行无线感知(radio-aware metrics),以支援广播/组播/单播传送网络封包的架构与协定.不确定的选"N".

Enable LED triggers

CONFIG_MAC80211_LEDS

允许在接受/发送数据时触发无线网卡的 LED 灯闪烁.

Export mac80211 internals in DebugFS

CONFIG_MAC80211_DEBUGFS

在 DebugFS 中显示 mac80211 模块内部状态的扩展信息,仅用于调试目的.

Trace all mac80211 debug messages

CONFIG_MAC80211_MESSAGE_TRACING

跟踪所有 mac80211 模块的调试信息,仅用于调试目的.

Select mac80211 debugging features

CONFIG_MAC80211_DEBUG_MENU

仅供调试

WiMAX Wireless Broadband support

CONFIG_WIMAX

WiMAX(IEEE 802.16)协议支持.随着 2010 年英特尔放弃 WiMAX 以及 LTE 在 4G 市场成了唯一的主流标准,WiMAX 的电信运营商也逐渐向 LTE 转移,WiMAX 论坛也于 2012 年将 TD-LTE 纳入 WiMAX2.1 规范,一些 WiMAX 运营商也开始将设备升级为 TD-LTE.

WiMAX debug level

CONFIG_WIMAX_DEBUG_LEVEL

设置允许使用的最大调试信息详细等级,推荐使用默认值"8",设为"0"表示允许使用所有调试信息.运行时默认禁止使用调试信息,但可通过 sysfs 文件系统中的 debug-levels 文件开启调试信息.

RF switch subsystem support

CONFIG_RFKILL

为了节约电力,很多无线网卡和蓝牙设备都有内置的射频开关(RF switche)用于开启和关闭设备(通过 rfkill 命令).建议选"Y".更多详情参见"Documentation/rfkill.txt"文档

RF switch input support

CONFIG_RFKILL_INPUT

这是个反对使用的特性,一般情况下建议关闭.若关闭此项导致某些笔记本的无线网卡开关按钮失效,可以考虑开启.

Generic rfkill regulator driver

CONFIG_RFKILL_REGULATOR

通用射频开关驱动,其射频开关连接在电压调节器(voltage regulator)上.依赖于 CONFIG_REGULATOR 框架.不确定的选"N"或"M"

GPIO RFKILL driver

CONFIG_RFKILL_GPIO

通用 GPIO 射频开关驱动.仅用于嵌入式环境,其射频开关连接在 GPIO 总线上,比如 NVIDIA 的 Tegra 和三星的 Exynos 4 智能手机 SoC 芯片.

Plan 9 Resource Sharing Support (9P2000)

CONFIG_NET_9P

实验性的支持 Plan 9 的 9P2000 协议.

CAIF support

CONFIG_CAIF

除非你为 Android/MeeGo 系统编译内核,并且需要使用 PF_CAIF 类型的 socket,否则请选"N".

Ceph core library

CONFIG_CEPH_LIB

仅在你需要使用 Ceph 分布式文件系统,或者 rados 块设备(rbd)时选"Y".否则应选"N".

NFC subsystem support

CONFIG_NFC

NFC(近场通信)子系统.这些设备主要用于智能手机之类的嵌入式领域.

Network light weight tunnels

CONFIG_LWTUNNEL

为 MPLS(多协议标签交换)之类的轻量级隧道提供基础结构支持.不确定的选"N".

Device Drivers

设备驱动程序

Generic Driver Options

驱动程序通用选项.[提示]Linux Kernel Driver DataBase 网站是搜索驱动程序与硬件型号对应关系的绝佳网站.如果你不知道某个驱动(例如"CONFIG_INTEL_IOATDMA")究竟对应着哪些型号的硬件,那么可以直接根据该驱动选项的首字母(本例是"I")进入对应的索引页去查找到该驱动的详情页面(本例是"https://cateee.net/lkddb/web-lkddb/INTEL_IOATDMA.html").[提示]可以使用"lspci -nn"与"lsusb"命令查看本机所有 PCI/USB 设备的"vendor id"与"device id"及文本名称.也可以根据已知的 id 到 pci.ids 与 usb.ids 数据库中搜索设备的名称.

Support for uevent helper

CONFIG_UEVENT_HELPER

早年的内核(切换到基于 netlink 机制之前),在发生 uevent 事件(通常是热插拔)时,需要调用用户空间程序(通常是

"/sbin/hotplug"),以帮助完成 uevent 事件的处理.此选项就是用于开启此功能.由于目前的发行版都已不再需要此帮助程序,所以请选"N".此外,如果你使用了 systemd 或 udev 则必须选"N".

path to uevent helper

CONFIG_UEVENT_HELPER_PATH

早年的内核(切换到基于 netlink 机制之前),在发生 uevent 事件(通常是热插拔)时,需要调用用户空间程序(通常是"/sbin/hotplug"),以帮助完成 uevent 事件的处理.此选项就是用于设定这个帮助程序的路径.由于目前的发行版都已不再需要此帮助程序,所以请保持空白.此外,如果你使用了 systemd 或 udev 则必须保持空白.

Maintain a devtmpfs filesystem to mount at /dev

CONFIG_DEVTMPFS

devtmpfs 是一种基于 CONFIG_TMPFS 的文件系统(与 proc 和 sys 有几分相似).在系统启动过程中,随着各个设备的初始化完成,内核将会自动在 devtmpfs 中创建相应的设备节点(使用默认的文件名和权限)并赋予正确的主次设备号.更进一步,在系统运行过程中,随着各种设备插入和拔除,内核也同样会自动在 devtmpfs 中创建和删除的相应的设备节点(使用默认的文件名和权限)并赋予正确的主次设备号.如果将 devtmpfs 挂载到

"/dev"目录(通常是系统启动脚本),那么便拥有了一个全自动且全功能的"/dev"目录,而且用户空间程序(通常是 udevd)还可以对其中的内容进行各种修改(增删节点,改变权限,创建符号链接).目前的发行版和各种嵌入式系统基本都依赖于此,除非你知道自己在做什么,否则请选"Y".

```
Automount devtmpfs at /dev, after the kernel
mounted the rootfs
CONFIG_DEVTMPFS_MOUNT
```

在内核挂载根文件系统的同时,立即自动将 devtmpfs 挂载到"/dev"目录.因为此时 init 进程都还尚未启动,所以这就确保在进入用户空间之前,所有设备文件就都已经准备完毕.开启此选项相当于设置内核引导参数"devtmpfs.mount=1",关闭此选项相当于设置内核引导参数"devtmpfs.mount=0".开启此项后,你就可以放心的使用"init=/bin/sh"直接进入救援模式,而不必担心"/dev"目录空无一物.注意:此选项并不影响基于 initramfs 的启动,此种情况下,devtmpfs 必须被手动挂载.所以,如果你的系统使用 initrd 或者有专门的启动脚本用于挂载"/dev"目录(大多数发行版都有这样的脚本),或者你看了前面的解释,还是不确定,那就选"N".对于实在想要使用"init=/bin/sh"直接进入救援模式的人来说,还是使用"init=/bin/sh devtmpfs.mount=1"吧!

Select only drivers that don't need compile-time external firmware

CONFIG_STANDALONE

只显示那些编译时不需要额外固件支持的驱动程序,除非你有某些怪异硬件,否则请选"Y".

Prevent firmware from being built

CONFIG_PREVENT_FIRMWARE_BUILD

不编译固件(firmware).固件一般是随硬件的驱动程序提供的,仅在更新固件的时候才需要重新编译.建议选"Y".

Userspace firmware loading support

CONFIG_FW_LOADER

用户空间固件加载支持.如果内核自带的模块需要它,它将会被自动选中.但某些内核树之外的模块也可能需要它,这时候就需要你根据实际情况手动开启了.

Include in-kernel firmware blobs in kernel binary

CONFIG_FIRMWARE_IN_KERNEL

内核源码树中包含了许多驱动程序需要的二进制固件(blob),推荐的方法是通过"make firmware_install"将"firmware"目录中所需的固件复制到系统的"/lib/firmware/"目录中,然后由用户空间帮助程序在需要的时候进行加载.开启此项后,将会把所需的"blob"直接编译进内核,这样就可以无需用户空间程序的帮助,而直接使用这些固件了(例如:当根文件系统依赖于此类

固件,而你又不想使用 initrd 的时候).每个需要此类二进制固件的驱动程序,都会有一个"Include firmware for xxx device"的选项,如果此处选"Y",那么这些选项都将被隐藏.建议选"N".

External firmware blobs to build into the kernel binary

CONFIG_EXTRA_FIRMWARE

指定要额外编译进内核的二进制固件(blob).此选项的值是一个空格分隔的固件文件名字符串,这些文件必须位于 CONFIG_EXTRA_FIRMWARE_DIR 目录中(其默认值是内核源码树下的"firmware"目录).

Firmware blobs root directory

CONFIG_EXTRA_FIRMWARE_DIR

指定 CONFIG_EXTRA_FIRMWARE 中列出的文件位于哪个目录.默认值是当前内核源码树下的"firmware"目录.若有需要,你也可以修改成其他目录(例如"/lib/firmware/").

Fallback user-helper invocation for firmware loading

CONFIG_FW_LOADER_USER_HELPER

在内核自己直接加载固件失败后,作为补救措施,调用用户空间帮助程序(通常是 udev)再次尝试加载.通常这个动作是不必要

的,因此应该选"N",如果你使用了 udev 或 systemd,则必须选"N".仅在某些特殊的固件位于非标准位置时,才需要选"Y".

Allow device coredump

CONFIG_ALLOW_DEV_COREDUMP

为驱动程序开启 coredump 机制,仅供调试.

Driver Core verbose debug messages

CONFIG_DEBUG_DRIVER

让驱动程序核心在系统日志中产生冗长的调试信息,仅供调试

Managed device resources verbose debug messages

CONFIG_DEBUG_DEVRES

为内核添加一个"devres.log"引导参数.当被设为非零值时,将会打印出设备资源管理驱动(devres)的调试信息.仅供调试使用.

Bus devices

总线设备.此类设备仅出现在 ARM 平台.

Connector - unified userspace <-> kernelspace linker

CONFIG_CONNECTOR

统一的用户空间和内核空间连接器,工作在 netlink socket 协议的顶层.连接器是非常便利的用户态与内核态的通信方式,这

些驱动使内核知道当进程 fork 并使用 proc 连接器更改 UID/GID/SID(会话 ID).内核需要知道什么时候进程 fork(CPU 中运行多个任务)并执行,否则,内核可能会低效管理资源.内核有几个连接器应用实例:CONFIG_HYPERV_UTILS,CONFIG_FB_UVESA,CONFIG_W1_CON,CONFIG_DM_LOG_USERSPACE.另外还有一个给 Gentoo 装上启动画面的例子.建议选"Y".

Report process events to userspace

CONFIG_PROC_EVENTS

提供一个向用户空间报告进程事件(fork,exec,id 变化(uid,gid,suid))的连接器.建议选"Y".

Memory Technology Device (MTD) support

CONFIG_MTD

MTD 子系统是一个闪存转换层.其主要目的是提供一个介于闪存硬件驱动程序与高级应用程序之间的抽象层,以简化闪存设备的驱动.注意:MTD 常用于嵌入式系统,而我们常见的 U 盘/MMC 卡/SD 卡/CF 卡等移动存储设备以及固态硬盘(SSD),虽然也叫"flash",但它们并不是使用 MTD 技术的存储器.仅在你需要使用主设备号为 31 的 MTD 块设备(/dev/romX,/dev/rromX,/dev/flashX,/dev/rflashX),或者主设备号为 90 的 MTD 字符设备(/dev/mtdX,/dev/mtdrX)时选"Y",否则选"N".

Device Tree and Open Firmware support

CONFIG_OF

Device Tree 基础架构与 Open Firmware 支持.主要用于嵌入式环境.不确定的选"N".内核中若有其它选项依赖于它,则会自动选中此项.

Parallel port support

CONFIG_PARPORT

25 针并口(LPT 接口)支持.古董级的打印机或扫描仪可能使用这种接口.目前已被淘汰.

Plug and Play support

CONFIG_PNP

即插即用(PnP)支持.选"Y"表示让 Linux 为 PnP 设备分配中断和 I/O 端口(需要在 BIOS 中开启"PnP OS"),选"N"则表示让 BIOS 来分配(需要在 BIOS 中关闭"PnP OS").建议选"Y".

PNP debugging messages

CONFIG_PNP_DEBUG_MESSAGES

允许使用"pnp.debug"内核参数在系统启动过程中输出 PnP 设备的调试信息,建议选"N".

Block devices

CONFIG_BLK_DEV

块设备,建议选"Y".

Null test block driver

CONFIG_BLK_DEV_NULL_BLK

仅供调试使用

Normal floppy disk support

CONFIG_BLK_DEV_FD

通用软驱支持.已被时代抛弃的设备

Parallel port IDE device support

CONFIG_PARIDE

通过并口与计算机连接的 IDE 设备,比如某些老旧的外接光驱或硬盘之类.此类设备早就绝种了

Block Device Driver for Micron PCIe SSDs

CONFIG_BLK_DEV_PCIESSD_MTIP32XX

Micron P320/P325/P420/P425 系列固态硬盘支持

Compressed RAM block device support

CONFIG_ZRAM

zram 是一种基于压缩内存的虚拟块设备,它允许你创建"/dev/zramN"块设备文件,并将它当作普通的磁盘一样使用.它完全位于物理内存中,并被实时压缩与解压以节约物理内存的用量,所有对"/dev/zramN"的读写实质上都是对内存的读写,从而可以获得比一般的磁盘快的多的 IO 速度.常将它用做'/tmp'分区或作为 swap 分区挂载.你可以把它看作是

CONFIG_BLK_DEV_RAM 的升级版.具体用法可以参考内核文档 'Documentation/blockdev/zram.txt'.

Compaq SMART2 support

CONFIG_BLK_CPQ_DA

基于 Compaq SMART2 控制器的磁盘阵列卡

Compaq Smart Array 5xxx support

CONFIG_BLK_CPQ_CISS_DA

基于 Compaq Smart 控制器的磁盘阵列卡

SCSI tape drive support for Smart Array 5xxx

CONFIG_CISS_SCSI_TAPE

在基于 Compaq Smart 控制器的磁盘阵列卡上使用的磁带机

Mylex DAC960/DAC1100 PCI RAID Controller support

CONFIG_BLK_DEV_DAC960

Mylex DAC960, AcceleRAID, eXtremeRAID PCI RAID 控制器.很古董的设备了.

Micro Memory MM5415 Battery Backed RAM support

CONFIG_BLK_DEV_UMEM

一种使用电池做后备电源的内存,但被用作块设备,可以像硬盘一样被分区

Loopback device support

CONFIG_BLK_DEV_LOOP

loop 是指拿文件来模拟块设备(/dev/loopX),比如可以将一个 iso9660 镜像文件当成文件系统来挂载.建议选"Y".

Number of loop devices to pre-create at init time

CONFIG_BLK_DEV_LOOP_MIN_COUNT

系统预先初始化的 loop 设备个数.此值可以通过内核引导参数"loop.max_loop"修改.如果你使用 util-linux-2.21 以上版本,建议设为"0"(loop 设备将通过/dev/loop-control 动态创建),否则保持默认值即可.

Cryptoloop Support

CONFIG_BLK_DEV_CRYPTOLOOP

使用系统提供的 CryptoAPI 对 loop 设备加密.注意:因为不能在 Cryptoloop 上创建日志型文件系统(CONFIG_DM_CRYPT 模块可以),所以 Cryptoloop 已经逐渐淡出了.建议选"N".

DRBD Distributed Replicated Block Device support

CONFIG_BLK_DEV_DRBD

DRBD(Distributed Replicated Block Device)是一种分布式储存系统.DBRD 处于文件系统之下,比文件系统更加靠近操作系统内核及 IO 栈.DRBD 类似 RAID1 磁盘阵列,只不过 RAID1 是在同一台电脑内,而 DRBD 是透过网络.注意:为了进行连接认证,你还需要选中 CONFIG_CRYPTO_HMAC 以及相应的哈希算法.不确定的选"N".

DRBD fault injection

CONFIG_DRBD_FAULT_INJECTION

模拟 IO 错误,以用于测试 DRBD 的行为.主要用于调试目的

Network block device support

CONFIG_BLK_DEV_NBD

让你的电脑成为网络块设备的客户端,也就是可以挂载远程服务器通过 TCP/IP 网络提供的块设备(/dev/ndX).提示:这与 NFS 或 Coda 没有任何关系.更多详情参见 "Documentation/blockdev/nbd.txt".不确定的选"N".

OSD object-as-blkdev support

CONFIG_BLK_DEV_OSD

允许将一个单独的 SCSI OSD(Object-Based Storage Devices) 对象当成普通的块设备来使用.举例来说,你可以在

OSD 设备上创建一个 2G 大小的对象,然后通过本模块将其模拟成一个 2G 大小的块设备使用.不确定的选"N".

STEC S1120 Block Driver

CONFIG_BLK_DEV_SKD

STEC 公司的 S1120 PCIe SSD

Promise SATA SX8 support

CONFIG_BLK_DEV_SX8

基于 Promise 公司的 SATA SX8 控制器的 RAID 卡

RAM block device support

CONFIG_BLK_DEV_RAM

内存中的虚拟磁盘,大小固定.详情参阅"Documentation/blockdev/ramdisk.txt".由于其功能比 CONFIG_TMPFS 和 CONFIG_ZRAM 弱许多,使用上也不方便,所以除非你有明确的理由,否则应该选"N",并转而使用 CONFIG_TMPFS 或 CONFIG_ZRAM.

Default number of RAM disks

CONFIG_BLK_DEV_RAM_COUNT

默认 RAM disk 的数量.请保持默认值,除非你知道自己在做什么.

Default RAM disk size (kbytes)

CONFIG_BLK_DEV_RAM_SIZE

默认 RAM disk 的大小.请保持默认值,除非你知道自己在做什么.

Support XIP filesystems on RAM block device

CONFIG_BLK_DEV_XIP

XIP(eXecute In Place)支持(指应用程序可以直接在 flash 闪存内运行,不必再把代码读到系统 RAM 中).一般用于嵌入式设备.

Packet writing on CD/DVD media

CONFIG_CDROM_PKTCDVD

CD/DVD 刻录机支持.详情参见"Documentation/cdrom/packet-writing.txt"文档

Free buffers for data gathering

CONFIG_CDROM_PKTCDVD_BUFFERS

用于收集写入数据的缓冲区个数(每个占用 64Kb 内存),缓冲区越多性能越好.

Enable write caching

CONFIG_CDROM_PKTCDVD_WCACHE

为 CD-R/W 设备启用写入缓冲,目前这是一个比较危险的选项.建议关闭.

ATA over Ethernet support

CONFIG_ATA_OVER_ETH

以太网 ATA 设备(ATA over Ethernet)支持.

Xen virtual block device support

CONFIG_XEN_BLKDEV_FRONTEND

XEN 虚拟块设备前端驱动.此驱动用于与实际驱动块设备的后端驱动(通常位于 domain0)通信.

Xen block-device backend driver

CONFIG_XEN

_BLKDEV_BA

CKEND

XEN 块设备后端驱动(通常位于 domain0)允许内核将实际的块设备通过高性能的共享内存接口导出给其他客户端的前端驱动(通常位于非 domain0)使用.

Virtio block driver

CONFIG_VIRTIO_BLK

Virtio 虚拟块设备驱动.仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

Very old hard disk (MFM/RLL/IDE) driver

CONFIG_BLK_DEV_HD

又老又旧的 MFM/RLL/ESDI 硬盘驱动.无需犹豫,选"N".

Rados block device (RBD)

CONFIG_BLK_DEV_RBD

rados 块设备(rbd)支持.它可以与分布式文件系统 Ceph 合作,也能独立工作.

IBM Flash Adapter 900GB Full Height

PCIe Device Driver CONFIG_BLK_DEV_RSXX

IBM Flash Adapter 900GB Full Height PCIe SSD 驱动

NVM Express block device

CONFIG_BLK_DEV_NVME

NVM Express 是专门针对 PCI-E 接口高性能固态硬盘的标准规范.有了这一标准,操作系统厂商只需要编写一种驱动,就可以支持不同厂商的不同 PCI-E SSD 设备,以解决过去 PCI-E SSD 产品形态与规格五花八门,缺乏通用性和互用性的问题.如果你有一块较新的 PCIE 固态硬盘,那么很大可能就是 NVMe 接口.

Misc devices

杂项设备

{省略的部分请按照实际的硬件状况进行选择}

Integrated Circuits ICS932S401

CONFIG_ICS932S401

IDT ICS932S401 系列时钟频率控制芯片支持(可能会出现在某些主板上).

Enclosure Services

CONFIG_ENCLOSURE_SERVICES

SES(SCSI Enclosure Services)是 SCSI 协议中用于查询设备状态(温度/风扇/电源/指示灯)的一项服务.这里的设备可以是移动硬盘盒/磁盘阵列柜/硬盘托架等.SES 可以让主机端透过 SCSI 命令去控制外接 SCSI 设备的电源/风扇以及其他与数据传输无关的东西.要使用这项技术,外置设备和主机上的 SCSI/ATA 控制芯片都需要支持 SES 技术才 OK.事实上,目前大多数外置移动硬盘和所有磁盘阵列柜都支持 SES 规范.

VMware Balloon Driver

CONFIG_VMWARE_BALLOON

VMware 物理内存 balloon 驱动(将客户机操作系统不需要的物理内存页交还给宿主机).参见 CONFIG_BALLOON_COMPACTION 选项.

Generic on-chip SRAM driver

CONFIG_SRAM

许多 SoC 系统都有芯片内嵌的 SRAM.开启此项后,就可以声明将此段内存范围交给通用内存分配器(genalloc)管理.不确定的选"N".

EEPROM support

EEPROM 主要用于保存主板或板卡的 BIOS,如果你想通过此 Linux 系统刷写 BIOS 可以考虑开启相应的子项.不确定的全部选"N".

Intel Management Engine Interface

CONFIG_INTEL_MEI

Intel 芯片组管理引擎,是一种面向企业环境的远程管理技术,其中的重头戏是英特尔主动管理技术.如果你的芯片组位于"CONFIG_INTEL_MEI_ME"中,可以考虑选"Y",不过如果你不明白这是什么东西,那就说明你不需要它,就应该选"N".此外,在某些服务器上此驱动(mei)还可能可能导致监视程序计时器错误,还可能导致无法正常关机.

ME Enabled Intel Chipsets

CONFIG_INTEL_MEI_ME

请根据帮助中列出的芯片组对照实际情况选择.

VMware VMCI Driver

CONFIG_VMWARE_VMCI

VMware VMCI(Virtual Machine Communication Interface)是一个在 host 和 guest 之间以及同一 host 上的 guest 和 guest 之间进行高速通信的虚拟设备.VMCI 主要是提供一个接口让 guest 内的程序来调用,通过这个接口能在一个主机上的多个虚拟机之间进行直接的通信,而且无需经过更上层的其他途径,这样将有效地降低网络通信所产生的开支,但是这需要修改虚拟机上的软件,所以 VMCI 只适用于对虚拟机间通信要求非常高的情况.不确定的选"N".

ATA/ATAPI/MFM/RLL support (DEPRECATED)

CONFIG_IDE

已被废弃的 IDE 硬盘和 ATAPI 光驱等接口的驱动(已被 CONFIG_ATA 取代).选"N",除非你确实知道自己在干什么.

SCSI device support

SCSI 子系统

RAID Transport Class

CONFIG_RAID_ATTRS

这只是用来得到 RAID 信息以及将来可能用于配置 RAID 方式的一个类.不管你的系统使用的是哪种 RAID,都可以放心的关闭此项.不确定的选"N".

SCSI device support

CONFIG_SCSI

SCSI 协议支持.有任何 SCSI/SAS/SATA/USB/Fibre Channel/FireWire 设备之一就必须选上.选"Y".

SCSI target support

CONFIG_SCSI_TGT

内核态的通用 SCSI Target 实现(原 LIO 项目).SCSI 子系统使用了一种客户机-服务器(C/S)模型.通常,一台计算机是这个模型中的客户机(称为"initiator"),向目标(target)发起块操作请求,这个"target"通常是一个存储设备(例如一块硬盘).此模块的功能是将一台计算机变成一个"target"(就像一个普通的硬盘一样),响应其他"initiator"节点的操作请求,从而让"target"能够提供更加高级的功能:复制,自动精简配置,重复数据删除,高可用性,自动备份等.不确定的选"N".

SCSI: use blk-mq I/O path by default

CONFIG_SCSI_MQ_DEFAULT

对所有 SCSI 块设备默认使用新式的多重队列 I/O 调度机制(blk-mq),也就是将 I/O 请求分散至多个 CPU 处理以提高性能.相当于开启"scsi_mod.use_blk_mq"内核模块参数.尤其适合于 SSD(高 IOP)/磁盘阵列(多通道)这类存储设备.

legacy /proc/scsi/ support

CONFIG_SCSI_PROC_FS

过时的/proc/scsi/接口.某些老旧的刻录程序可能需要它,建议选"N".

SCSI disk support

CONFIG_BLK_DEV_SD

使用 SCSI/SAS/SATA/PATA/USB/Fibre Channel 存储设备的必选.选"Y".

SCSI tape support

CONFIG_CHR_DEV_ST

通用 SCSI 磁带驱动

SCSI OnStream SC-x0 tape support

CONFIG_CHR_DEV_OSST

专用于 OnStream SC-x0/USB-x0/DI-x0 的 SCSI 磁带/USB 盘驱动

SCSI CDROM support

CONFIG_BLK_DEV_SR

通过 SCSI/FireWire/USB/SATA/IDE 接口连接的 DVD/CD 驱动器(基本上涵盖了所有常见的接口).

Enable vendor-specific extensions (for SCSI CDROM)

CONFIG_BLK_DEV_SR_VENDOR

仅在某些古董级的 SCSI CDROM 设备上才需要:NEC/TOSHIBA cdrom, HP Writers

SCSI generic support

CONFIG_CHR_DEV_SG

通用 SCSI 协议(/dev/sg*)支持.也就是除硬盘/光盘/磁带之外的 SCSI 设备(例如光纤通道).这些设备还需要额外的用户层工具支持才能正常工作.例如:SANE,Cdrtools,CDRDAO,Cdparanoia

SCSI media changer support

CONFIG_CHR_DEV_SCH

SCSI 介质转换设备(SCSI Medium Changer device)是一种控制多个 SCSI 介质的转换器(例如在多个磁带/光盘之间进行切换),常用于控制磁带库或者 CD 自动点歌机(jukeboxes).此种设备会在/proc/scsi/scsi 中以"Type: Medium Changer"列出.控制此类设备的用户层工具包是 scsi-changer.更多细节参见"Documentation/scsi/scsi-changer.txt"文档.不确定的选"N".

SCSI Enclosure Support

CONFIG_SCSI_ENCLOSURE

"Enclosure"是一种用于管理 SCSI 设备的背板装置.移动硬盘盒与磁盘阵列柜就是最常见的"Enclosure"设备.此项主要用于向用户层报告一些"Enclosure"设备的状态,这些状态对于 SCSI 设备的正常运行并非必须.此项依赖于 CONFIG_ENCLOSURE_SERVICES 选项.

Probe all LUNs on each SCSI device

CONFIG_SCSI_MULTI_LUN

默认强制在每个 SCSI 设备上探测所有的逻辑设备数量(Logical Unit Number),其值会被该模块的内核引导参数"max_luns"覆盖.只在一个 SCSI 设备上有多个逻辑设备时才需要选它,一般的 SCSI 设备并不需要.一个 SCSI 设备上有多个逻辑设备的典型例子:多端口 USB 读卡器,CD 点唱机(jukebox),处于"mass storage"模式的智能手机,量产为多个设备后的 U 盘.注意:此项并不影响符合 SCSI-3 或更高标准的设备,因为这些设备会明确的向内核报告逻辑设备数.

Verbose SCSI error reporting (kernel size +=75K)

CONFIG_SCSI_CONSTANTS

以易读的方式报告 SCSI 错误,内核将会增大 75K

SCSI logging facility

CONFIG_SCSI_LOGGING

启用 SCSI 日志(默认并不开启,需要"echo [bitmask] > /proc/sys/dev/scsi/logging_level"),可用于跟踪和捕获 SCSI 设备的错误.关于[bitmask]的说明可以查看"drivers/scsi/scsi_logging.h"文件.不确定的选"N".

Asynchronous SCSI scanning

CONFIG_SCSI_SCAN_ASYNC

异步扫描的意思是,在内核引导过程中,SCSI 子系统可以在不影响其他子系统引导的同时进行 SCSI 设备的探测(包括同时在多个总线上进行检测),这样可以加快系统的引导速度.但是如果 SCSI 设备驱动被编译为模块,那么异步扫描将会导致内核引导出现问题(解决方法是加载 scsi_wait_scan 模块,或者使用 "scsi_mod.scan=sync"内核引导参数).不确定的选"N".

SCSI Transports

SCSI 接口类型,下面的子项可以全不选,内核中若有其他部分依赖它,会自动选上

Parallel SCSI (SPI) Transport Attributes

CONFIG_SCSI_SPI_ATTRS

传统的并行 SCSI(Ultra320/160 之类),已逐渐被淘汰

FiberChannel Transport Attributes

CONFIG_SCSI_FC_ATTRS

光纤通道接口

SCSI target support for FiberChannel Transport Attributes

CONFIG_SCSI_FC_TGT_ATTRS

为光纤通道添加"target"模式驱动

iSCSI Transport Attributes

CONFIG_SCSI_ISCSI_ATTRS

iSCSI 协议是利用 TCP/IP 网络传送 SCSI 命令和数据的 I/O 技术

SAS Transport Attributes

CONFIG_SCSI_SAS_ATTRS

串行 SCSI 传输属性支持(SAS 对于 SPI 的关系犹如 SATA 对于 IDE),这是目前的主流接口

SAS Domain Transport Attributes

CONFIG_SCSI_SAS_LIBSAS

为使用了 SAS Domain 架构的驱动程序提供帮助.SAS Domain 即整个 SAS 交换构架,由"SAS device"和"SAS expander device"组成,其中 Device 又区分为 Initiator 和 Target,它们可以直接对接起来,也可以经过 Expander 进行连接,Expander 起到通道交换或者端口扩展的作用.看不懂就说明你不需要它.

ATA support for libsas (requires libata)

CONFIG_SCSI_SAS_ATA

在 libsas 中添加 ATA 支持,从而让 libata 和 libsas 协同工作.

Support for SMP interpretation for SAS hosts

CONFIG_SCSI_SAS_HOST_SMP

在 libsas 中添加 SMP 解释器,以允许主机支持 SAS SMP 协议.

SRP Transport Attributes

CONFIG_SCSI_SRP_ATTRS

SCSI RDMA 协议(SCSI RDMA Protocol)通过将 SCSI 数据传输阶段映射到 Infiniband 远程直接内存访问(Remote Direct Memory Access)操作加速了 SCSI 协议.

SCSI target support for SRP Transport Attributes

CONFIG_SCSI_SRP_TGT_ATTRS

为 SRP 添加"target"模式驱动

SCSI low-level drivers

CONFIG_SCSI_LOWLEVEL

底层 SCSI 驱动程序

iSCSI Initiator over TCP/IP

CONFIG_ISCSI_TCP

iSCSI 协议利用 TCP/IP 网络在"initiator"与"targets"间传送 SCSI 命令和数据.此选项便是 iSCSI initiator 驱动.相关的用户层工具/文档/配置示例,可以在 open-iscsi 找到.

iSCSI Boot Sysfs Interface

CONFIG_ISCSI_BOOT_SYSFS

通过 sysfs 向用户空间显示 iSCSI 的引导信息.不确定的选 "N".

{此处省略的部分按照实际使用的控制器进行选择}

VMware PVSCSI driver support

CONFIG_VMWARE_PVSCSI

VMware 半虚拟化的 SCSI HBA 控制器

Microsoft Hyper-V virtual storage driver

CONFIG_HYPERV_STORAGE

微软的 Hyper-V 虚拟存储控制器

Intel(R) C600 Series Chipset SAS Controller

CONFIG_SCSI_ISCI

Intel C600 系列芯片组 6Gb/s SAS 控制器

virtio-scsi support

CONFIG_SCSI_VIRTIO

virtio 虚拟 HBA 控制器.仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

PCMCIA SCSI adapter support

CONFIG_SCSI_LOW_LEVEL_PCMCIA

通过 PCMCIA 卡与计算机连接的 SCSI 设备

SCSI Device Handlers

CONFIG_SCSI_DH

针对某些多路径安装的 SCSI 设备的驱动,用在每个节点都需要一个到 SCSI 存储单元的直接路径的集群中,具体子项请按照实际使用的控制器进行选择

OSD-Initiator library

CONFIG_SCSI_OSD_INITIATOR

OSD(Object-Based Storage Device)协议是一个 T10 SCSI 命令集,和 SCSI 处于同一级别,也跟 SCSI 很类似,分成 osd-initiator/osd-target 两部分,用于对象存储文件系统,此选项实现了 OSD-Initiator 库(libosd.ko).更多细节参见 "Documentation/scsi/osd.txt"文件.看不懂就说明你不需

要.[提示]此选项依赖于 CONFIG_CRYPTO_SHA1 和 CONFIG_CRYPTO_HMAC 模块.

OSD Upper Level driver

CONFIG_SCSI_OSD_ULD

提供 OSD 上层驱动(也就是向用户层提供/dev/osdX 设备).从而允许用户层控制 OSD 设备(比如挂载基于 OSD 的 exofs 文件系统).

Serial ATA and Parallel ATA drivers

CONFIG_ATA

SATA 与 PATA(IDE)设备.桌面级 PC 以及低端服务器的硬盘基本都是此种接口

Verbose ATA error reporting

CONFIG_ATA_VERBOSE_ERROR

输出详细的 ATA 命令描述信息.大约会让内核增大 6KB.禁用它将会导致调试 ATA 设备错误变得困难.

ATA ACPI Support

CONFIG_ATA_ACPI

与 ATA 相关的 ACPI 对象支持.这些对象与性能/安全/电源管理等相关.不管你使用的是 IDE 硬盘还是 SATA 硬盘,都建议开启(可以使用内核引导参数"libata.noacpi=1"关闭).

SATA Zero Power Optical Disc Drive (ZPODD) support

CONFIG_SATA_ZPODD

这是 SATA-3.1 版规范新增的节能相关内容,用新的电源管理策略降低了整个系统的电力需求,可以让处于空闲状态的光驱耗电量近乎于零.这需要主板和光驱两者都支持 SATA-3.1 规范才行.

SATA Port Multiplier support

CONFIG_SATA_PMP

SATA 端口复用器(Port Multiplier)是一个定义在 SATA 规范里面的可以选择的功能,可以把一个活动主机连接多路复用至多个设备连接,相当于一个 SATA HUB.不确定的选"N".

AHCI SATA support

CONFIG_SATA_AHCI

AHCI SATA 支持.这是最佳的 SATA 模式(NCQ 功能依赖于它).某些主板还需要在 BIOS 中将硬盘明确设为 AHCI 模式.使用 SATA 硬盘者必选"Y".[提示]由于各厂商芯片组内的 SATA 控制器都遵循同一种规范,所以并不需要各种各样针对不同 SATA 控制器的驱动,就这一个驱动基本就能通吃所有 SATA 控制器了,这比丰富多彩的网卡驱动省事多了.

Platform AHCI SATA support

CONFIG_SATA_AHCI_PLATFORM

这是用于嵌入式系统的与 AHCI 接口兼容的 SATA 驱动.并不是常见的芯片组中的 SATA 控制器驱动.不确定的选"N".

{此处省略几个特殊且不常见的 SATA 控制器驱动}

ATA SFF support (for legacy IDE and PATA)

CONFIG_ATA_SFF

使用 SATA 硬盘的用户可无视此项,选"N"即可.对于依然使用老旧的 IDE/PATA 硬盘的用户而言,按照实际情况在子项中选择相应的控制器驱动即可.

{此处省略几个罕见的 Pacific/Promise 芯片组}

ATA BMDMA support

CONFIG_ATA_BMDMA

这是 IDE 控制器的事实标准.除了上世纪的古董外,绝大多数芯片组都遵守这个标准,选"Y",然后从子项中选择恰当的芯片组/控制器.

{此处省略的 PIO-only SFF 芯片组都是早就绝迹的老古董}

ACPI firmware driver for PATA

CONFIG_PATA_ACPI

通过 ACPI BIOS 去操作 IDE 控制器.仅用于某些比较奇特的 IDE 控制器.选"N".

Generic ATA support

CONFIG_ATA_GENERIC

这是通用的 IDE 控制器驱动.如果你无法确定 IDE 控制器的具体型号(比如需要面对未知的硬件状况),或者不想使用针对特定芯片组的 IDE 驱动,就选"Y"吧.

Multiple devices driver support (RAID and LVM)

CONFIG_MD

多设备支持(RAID 和 LVM).RAID 和 LVM 的功能是使用多个物理设备组建成一个单独的逻辑设备

RAID support

CONFIG_BLK_DEV_MD

"Software RAID"(需要使用 mdadm 工具)支持.也就是"软 RAID".使用硬件 RAID 卡的用户并不需要此项.

Autodetect RAID arrays during kernel boot

CONFIG_MD_AUTODETECT

在内核启动过程中自动检测 RAID 模式.如果你没有使用 RAID,那么选中此项将会让内核在启动过程中增加几秒延迟.如果你使用了"raid=noautodetect"内核引导参数关闭了自动检测,或者此处选了"N",那么你必须使用"md=???"内核引导参数明确告诉内核 RAID 模式及配置.

Linear (append) mode

CONFIG_MD_LINEAR

线性模式(简单的将一个分区追加在另一个分区之后),一般不使用这种模式.

RAID-0 (striping) mode

CONFIG_MD_RAID0

RAID-0(等量分割)模式,可以获取最高性能,但是却损害了可靠性,一般也不使用这种模式.

RAID-1 (mirroring) mode

CONFIG_MD_RAID1

RAID-1(镜像)模式.包含内核的引导分区只能使用这种模式.

RAID-10 (mirrored striping) mode

CONFIG_MD_RAID10

RAID 1+0 模式

RAID-4/RAID-5/RAID-6 mode

CONFIG_MD_RAID456

RAID-4/RAID-5/RAID-6 模式

Multipath I/O support

CONFIG_MD_MULTIPATH

多路径 IO 支持是指在服务器和存储设备之间使用冗余的物理路径组件创建"逻辑路径",如果这些组件发生故障并造成路径失

败,多路径逻辑将为 I/O 使用备用路径以使应用程序仍然可以访问其数据.该选项已废弃,并已被 CONFIG_DM_MULTIPATH 所取代.选"N".

Faulty test module for MD

CONFIG_MD_FAULTY

用于 MD(Multi-device)的缺陷测试模块,仅用于调试.

Block device as cache

CONFIG_BCACHE

将一个块设备用作其他块设备的缓存(Bcache).此缓存使用 btree(平衡树)索引,并专门为 SSD 进行了优化.仅在你打算使用高速 SSD 作为普通硬盘的缓存时才需要此功能.详情参见"Documentation/bcache.txt"文档.

Bcache debugging

CONFIG_BCACHE_DEBUG

仅供内核开发者调试使用

Extended runtime checks

CONFIG_BCACHE_EDEBUG

仅供内核开发者调试使用

Debug closures

CONFIG_BCACHE_CLOSURES_DEBUG

仅供内核开发者调试使用

Device mapper support

CONFIG_BLK_DEV_DM

Device-mapper 是一个底层的卷管理器,提供了一种从逻辑设备到物理设备的映射框架,用户可以很方便的根据自己的需要制定存储资源的管理策略.它不像 RAID 那样工作在设备层,而是通过块和扇区的映射机制,将不同磁盘的不同部分组合成一个大的块设备供用户使用.LVM2 和 EVMS 都依赖于它.此外,那些集成在南桥(例如 ICH8R/ICH9R/ICH10R 系列等)中所谓的"硬 RAID"(准确的称呼应该是"Device Mapper RAID",又称为"Fake RAID"/"BIOS RAID")也依赖于它.还有企业级高可用环境中经常使用的多路径设备也依赖于它.

request-based DM: use blk-mq I/O path by default

CONFIG_DM_MQ_DEFAULT

对所有 Device-mapper 块设备默认使用新式的多重队列 I/O 调度机制(blk-mq),也就是将 I/O 请求映射至多个硬件或软件队列以提高性能.相当于开启"dm_mod.use_blk_mq"内核模块参数.推荐选"Y".

Device mapper debugging support

CONFIG_DM_DEBUG

仅供内核开发者调试使用

Keep stack trace of persistent data block lock holders

CONFIG_DM_DEBUG_BLOCK_STACK_TRACING

仅供内核开发者调试使用

Crypt target support

CONFIG_DM_CRYPT

此模块允许你创建一个经过透明加密的逻辑设备(使用 cryptsetup 工具),要使用加密功能,除此项外,还需要在"Cryptographic API"里选中相应的加密算法,例如CONFIG_CRYPTO_AES.更多文档请参考 LUKS FAQ.

Snapshot target

CONFIG_DM_SNAPSHOT

允许卷管理器为 DM 设备创建可写的快照(定格于特定瞬间的一个设备虚拟映像).LVM2 Snapshot 需要它的支持.更多详情参见"Documentation/device-mapper/snapshot.txt"文档.不确定的选"N".

Thin provisioning target

CONFIG_DM_THIN_PROVISIONING

"Thin provisioning"(某些地方翻译为"精简配置")的意思是允许分配给所有用户的总存储容量超过实际的存储容量(使用 thin-provisioning-tools 工具).例如给 100 个用户分配空间,每个用户最大允许 10G 空间,共计需要 1000G 空间.但实际情

况是 95%的用户都只使用了不到 1G 的空间,那么实际准备 1000G 空间就是浪费.有了"thin provisioning"的帮助,你实际只需要准备 150G 的空间就可以了,之后,可以随着用户需求的增加,添加更多的实际存储容量,从而减少存储投资和避免浪费.更多详情参见"Documentation/device-mapper/thin-provisioning.txt"文档.

Keep stack trace of thin provisioning block lock holders

CONFIG_DM_DEBUG_BLOCK_STACK_TRACING

仅用于调试目的

Cache target

CONFIG_DM_CACHE

dm-cache 通过将频繁使用的热点数据缓存到一个容量较小但性能很高的存储设备上,从而提升块设备的性能.它支持 writeback 和 writethrough 两种模式,并可以使用多种缓存策略(policy)以判断哪些是热点数据以及哪些数据需要从缓存中移除.更多详情参见"Documentation/device-mapper/cache.txt"文档.不确定的选"N".

MQ Cache Policy

CONFIG_DM_CACHE_MQ

MQ 缓存策略.这是目前唯一真正可用的缓存策略.

Cleaner Cache Policy

CONFIG_DM_CACHE_CLEANER

Cleaner 简单的把所有数据都同步写入到原始设备上,相当于关闭缓存.

Era target

CONFIG_DM_ERA

跟踪块设备上的哪些部分被写入,用于在使用 vendor 快照时维护缓存一致性.不确定的选"N".

Mirror target

CONFIG_DM_MIRROR

允许对逻辑卷进行镜像,同时实时数据迁移工具 pvmove 也需要此项的支持.

RAID 1/4/5/6/10 target

CONFIG_DM_RAID

RAID 1/4/5/6/10 支持.即使使用 ICH8R/ICH9R/ICH10R 这样的南桥,也不推荐使用"Device Mapper RAID"(既无性能优势又依赖于特定硬件),应该直接使用更成熟的 CONFIG_BLK_DEV_MD 模块.

Mirror userspace logging

CONFIG_DM_LOG_USERSPACE

device-mapper 用户空间日志功能由内核模块和用户空间程序两部分组成,此选项是内核模块(API 定义于"linux/dm-dirty-log.h"文件).不确定的选"N".

Zero target

CONFIG_DM_ZERO

"Zero target"类似于"/dev/zero",所有的写入都被丢弃,所有的读取都可以得到无限多个零.可用于某些恢复场合.

Multipath target

CONFIG_DM_MULTIPATH

设备映射多路径(DM-Multipath)支持.常用于对可靠性要求比较苛刻的场合.

I/O Path Selector based on the number of in-flight I/Os

CONFIG_DM_MULTIPATH_QL

这是一个动态负载均衡路径选择器:选择当前正在处理中的 I/O 数量最小的通路.

I/O Path Selector based on the service time

CONFIG_DM_MULTIPATH_ST

这是一个动态负载均衡路径选择器:选择完成此 I/O 操作预期时间最少的通路.

I/O delaying target

CONFIG_DM_DELAY

对读/写操作进行延迟,并可将其发送到不同的设备.仅用于测试 DM 子系统.

DM uevents

CONFIG_DM_UEVENT

为 DM 事件透过 netlink 向用户层的 udevd 发出 uevent 通知,这样就允许 udevd 在"/dev/"目录中执行相应的操作.

Flakey target

CONFIG_DM_FLAKEY

模拟间歇性的 I/O 错误,以用于调试 DM 子系统.

Verity target

support

CONFIG_DM_VERITY

Verity target 可以创建一个只读的逻辑设备,然后根据预先生成的哈希校验和(存储在其他设备上),校验底层设备上的数据正确性.要使此模块正常工作,还需要在"Cryptographic API"部分选中相应的哈希算法.

Switch target

support

CONFIG_DM_SWITCH

Switch target 可以创建这样的逻辑设备:将固定尺寸区块的 I/O 操作任意映射到一组固定的路径上.通过向 target 发送一个消息,即可动态的切换指定区块的 I/O 操作所使用的路径.

Log writes target support

CONFIG_DM_LOG_WRITES

此种 target 需要两个设备:主设备按照常规方式使用,辅设备则专门记录所有主设备的写操作.主要用于帮助文件系统的开发者验证文件系统的一致性,仅供开发调试使用.

Generic Target Core Mod (TCM) and ConfigFS Infrastructure

CONFIG_TARGET_CORE

通用 TCM 存储引擎与 ConfigFS 虚拟文件系统(/sys/kernel/config)支持.看不懂就说明你不需要.

Fusion MPT device support

CONFIG_FUSION

Fusion MPT(Message Passing Technology) 是 LSI Logic 公司为了更容易实现 SCSI 和光纤通道而提出的技术,支持 Ultra320 SCSI/光纤通道/SAS. VirtualBox 与 VMware 的虚拟 SCSI 控制器"LSI Logic SAS"就使用该目录下的 CONFIG_FUSION_SAS 驱动.

IEEE 1394 (FireWire) support

火线(IEEE 1394)是苹果公司开发的串行接口,类似于 USB,但 PC 上并不常见,算得上是个没有未来的技术了.

I2O device support

CONFIG_I2O

智能输入输出(Intelligent Input/Output)架构让硬件驱动分成了两部分:OSM(特定于操作系统)+HDM(特定于硬件,与操作系统无关).由于 I2O 设备上集成有专用的 I/O 处理器,从而加快 I/O 速度(因为避免了 CPU 的参与).I2O 属于已被废弃的技术,目前能见到的此类设备都属于老古董了.

Macintosh device drivers

CONFIG_MACINTOSH_DRIVERS

苹果的 Macintosh 电脑上的专有设备驱动

Network device support

CONFIG_NETDEVICES

网络设备.除非你不想连接任何网络,否则必选"Y".

Network core driver support

CONFIG_NET_CORE

如果你不想使用任何高级网络功能(拨号网络/EQL/VLAN/bridging/bonding/team/光纤通道/虚拟网络

等),仅仅是一般性质的联网(普通低端服务器,通过路由器或者局域网上网的常规个人电脑或办公电脑),可以选"N".

Bonding driver support

CONFIG_BONDING

链路聚合技术拥有多个不同的称谓:Linux 称为"Bonding",IEEE 称为"802.3ad",Sun 称为"Trunking",Cisco 称为"Etherchannel".该技术可以将多个以太网通道聚合为一个单独的虚拟适配器,例如将两块网卡聚合成一个逻辑网卡,可以用来实现负载均衡或硬件冗余.此项技术目前已逐渐被 CONFIG_NET_TEAM 取代.

Dummy net driver support

CONFIG_DUMMY

Dummy 网络接口本质上是一个可以配置 IP 地址的 bit-bucket(位桶,所有发送到此设备的流量都将被湮灭),以使应用程序看上去正在和一个常规的网络接口进行通信.使用 SLIP(小猫拨号,目前应该已经绝迹了)或 PPP(常用于 PPPoE ADSL)的用户需要它

EQL (serial line load balancing) support

CONFIG_EQUALIZER

串行线路的负载均衡.如果有两个 MODEM 和两条 SLIP/PPP 线路,该选项可以让你同时使用这两个通道以达到双倍速度(网络

的对端也要支持 EQL 技术).曾经昙花一现的 ISDN 就这项技术的一个实例.

Fibre Channel driver support

CONFIG_NET_FC

光纤通道(Fibre Channel)是一种高速网络串行协议,主要用于存储局域网(SAN),与传统的 iSCSI 技术相比,除了提供更高的数据传输速度(此优势不是绝对的),更远的传输距离,更多的设备连接支持,更稳定的性能,更简易的安装以外,最重要的是支持网络区域存储(SAN)技术.FC 与 SCSI 兼容,并意在取代 iSCSI(看起来难以如愿,并且有可能被 40Gb 以上的 iSCSI 反超).如果你的机器上有光纤通道卡(FC 卡),除了需要开启此项外,还需要开启相应的 FC 卡驱动,以及 CONFIG_CHR_DEV_SG 选项.

Generic Media Independent Interface device support

CONFIG_MII

媒体独立接口(Media Independent Interface)又称介质无关接口,是 IEEE-802.3(规定了以太网相关协议的具体内容)定义的以太网行业标准.它包括一个数据接口,以及一个位于 MAC 和 PHY 之间的控制接口.[提示]大多数以太网卡都有 MII 收发器,其驱动都依赖于此项,也会自动选中此项.

Intermediate Functional Block support

CONFIG_IFB

IFB是一个中间层驱动,可以用来灵活的配置资源共享.更多信息参见 iproute2 文档.看不懂就说明你不需要.

Ethernet team driver support

CONFIG_NET_TEAM

team 驱动.允许通过"ip link add link [address MAC] [NAME] type team"命令,或者使用将多个以太网卡(称为"port")组合在一起,创建一个虚拟的"team"网络设备,从而允许故障转移或者提高吞吐率,其目的是取代传统的"Bonding"(CONFIG_BONDING)驱动."ip"是 iproute2 包中的一个命令.不确定的选"N".

Broadcast mode support

CONFIG_NET_TEAM_MODE_BROADCAST

广播模式:所有网卡共用同一个 MAC 地址,每一个包都从所有网卡同时发送,不做负载均衡,仅做链路冗余,需要和交换机的"聚合强制不协商"方式配合使用.此模式最浪费资源,但可靠性最高,容错能力最强.常用于强调极端可靠的金融业.

Round-robin mode support

CONFIG_NET_TEAM_MODE_ROUNDROBIN

循环均衡模式:所有网卡共用同一个 MAC 地址,数据包依次从每个网卡循环分发(例如,在三个网卡一组的情况下,第 0 个包走

eth0,第 1 个包走 eth1,第 2 个包走 eth2,第 3 个包走 eth0,第 4 个包走 eth1,第 5 个包走 eth2,第 6 个包走 eth0,...一直循环分发下去,直到传输完毕),带宽增加,支持容错(故障链路会被自动踢出),交换机需要配置聚合口(思科叫"port channel").数据包从不同的网卡发出,若中途再经过不同的链路,在到达客户端时可能会乱序,从而造成吞吐量达不到理论上的翻倍效果.

Random mode support

CONFIG_NET_TEAM_MODE_RANDOM

随机均衡模式:所有网卡共用同一个 MAC 地址,数据包依次随机选择一个网卡分发(例如,在三个网卡一组的情况下,第 0 个包走 eth2,第 1 个包走 eth0,第 2 个包走 eth2,第 3 个包走 eth1,第 4 个包走 eth1,第 5 个包走 eth0,第 6 个包走 eth2,...一直随机分发下去,直到传输完毕),带宽增加,支持容错(故障链路会被自动踢出),交换机需要配置聚合口(思科叫"port channel").数据包从不同的网卡发出,若中途再经过不同的链路,在到达客户端时可能会乱序,从而造成吞吐量达不到理论上的翻倍效果.

Active-backup mode support

CONFIG_NET_TEAM_MODE_ACTIVEBACKUP

主备模式:无需更改每个网卡的原生 MAC 地址,但是 team 的 MAC 地址对外仅在主网卡上可见并且保持不变,同一时刻仅有主网卡处于激活状态,其他备用网卡都处于等待状态,所有流量仅通过主网卡发送,仅在主网卡故障时,某个备用网卡才会被激活成主

网卡.此模式仅提供容错能力,可靠性高,但是资源利用率最低.此模式最大的好处是不需要在交换机上做特别的设置.

Load-balance mode support

CONFIG_NET_TEAM_MODE_LOADBALANCE

BPF 均衡模式:均衡算法由用户空间通过 BPF 接口(bpf_hash_func)设置.

MAC-VLAN support

CONFIG_MACVLAN

MAC-VLAN 是通过 MAC 地址来划分 VLAN 的方式,在 Linux 则用来给网卡添加多个 MAC 地址.你可以使用"ip link add link <real dev> [address MAC] [NAME] type macvlan"命令创建一个虚拟的"macvlan"设备(系统会自动打开网卡的混杂模式),然后就可以在同一个物理网卡上虚拟出多个以太网口.Docker 依赖于它.

MAC-VLAN based tap driver

CONFIG_MACVTAP

基于 MAC-VLAN 接口的 tap(虚拟以太网设备)字符设备(macvtap)驱动,旨在简化虚拟化的桥接网络,目的是替代 TUN/TAP(CONFIG_TUN)和 Bridge(CONFIG_BRIDGE)内核模块.可以通过与创建 macvlan 设备相同的"ip"命令创建一个虚拟的"macvtap"设备,并通过 TAP 用户空间接口进行访问.

Virtual eXtensible Local Area Network (VXLAN)

CONFIG_VXLAN

"vxlan"虚拟接口可以在第三层网络上创建第二层网络(跨多个物理 IP 子网的虚拟二层子网),是一种在 UDP 中封装 MAC 的简单机制,主要用于虚拟化环境下的隧道虚拟网络(tunnel virtual network).

Network console logging support

CONFIG_NETCONSOLE

网络控制台(netconsole)的作用是通过网络记录内核日志信息.详情参见"Documentation/networking/netconsole.txt"文档.不确定的选"N".

Dynamic reconfiguration of logging targets

CONFIG_NETCONSOLE_DYNAMIC

允许通过 configfs 导出的用户空间接口,在运行时更改日志目标(网口, IP 地址, 端口号, MAC 地址).

Netpoll traffic trapping

CONFIG_NETPOLL_TRAP

netpoll 的目的是让内核在网络和 I/O 子系统尚不能完整可用时,依然能发送和接收数据包.主要用于网络控制台(netconsole)和远程内核调试(KGDBoE)中.不确定的选"N".

Virtual Ethernet over NTB

CONFIG_NTB_NETDEV

PCI-E 非透明桥(CONFIG_NTB)上的虚拟网卡.不确定的选"N".

RapidIO Ethernet over messaging driver support

CONFIG_RIONET

在标准的 RapidIO 通信方式上发送以太网数据包.不确定的选"N".

Universal TUN/TAP device driver support

CONFIG_TUN

TUN/TAP 可以为用户空间提供包的接收和发送服务,可以用来虚拟一张网卡或点对点通道(例如为 QEMU 提供虚拟网卡支持).当程序打开"/dev/net/tun"设备时,驱动程序就会注册相应的"tunX"或"tapX"网络设备,当程序关闭"/dev/net/tun"设备时,驱动程序又会删除相应的"tunX"或"tapX"网络设备以及所有与之相关联的路由.详情参见"Documentation/networking/tuntap.txt"文档.看不懂就表明你不需要.

Support for cross-endian vnet headers on little-endian kernels

CONFIG_TUN_VNET_CROSS_LE

允许小端序(little-endian)内核中的 TUN/TAP 与 MACVTAP 设备驱动解析来自大端序(big-endian)内核的老旧的 virtio 设备的 vnet 头.不确定的选"N".

Virtual ethernet pair device

CONFIG_VETH

该驱动提供了一个本地以太网隧道(设备会被成对的创建).Docker 依赖于它.

Virtio network driver

CONFIG_VIRTIO_NET

virtio 虚拟网卡驱动.仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

Virtual netlink monitoring device

CONFIG_NLMON

提供一个可以监视 netlink skbs 的网络设备,以允许 tcpdump 之类的工具通过 packet socket 来分析 netlink 消息.仅供调试使用.

ARCnet support

CONFIG_ARCNET

ARCnet 是 1977 年由 Datapoint 公司开发的一种局域网技术，它采用令牌总线方案来管理 LAN 上工作站和其他设备之间的共享线路，主要用于工业控制领域中.

ATM drivers

CONFIG_ATM_DRIVERS

可怜的 ATM(异步传输模式)，曾经在 90 年代风靡一时，现在已经消失的无影无踪了.

{这里省略几个专用于 Android/MeeGo 系统的 PF_CAIF 类型套接字相关的选项}

Distributed Switch Architecture drivers

分布式交换架构驱动，其子项都是 Marvell 系列以太网交换机芯片组的驱动

Ethernet driver support

CONFIG_ETHERNET

最常见的以太网卡驱动

{省略的部分请按照实际的硬件状况进行选择，这里仅以两个常见公司的以太网芯片为例进行说明}

AMD devices

CONFIG_NET_VENDOR_AMD

AMD 出品的以太网控制芯片

AMD PCnet32 PCI support

CONFIG_PCNET32

这是 VMware/VirtualBox 虚拟机中常用的网卡

Broadcom devices

CONFIG_NET_VENDOR_BROADCOM

博通(Broadcom)公司的网卡

Broadcom 440x/47xx ethernet support

CONFIG_B44

Broadcom 44xx/47xx 10/100M PCI

Broadcom NetXtremeII support

CONFIG_BNX2

NetXtreme II 1 Gigabit (BCM5706/5708/5709/5716)

Broadcom CNIC support

CONFIG_CNIC

NetXtremeII 系列网卡的 TCP 减负引擎(TCP Offload Engine)特性支持.不过,TOE 并不适合高连接数/小文件的 Web 服务器类应用,它的主要目的是和 IP 存储协议(iSCSI/NFS)一起使用.[注意]TOE 与"Large Receive Offload"是两个不同的东西,不要混淆.

Broadcom Tigon3 support

CONFIG_TIGON3

这是最流行的驱动,其涵盖的型号特别多,但是"Tigon3"的名称却非常具有迷惑性.简单说来,除了 B44,BNX2,BNX2X 中明确列出的型号外,其他型号用的都是这个驱动.[注意]某些型号还需要额外到固件支持,例如:BCM5703/BCM5704 需要 tigon/tg3_tso.bin,BCM5701A0 需要 tigon/tg3.bin,BCM5705 需要 tigon/tg3_tso5.bin

Broadcom NetXtremeII 10Gb support

CONFIG_BNX2X

NetXtreme II 10 Gigabit

(BCM57710/57711/57711E/57712/57800/57810)

Broadcom 578xx and 57712 SR-IOV support

CONFIG_BNX2X_SRIOV

支持 578xx/57712 的单根 I/O 虚拟化(Single Root IOV)技术

FDDI driver support

CONFIG_FDDI

光纤分布式数据接口(FDDI)

HIPPI driver support

CONFIG_HIPPI

高性能并行接口(HIgh Performance Parallel Interface)是一个在短距离内高速传送大量数据的点对点协议.常用于集群和超级计算机.

General Instruments Surfboard 1000

CONFIG_NET_SB1000

SURFboard 1000 插卡式 Cable Medem(ISA 接口),这玩意早就绝种了

PHY Device support and infrastructure

CONFIG_PHYLIB

数据链路层芯片简称为 MAC 控制器,物理层芯片简称之为 PHY,通常的网卡把 MAC 和 PHY 的功能做到了一颗芯片中,但也有一些仅含 PHY 的"软网卡".此选项就是对这些"软网卡"的支持.请根据实际情况选择其下的子项.

Micrel KS8995MA 5-ports 10/100 managed Ethernet switch

CONFIG_MICREL_KS8995MA

Micrel KS8995MA 5 端口 10/100M 以太网交换芯片

PLIP (parallel port) support

CONFIG_PLIP

PLIP(Parallel Line Internet Protocol)用于将两台电脑通过并口进行联网,组成一个简单的客户机/服务器结构.详情参见"Documentation/networking/PLIP.txt".现在的电脑都使用网卡进行互联,并口早就经被丢进历史的垃圾箱了.

PPP (point-to-point protocol) support

CONFIG_PPP

点对点协议(Point to Point Protocol)是 SLIP 的继任者,使用 PPP 需要用户层程序 pppd 的帮助.PPP 实际上有两个版本:基于普通模拟电话线的"异步 PPP"和基于数字线路(例如 ISDN 线路)的"同步 PPP".使用电脑直接拨号的 PPPoE ADSL 用户需要此项.

PPP BSD-Compress compression

CONFIG_PPP_BSDCOMP

为 PPP 提供 BSD(等价于 LZW 压缩算法,没有 gzip 高效)压缩算法支持,需要通信双方的支持才有效.大多数 ISP 都不支持此算法.

PPP Deflate compression

CONFIG_PPP_DEFLATE

为 PPP 提供 Deflate(等价于 gzip 压缩算法)压缩算法支持,需要通信双方的支持才有效.这是比 BSD 更好的算法(压缩率更高且无专利障碍).

PPP filtering

CONFIG_PPP_FILTER

允许对通过 PPP 接口的包进行过滤.仅在你需要使用 pppd 的 pass-filter/active-filter 选项时才需要开启.不确定的选 "N".

PPP MPPE compression (encryption)

CONFIG_PPP_MPPE

为 PPP 提供 MPPE 加密协议支持,它被用于微软的 P2P 隧道协议中.此特性需要 PPTP Client 工具的支持.

PPP multilink support

CONFIG_PPP_MULTILINK

多重链路协议(RFC1990)允许你将多个线路(物理的或逻辑的)组合为一个 PPP 连接一充分利用带宽,这不但需要 pppd 的支持,还需要 ISP 的支持

PPP over ATM

CONFIG_PPPOATM

在 ATM 上跑的 PPP.果断"N".

PPP over Ethernet

CONFIG_PPPOE

这就是 ADSL 用户最常见的 PPPoE,也就是在以太网上跑的 PPP 协议.这需要 RP-PPPoE 工具的帮助

PPP over IPv4 (PPTP)

CONFIG_PPTP

点对点隧道协议(Point-to-Point Tunneling Protocol)是一种主要用于 VPN 的数据链路层网络协议.此功能需要 ACCEL-PPTP 工具的支持.

PPP over L2TP

CONFIG_PPPOL2TP

第二层隧道协议(L2TP)是一种通过 UDP 隧道传输 PPP 流量的技术,对于 VPN 用户来说,L2TP VPN 是比 PPTP VPN 的更好解决方案.

PPP support for async serial ports

CONFIG_PPP_ASYNC

基于普通模拟电话线或标准异步串口(COM1,COM2)的"异步 PPP"支持. PPPoE ADSL 使用的就是这个.不能与下面的 CONFIG_PPP_SYNC_TTY 同时并存.

PPP support for sync tty ports

CONFIG_PPP_SYNC_TTY

基于同步 tty 设备(比如 SyncLink 适配器)的"同步 PPP"支持.常用于高速租用线路(比如 T1/E1).不确定的选"N".

SLIP (serial line) support

CONFIG_SLIP

一个在串行线上(例如电话线)传输 IP 数据报的 TCP/IP 协议.最原始的通过电话线拨号上网就用这个协议,如今基本绝迹了.不确定的选"N".

CSLIP compressed headers

CONFIG_SLIP_COMPRESSED

CSLIP 协议基于 SLIP,但比 SLIP 快,它将 TCP/IP 头(而非数据)进行压缩传送,需要通信双方的支持才有效

Keepalive and linefill

CONFIG_SLIP_SMART

让 SLIP 驱动支持 RELCOM linefill 和 keepalive 监视,这在信号质量比较差的模拟线路上是个好主意

Six bit SLIP encapsulation

CONFIG_SLIP_MODE_SLIP6

这种线路非常罕见,选"N".

USB Network Adapters

USB 网络适配器

Wireless LAN

CONFIG_WLAN

无线网卡

{省略的部分请按照实际的硬件状况进行选择,这里仅以 Intel 公司的主流无线网卡为例进行说明}

Intel Wireless WiFi Next Gen AGN - Wireless-N/Advanced-N/Ultimate-N (iwlwifi)

CONFIG_IWLWIFI

这是目前主流 Intel 无线网卡的驱动.此驱动依赖于二进制 uCode 微代码,它通常被安装到"/lib/firmware"目录,不过你最好亲自用眼睛检查一下其中是否存在"iwlwifi-*.ucode"这样的文件.[提示]下面两种固件与网卡之间的对应关系可参考 iwlwifi 驱动的 wiki 文档.

Intel Wireless WiFi DVM Firmware support

CONFIG_IWLDVM

DVM 固件支持(适用于较老的无线网卡).选"Y/M".[提示]如果你将此驱动静态编译进内核,那么务必使用 CONFIG_EXTRA_FIRMWARE 功能将固件也一起编译进内核.

Intel Wireless WiFi MVM Firmware support

CONFIG_IWLMVM

MVM 固件支持(适用于较新的无线网卡).选"Y/M".[提示]如果你将此驱动静态编译进内核,那么务必使用 CONFIG_EXTRA_FIRMWARE 功能将固件也一起编译进内核.

Debugging Options

仅供调试使用,其下所有选项都选"N".

iwlwifi experimental P2P support

CONFIG_IWLWIFI_P2P

iwlwifi 驱动实验性的 P2P 支持.不确定的选"N".

WiMAX Wireless Broadband devices

WiMAX 无线设备

Wan interfaces support

CONFIG_WAN

广域网(Wide Area Network)网卡支持.这种网卡很罕见.不确定的选"N".

IEEE 802.15.4 drivers CONFIG_IEEE802154_DRIVERS

IEEE 802.15.4描述了低速率无线个人局域网的物理层和媒体接入控制协议

Xen network device frontend driver CONFIG_XEN_NETDEV_FRONTEND

XEN半虚拟化网络设备前端驱动(通常是被"domain 0"导出的)

Xen backend network device CONFIG_

XEN_NET

DEV_BAC

KEND

XEN 半虚拟化网络设备后端驱动,通常被用在"domain 0"内核上,用于向其他 domain 导出半虚拟化网络设备.

VMwar

e

VMXNE

T3

ether

net

drive

r

CONFI

G_VMX

NET3

VMware vmxnet3 虚拟以太网卡驱动

FUJI

TSU

Exte

nded

Sock

et

Netw

ork

Devi

ce

driv

er

CONF

IG_F

UJIT

SU_E

S

FUJITSU PRIMEQUEST 2000 E2 系列网卡

Mi

cr

os

of

t

Hy

pe

r-

V

virtual network driver CONFIG_HYPERV_NET

Microsoft Hyper-V 虚拟以太网卡驱动

ISDNsupportCONFIG_ISDN

上世纪在 ADSL 流行之前曾经有过短暂流行,但现在已经绝迹了

Open-Channel SSD target support

CONFIG_NVM

Open-channel SSD 是一种遵守 NVMe 规范且不使用 FTL 技术的固态硬盘.目前此种 SSD 由于过于前卫还非常罕见.但是非常有前途.

Open-Channel SSD debugging support

CONFIG_NVM_DEBUG

提供 /sys/module/lnvm/parameters/configure_debug 调试接口以允许创建/删除"target".仅供调试使用.

Generic NVM manager for Open-Channel SSDs

CONFIG_NVM_GENNVM

为 Open-Channel SSD 提供 NVM 介质管理(media manager)支持,也就是将数据存放位置与垃圾回收策略的决定权收归操作系统内核,以剥夺固态硬盘设备自身的介质管理功能,或者说允许使用自身不包含介质管理功能的固态硬盘.

Round-robin Hybrid Open-Channel SSD target

CONFIG_NVM_RRPC

允许将 open-channel SSD 在主机上显示为一个块设备:使用线性映射表实现,使用基于开销(cost-based)的垃圾回收机制,并对 4K 大小的 IO 操作进行了优化.

Input device support

输入设备

Generic input layer (needed for keyboard, mouse, ...)

CONFIG_INPUT

通用输入层.只要你有任何输入设备(键盘,鼠标,手写板,触摸板,游戏杆,方向盘,游戏键盘...),就必须选"Y".

Export input device LEDs in sysfs

CONFIG_INPUT_LEDS

将输入设备上的 LED 指示灯当作标准的 LED 类设备导出到 sysfs 中.不确定的选"Y".

Support for memoryless force-feedback devices

CONFIG_INPUT_FF_MEMLESS

游戏玩家使用的力反馈设备,例如: Logitech WingMan Force 3D 飞行摇杆, ThrustMaster FireStorm Dual Power 2.如果你有此类设备,除了本项之外,还需要开启特定于硬件的驱动.

Polled input device skeleton

CONFIG_INPUT_POLLDEV

使用轮询机制的输入设备支持,此项主要是为源码树之外的驱动准备的,内核自带的驱动若有需要会自动选中.不确定的选"N".

Sparse keymap support library

CONFIG_INPUT_SPARSEKMAP

使用"sparse keymap"的输入设备支持,此项主要是为源码树之外的驱动准备的,内核自带的驱动若有需要会自动选中.不确定的选"N".

Matrix keymap support library

CONFIG_INPUT_MATRIXKMAP

使用"matrix keymap"的输入设备支持,此项主要是为源码树之外的驱动准备的,内核自带的驱动若有需要会自动选中.不确定的选"N".

Mouse interface

CONFIG_INPUT_MOUSEDEV

鼠标接口(/dev/input/mouseX,/dev/input/mice).用鼠标的必选(包括 USB 鼠标).[提示]如果系统上有多个鼠标,那么,mouseX 对应单个特定的鼠标,而 mice 则是所有鼠标的集合(所有鼠标的事件都会被发送到这个设备文件中).

Provide legacy /dev/psaux device

CONFIG_INPUT_MOUSEDEV_PSAUX

仍然支持传统的/dev/psaux 接口,这是为兼容老旧的程序而设置.选"N".

Horizontal screen resolution

CONFIG_INPUT_MOUSEDEV_SCREEN_X

作为鼠标使用的数字化转换器(digitizer)或手写板(graphic tablet)需要知道 X window 的水平分辩率.一般可理解为显示屏的水平分辨率.

Vertical screen resolution

CONFIG_INPUT_MOUSEDEV_SCREEN_Y

作为鼠标使用的数字化转换器(digitizer)或手写板(graphic tablet)需要知道 X window 的垂直分辩率.一般可理解为显示屏的垂直分辨率.

Joystick interface

CONFIG_INPUT_JOYDEV

游戏杆(joystick)和游戏键盘(gamepad)支持(/dev/input/jsX)

Event interface

CONFIG_INPUT_EVDEV

将所有的输入设备事件都通过"/dev/input/eventX"以一种通用的方式进行处理.Xorg 需要使用此接口.不确定的选"Y".

Event debugging

CONFIG_INPUT_EVBUG

将所有输入设备的动作(键盘按下,鼠标移动等)都记录到系统日志当中.主要用于调试,同时也会带来安全漏洞(键盘输入中很可能包含你的密码).选"N".

Keyboards

CONFIG_INPUT_KEYBOARD

键盘驱动

AT keyboard

CONFIG_KEYBOARD_ATKBD

标准 AT 键盘或者 PS/2 键盘.[提示]除了台式机 PS/2 接口上的键盘外,许多笔记本的键盘其实也是 PS/2 键盘.使用 USB 键盘或者 ADB 键盘(旧式苹果键盘)的可以选"N".

{此处被省略的键盘都很罕见,基本上不必考虑}

Mice

CONFIG_INPUT_MOUSE

鼠标驱动

PS/2 mouse

CONFIG_MOUSE_PS2

标准的两键或三键的 PS/2 鼠标,同时兼容 Microsoft/Logitech/Genius 生产的带有滚轮或者额外按键的 PS/2 鼠标.使用 Synaptics/ALPS/Elantech 触摸板的用户还可以看看其专用的 X 驱动,这些驱动可以提供更多的高级功能.

使用 USB 鼠标的可以选"N".其下的子项是针对各厂商特定产品的扩展协议支持.按需选择即可.[提示]除了台式机 PS/2 接口上的鼠标外,许多笔记本的触摸板其实也是 PS/2 鼠标.

{此处被省略的鼠标都很罕见,基本上不必考虑}

Joysticks/Gamepads

游戏杆,6 自由度摇杆,游戏键盘,方向盘,射击武器...等各种游戏装置

Tablets

CONFIG_INPUT_TABLET

平板输入设备

Touchscreens

CONFIG_INPUT_TOUCHSCREEN

触摸屏输入设备

Miscellaneous devices

CONFIG_INPUT_MISC

其他杂项输入设备

PC Speaker support

CONFIG_INPUT_PCSPKR

标准蜂鸣器.建议开启.

{此处被省略的其他设备都很罕见,基本上不必考虑}

Hardware I/O ports

硬件 I/O 端口

Serial I/O support

CONFIG_SERIO

串行 I/O 硬件支持.标准 AT 键盘,PS/2 鼠标,串口鼠标,Sun 键盘,游戏杆,6 自由度摇杆等设备都依赖于它.不确定的选"Y".

i8042 PC Keyboard controller

CONFIG_SERIO_I8042

标准 AT 键盘,PS/2 鼠标,这两种设备需要它的支持.

Serial port line discipline

CONFIG_SERIO_SERPORT

RS232 串口(COM).串口鼠标,游戏杆,6 自由度摇杆等设备都依赖于它.

ct82c710 Aux port controller

CONFIG_SERIO_CT82C710

一种德州仪器 TravelMate 笔记本上使用 QuickPort 接口的鼠标

Parallel port keyboard adapter

CONFIG_SERIO_PARKBD

并口键盘适配器,用于将 AT/XT 键盘或 PS/2 鼠标转接到并口上.非常罕见.

PCI PS/2 keyboard and PS/2 mouse controller

CONFIG_SERIO_PCIPS2

接在移动式扩展坞(Docking station)上的 PS/2 键盘或鼠标

PS/2 driver library

CONFIG_SERIO_LIBPS2

为 PS/2 接口上的设备提供驱动(比如 PS/2 鼠标和标准 AT 键盘)

Raw access to serio ports

CONFIG_SERIO_RAW

以 raw 方式访问 serio 接口(echo -n "serio_raw" > /sys/bus/serio/devices/serioX/drvctl),例如 i8042 键盘控制器的 AUX 端口.看不懂的就别选了.

Altera UP PS/2 controller

CONFIG_SERIO_ALTERA_PS2

Altera University Program PS/2 端口支持.不确定的选 "N".

TQC PS/2 multiplexer

CONFIG_SERIO_PS2MULT

TQC 板上的 PS/2 端口复用器(multiplexer)

ARC PS/2 support

CONFIG_SERIO_ARC_PS2

ARC FPGA 平台上的 PS/2 控制器

Gameport support

CONFIG_GAMEPORT

15 针电脑游戏接口(Gameport).图

Character devices

字符设备

Enable TTY

CONFIG_TTY

字符终端和串口都需要 TTY 的支持.选"Y",除非你知道自己在干什么.[提示]你想在控制台上显示汉字吗?试试 CJKTTY 补丁吧!

Virtual terminal

CONFIG_VT

虚拟终端可以在一个物理终端设备上虚拟出多个"显示器+键盘"的组合(可以使用"Alt+Fn"组合键在多个虚拟终端间切换).除非是嵌入式系统,否则必选"Y".

Enable character translations in console

CONFIG_CONSOLE_TRANSLATIONS

在虚拟控制台(console)上支持字体映射和 Unicode 转换.建议选"Y",否则将无法在控制台上显示 Unicode 字符.[提示]如果已经使用了 CJKTTY 补丁,则必选"Y".

Support for console on virtual terminal

CONFIG_VT_CONSOLE

内核默认将第一个虚拟终端(/dev/tty0)用作系统控制台(可以通过"console=tty3"这样的参数去修改),将诸如模块错误/内核错误/启动信息之类的警告信息发送到这里,而且以单用户模式登录时也需要使用这个控制台.若选"N"则会导致黑屏.除非是嵌入式系统,否则必选"Y".

Support for binding and unbinding console drivers

CONFIG_VT_HW_CONSOLE_BINDING

虚拟终端是通过控制台驱动程序与物理终端交互的,但在某些系统上可以使用多个控制台驱动程序(如 framebuffer 控制台驱动程序),该选项使得你可以选择其中之一.如果你需要使用多个控制台驱动,可以选"Y",不确定的选"N".参见"Documentation/console/console.txt"和"Documentation/fb/fbcon.txt"获取更多细节.

Unix98 PTY support

CONFIG_UNIX98_PTYS

伪终端(PTY)是指一个"软件终端",它是由 slave(等价于一个物理终端)和 master(被一个诸如 xterm 之类的进程用来读写 slave 设备)两部分组成的软设备.图形界面用户与需要支持 ssh/telnet 远程登录者必选.

Support multiple instances of devpts

CONFIG_DEVPTS_MULTIPLE_INSTANCES

允许多个"devpts"文件系统实例(使用"-o newinstance"挂载选项),以允许相互隔离的 PTY 命名空间(比如在虚拟化容器中).Docker 依赖于它.systemd 的 PrivateNetwork/PrivateDevices 特性依赖于它.

Legacy (BSD) PTY support

CONFIG_LEGACY_PTYS

使用过时的 BSD 风格的/dev/ptyxx 作为 master,/dev/ttyxx 作为 slave,这个方案有一些安全问题,选"N".

Non-standard serial port support

CONFIG_SERIAL_NONSTANDARD

非标准串口支持.这样的设备非常罕见,选"N".

HSDPA Broadband Wireless Data Card - Globe Trotter

CONFIG_NOZOMI

一种 PCMCIA 接口的 HSDPA(WCDMA) 3G 无线上网卡

Multi-Tech multiport card support

CONFIG_ISI

Multi-Tech 公司生产的多端口卡(拥有多个串口)实验性支持.

不确定的选"N".

HDLC line discipline support

CONFIG_N_HDLC

Microgate SyncLink.不确定的选"N".

GSM MUX line discipline support

CONFIG_N_GSM

GSM MUX(多路复用器)支持.不确定的选"N".

Trace data router for MIPI

P1149.7 cJTAG standard

CONFIG_TRACE_ROUTER

仅用于调试内含 modem 设备的手机系统.

Trace data sink for MIPI

P1149.7 cJTAG standard

CONFIG_TRACE_SINK

仅用于调试内含 modem 设备的手机系统.

KCopy

内核 Copy

Memory-to-memory copies using kernel assist

CONFIG_KCOPY

高性能的进程间内存复制(可以减少一次向共享内存的复制动作).主要用于高性能并行计算领域,比如基于消息传递接口(Message Passing Interface)协议的开发的并行程序.不确定的选"N".

/dev/mem virtual device support

CONFIG_DEVMEM

"/dev/mem"虚拟设备是整个处理器地址空间的全映射(包括所有物理内存/设备 IO 空间/总线映射空间),可以用来直接存取物理内存,常用于访问物理 IO 设备,例如 dmidecode 工具可以从中提取系统识别信息(序列号,制造商,型号,等等),或者 Xorg 可以用来访问显卡的物理内存或者实现用户空间驱动,同时拥有 root 权限的攻击者也可以使用它完成很多标准 rootkit 的行为.如果你需要使用用户空间的驱动或不确定,那么选"Y".如果你觉得安全特别重要,可以选"N".

/dev/kmem virtual device support

CONFIG_DEVKMEM

"/dev/kmem"虚拟设备是内核看到的虚拟内存的全镜像,可以用来访问内核内存.一般可以用来查看内核变量或者用作 rootkit 之类(!危险!).仅供调试,不确定的选"N".

Serial drivers

串口(COM)驱动.串口在台式机主板上正在逐渐消亡,而在笔记本和服务器上早就已经绝迹了.大多数人应该将所有子项都选"N".

8250/16550 and compatible serial support

CONFIG_SERIAL_8250

这是标准串口(COM)驱动.只要你想使用串口,就必选此项.不过,大多数人应该选"N".

Support 8250_core.* kernel options

CONFIG_SERIAL_8250_DEPRECATED_OPTIONS

选"N".

8250/16550 PNP device support

CONFIG_SERIAL_8250_PNP

即插即用串口支持,不确定的选"Y".

Console on 8250/16550 and compatible serial port

CONFIG_SERIAL_8250_CONSOLE

将串口当做系统控制台(接受所有内核消息,单用户模式登录)使用(需要使用"console=ttyS1"参数).仅在没有显示接口嵌入式设备上有用.不确定的选"N".

DMA support for 16550 compatible UART controllers

CONFIG_SERIAL_8250_DMA

与标准 8250/16650 兼容的通用异步收发传输器(Universal Asynchronous Receiver/Transmitter)的 DMA 支持.在嵌入式设计中,UART 用来主机与辅助设备通信,如汽车音响与外接 AP 之间的通信,与 PC 机通信包括与监控调试器和其它器件(如 EEPROM)通信.不确定的选"Y".

8250/16550 PCI device support

CONFIG_SERIAL_8250_PCI

PCI 串口支持.选"N"表示仅支持传统的标准串口.

8250/16550 PCMCIA device support

CONFIG_SERIAL_8250_CS

16-bit PCMCIA 串口支持.选"N"表示仅支持传统的标准串口.

Maximum number of 8250/16550 serial ports

CONFIG_SERIAL_8250_NR_UARTS

允许的最大串口数量,保持默认值即可.

Number of 8250/16550 serial ports to register at runtime

CONFIG_SERIAL_8250_RUNTIME_UARTS

内核在启动时注册的串口数量(可以通过"8250.nr_uarts"参数修改),保持默认即可.

Extended 8250/16550 serial driver

options

CONFIG_SERIAL_8250_EXTENDED

非标准的串口驱动选项(例如 HUB6,中断共享,多端口,超过 4 个 COM 口).不确定的选"N".

Support more than 4 legacy serial ports

CONFIG_SERIAL_8250_MANY_PORTS

如果你的板子上有超过 4 个 COM 接口就选"Y".

Support for sharing serial interrupts

CONFIG_SERIAL_8250_SHARE_IRQ

有些板子上集成了共享 IRQ 的硬件支持.如果有就选"Y".

Autodetect IRQ on standard ports (unsafe)

CONFIG_SERIAL_8250_DETECT_IRQ

让内核去猜串口的 IRQ 号.不安全,选"N".

Support RSA serial ports

CONFIG_SERIAL_8250_RSA

RSA(Remote Supervisor Adapter)串口,是一种 IBM 特定的硬件.看不懂的选"N".

Support for Synopsys DesignWare

8250 quirks

CONFIG_SERIAL_8250_DW

Synopsys DesignWare APB UART 中非标准特性的支持.

Medfield High Speed UART support

CONFIG_SERIAL_MFD_HSU

Medfield 是 Intel 以 x86 为架构开发的 SoC 手机芯片.

{此处被省略的都是非标准的串口设备,按实际情况选择即可}

TTY driver to output user messages via printk

CONFIG_TTY_PRINTK

通过"/dev/ttyprintk"设备使用 printk 发送用户消息.用于在内核中嵌入用户消息.不确定的选"N".

Parallel printer support

CONFIG_PRINTER

并口打印机

Support for user-space parallel port device drivers

CONFIG_PPDEV

用户空间的原始并口设备(/dev/parportN)支持,这样用户空间的程序就可以用原始模式直接访问并口(相当于并口版本的 CONFIG_CHR_DEV_SG).并口打印机/CD-ROM/硬盘都不依赖于此项,所以大部分人可以关闭该选项.

Xen Hypervisor Console support

CONFIG_HVC_XEN

XEN 虚拟控制台设备驱动

Xen Hypervisor Multiple Consoles support

CONFIG_HVC_XEN_FRONTEND

如果你需要多个虚拟控制台,可以选"Y".

Virtio console

CONFIG_VIRTIO_CONSOLE

Virtio 虚拟控制台设备驱动.此外,该驱动还可以作为普通的串口设备(/dev/vportNpX),用于客户机和宿主机之间的通信.仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

IPMI top-level message handler

CONFIG_IPMI_HANDLER

智能平台管理接口(Intelligent Platform Management Interface)是标准的传感器(温度,电压,风扇,电源,机箱入侵)管理规范.IPMI 的核心是专用的基板管理控制器(BMC)硬件,BMC 并不依赖于服务器的 CPU/BIOS/OS,是一个独立运行的管理子系统,只要有 BMC 与 IPMI 固件便可工作.BMC 通常是一个安装在服务器主板上的独立的板卡(也有少数服务器主板内置).IPMI 良好的独立特性便克服了以往基于操作系统的管理方式所受的限

制,例如操作系统不响应或未加载的情况下,仍然可以进行开关机等操作.更多详情参见"Documentation/IPMI.txt"文档.此项技术主要用于服务器领域,个人 PC 和笔记本上是没有的.

Generate a panic event to all BMCs on a panic

CONFIG_IPMI_PANIC_EVENT

当内核 panic(发生紧急情况)时,IPMI 消息处理器将会向每一个已注册的底板管理控制器(BMC)接口生成一个描述该 panic 的 IPMI 事件,这些事件可以引发日志记录/报警/重启/关机等动作.

Generate OEM events containing the panic string

CONFIG_IPMI_PANIC_STRING

当发生紧急情况(panic)时,IPMI 消息处理器将会产生 OEM 类型(f0)的事件

Device interface for IPMI

CONFIG_IPMI_DEVICE_INTERFACE

为 IPMI 消息处理器提供一个 IOCTL 接口以便用户空间进程也可以使用 IPMI,目前支持 poll() 和 select()

IPMI System Interface handler

CONFIG_IPMI_SI

向系统提供接口(KCS,SMIC),建议选"Y".

Probe for all possible IPMI system interfaces by default

CONFIG_IPMI_SI_PROBE_DEFAULTS

较新的硬件通常使用 ACPI 或 DMI 机制导出 IPMI 接口,但是老旧的硬件并不这么做,所以驱动程序必须直接探测硬件,这会导致启动延迟.选"N"表示禁止直接探测.选"Y"表示强制直接探测(相当于使用"ipmi_si_intf.trydefaults=1"内核引导选项).

IPMI SMBus handler (SSIF)

CONFIG_IPMI_SSIF

使用 I2C 总线上的 SMBus 接口访问 BMC(而不是标准接口).建议选"N".

IPMI Watchdog Timer

CONFIG_IPMI_WATCHDOG

启用 IPMI Watchdog 定时器.如果硬件有这种功能,推荐选"Y".

IPMI Poweroff

CONFIG_IPMI_POWEROFF

允许通过 IPMI 消息处理器关闭机器

Hardware Random Number Generator Core support

CONFIG_HW_RANDOM

硬件随机数发生器设备(/dev/hw_random)支持.此设备并不会直接向内核的随机数发生器填充(这是"rngd"守护进程的职责).详情参见"Documentation/hw_random.txt"文档.

Timer IOMEM HW Random Number Generator support

CONFIG_HW_RANDOM_TIMERIOMEM

Technologic Systems 的 TS-7800 单板计算机,这是一个嵌入式设备.

Intel HW Random Number Generator support

CONFIG_HW_RANDOM_INTEL

Intel 基于 i8xx 芯片组(这是 2005 年以前的老产品了)的硬件随机数发生器

AMD HW Random Number Generator support

CONFIG_HW_RANDOM_AMD

AMD 基于 76x 芯片组的硬件随机数发生器

Atmel Random Number Generator support

CONFIG_HW_RANDOM_ATMEL

Atmel AT91 硬件随机数发生器

VIA HW Random Number Generator support

CONFIG_HW_RANDOM_VIA

VIA 芯片组的硬件随机数发生器

VirtIO Random Number Generator support

CONFIG_HW_RANDOM_VIRTIO

Virtio 虚拟的硬件随机数发生器.仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

EXYNOS HW random number generator support

CONFIG_HW_RANDOM_EXYNOS

基于 EXYNOS 的 SOC 嵌入式系统上的硬件随机数发生器

TPM HW Random Number Generator support

CONFIG_HW_RANDOM_TPM

可信赖平台模块(Trusted Platform Module)提供的硬件随机数发生器

/dev/nvram support

CONFIG_NVRAM

直接存取主板上"CMOS RAM"的接口,太危险!建议选"N".

Siemens R3964 line discipline

CONFIG_R3964

与使用西门子 R3964 协议的设备同步通信,除非你有一些诸如 PLC 之类的特殊设备,否则别选

Applicom intelligent fieldbus card support

CONFIG_APPLICOM

Applicom international 公司生产的用于现场总线(fieldbus)的连接卡.不确定的选"N".

PCMCIA character devices

PCMCIA 接口的字符设备

ACP Modem (Mwave) support

CONFIG_MWAVE

IBM Thinkpad 上的一种软猫,古董产品

RAW driver (/dev/raw/rawN) CONFIG_RAW_DRIVER

裸设备的含义是将一个原始块设备(可以是一整块磁盘,也可以是一个分区)当做一个线性的字节流来访问.它是一种没有经过格式化,不经过操作系统缓存,也不能通过文件系统来访问的特殊字符设备.与 FreeBSD 不同,Linux 反对使用裸设备,且被列入了废除计划(建议的做法是使用"O_DIRECT"标志打开对应的块设备文件,例如"/dev/hda1").不确定的选"N".

HPET - High Precision Event Timer CONFIG_HPET

高精度事件定时器(HPET Timer),又被称为"Multimedia Timer",是一种取代传统"ACPI Timer"(CONFIG_X86_PM_TIMER)的硬件时钟发生器,提供14.31818MHz 固定频率.2007 年以后的芯片组一般都支持(有的主板还需要在 BIOS 里面明确开启 HPET 支持),建议开启.

Allow mmap of HPET

CONFIG_HPET_MMAP

允许对 HPET 寄存器进行映射以提高访问速度.但是某些包含 HPET 硬件寄存器的页中同时还含有其他不该暴露给用户的信息,在此种情况下,需要选"N".

Enable HPET MMAP access by default

CONFIG_HPET_MMAP_DEFAULT

默认开启 HPET 寄存器映射

Hang check timer

CONFIG_HANGCHECK

_TIM

ER

宕机检测定时器周期性地检查系统任务调度程序以确定系统的运行状况,如果超过阈值,计算机将重新启动.不确定的选"N".

TP

M

Ha

rd

wa

re

Su

pp

or

t

CO

NF

IG

_T

CG

_T

PM

基于硬件的可信赖平台模块(Trusted Platform Module),它实际上是一个含有密码运算部件和存储部件的小芯片上的系统,由 CPU,存储器,I/O,密码运算器,随机数产生器和嵌入式操作系统等部件组成.使用此功能需要 TrouSerS 工具的帮助.

TPM Interface Specification 1.2 Interface

CONFIG_TCG_TIS

TCG TIS 1.2 TPM 规范支持

TPM Interface Specification 1.2 Interface (I2C - Infineon)

CONFIG_TCG_TIS_I2C_INFINEON

仅需要对 Infineon 的 TPM 设备选"Y".

National Semiconductor TPM Interface

CONFIG_TCG_NSC

仅需要对 National 的 TPM 设备选"Y".

Atmel TPM Interface

CONFIG_TCG_ATMEL

仅需要对 Atmel 的 TPM 设备选"Y".

Infineon Technologies TPM Interface

CONFIG_TCG_INFINEON

仅需要对 Infineon 的 TPM 设备(SLD 9630 TT 1.1 或 SLB 9635 TT 1.2)选"Y".此驱动支持的硬件列表.

STMicroelectronics ST33 I2C TPM

CONFIG_TCG_ST33_I2C

意法半导体(STMicroelectronics)出品的 I2C 总线的 TPM 安全芯片.

for ATCA SBCCONFIG_TELCL

OCK

没见过这种硬件,选"N".

Xillybus generic FPG

AinterfaceCONFIG_XILLY

B

U

S

Xillybus 是一个通用的 FPGA 接口,仅用于嵌入式设备.

I2C support

CONFIG_I2C

I2C 与 SMBus 支持.I2C(读着"I-squared-C")是用于单片机(又称"微控制器")的低速串行总线协议,它为微控制器(Microcontroller)与各种不同的低速设备通信提供了一种廉价的总线(因为只需要使用两个引脚,称为"2 线"),因此广泛的应用于嵌入式环境.SMBus(System Management Bus)差不多相当于是 I2C 的子集,最初的目的是为了管理智能电池,现在常用于硬件监控(电压/风扇转速/温度/电池等)以及内存模块的配置(使用 I2C EEPROM),因此所有 PC 主板都依赖于 SMBus 协议.系统硬件监控工具 lm_sensors 和 i2c-tools 依赖于此模块,硬件传感器和"Video For Linux"也需要该模块的支持.详情参见"Documentation/i2c/summary"文档及整个"i2c"文件夹.不确定的选"Y".

ACPI I2C Operation region support

CONFIG_ACPI_I2C_OPREGION

3.17 版内核新增功能,允许 BIOS 中的代码通过 I2C 主机控制器驱动访问 I2C slave 设备(例如智能电池).

Enable compatibility bits for old user-space

CONFIG_I2C_COMPAT

为了与 lm-sensors 3.1.2 之前的版本兼容而设置.某些 2011 年之前版本的 i2c 相关程序也需要此兼容性.

I2C device interface

CONFIG_I2C_CHARDEV

I2C 设备通常都是由内核控制的,但此选项可以向用户空间提供 I2C 设备接口,以允许用户空间的程序通过/dev/i2c-*字符设备文件使用 I2C 总线."sensors-detect"工具依赖于此功能.详情参见"Documentation/i2c/dev-interface"文档.建议选"M".

I2C bus multiplexing support

CONFIG_I2C_MUX

多路复用 I2C 总线支持.不确定的选"N".

Multiplexer I2C Chip support

I2C 多路复用芯片,其下的子项按实际情况选择就 OK 了

Autoselect pertinent helper modules

CONFIG_I2C_HELPER_AUTO

有一些 I2C 驱动程序需要"I2C algorithm"的帮助才能工作.而"I2C 算法"本质上是 I2C 接口的纯软件抽象.开启此项后,如有需要,则会自动选上这些算法,而无需你再手动选择.推荐选"Y".仅在你想使用额外的算法时,才选"N".

SMBus-specific protocols

CONFIG_I2C_SMBUS

SMBus 特有的扩展支持.目前唯一实际支持的扩展是 SMBus 报警协议.建议选"Y".

I2C Algorithms

I2C 算法,子项可以全不选,若有其他部分依赖其子项时,会自动选上

I2C Hardware Bus support

I2C 硬件支持

*** PC SMBus host controller drivers ***

这部分按照主板芯片组的实际情况选择就 OK 了.[提示]可用"sensors-detect"工具帮助检测

SMBus Control Method Interface

CONFIG_I2C_SCMI

SMBus 控制方法接口(Control Method Interface)是 SMBus 的 ACPI 接口.用于在 ACPI 环境中使用 SMBus 设备.不确定的选"M"(i2c-scmi).

{其余被省略的都是用于嵌入式系统或者额外的 I2C/SMBus 扩展卡,按实际情况选择即可}

I2C/SMBus Test Stub

CONFIG_I2C_STUB

用于帮助开发 SMBus client 驱动(特别是某些传感器芯片).详情参见"Documentation/i2c/i2c-stub"文档.不确定的选"N".

I2C slave support

CONFIG_I2C_SLAVE

I2C slave 模式支持.不确定的选"N".

I2C Core debugging messages

CONFIG_I2C_DEBUG_CORE

向系统日志中传递大量的 I2C Core 调试信息.仅用于调试 I2C 设备故障

I2C Algorithm debugging messages

CONFIG_I2C_DEBUG_ALGO

向系统日志中传递大量的 I2C Algorithm 调试信息.仅用于调试 I2C 设备故障

I2C Bus debugging messages

CONFIG_I2C_DEBUG_BUS

向系统日志中传递大量的 I2C Bus 调试信息.仅用于调试 I2C 设备故障

SPI support

CONFIG_SPI

串行外设接口(Serial Peripheral Interface)是一种标准的四线同步双向串行总线.SPI 类似于 I2C,但比 I2C 的"2 线"稍微复杂一些,SPI 需要 4 个引脚("4 线"),不但传输速率比 I2C 更高,还能实现全双工通信.大多数 SPI 设备不支持动态设备检测,有些甚至是只读或者只写的.SPI 常用于 SoC 与外围设备(RTC,LCD,传感器[温度,压力,触摸屏,游戏控制器],EEPROM,FLASH,数字电位器,相机镜头,音频编解码器,模数转换器,数字信号处理器,通信设备[以太网,USB,USART,CAN,IEEE 802.15.4,IEEE 802.11])之间的通信,MMC 和 SD 卡也可以通过 SPI 协议访问,而 MMC 接口的 DataFlash 卡则必须通过 SPI 才能访问.2016 年英特尔推出了旨在取代传统 LPC/SPI/SMbus 总线的 eSPI(Enhanced Serial Peripheral Interface)总线,eSPI 总线可与 SPI 设备共用以节省引脚或与 SPI 总线分离以提高性能,使用 eSPI 总线可以降低主板布线复杂度,提高与外设的通信性能,降低功耗.不确定的选"Y".

SPMI support

CONFIG_SPMI

系统电源管理接口(SPMI, System Power Management Interface)是一种连接 PMIC(Power Management Integrated Circuits)的双线串行接口.仅用于嵌入式环境.

Qualcomm MSM SSBI bus support

CONFIG_SSBI

高通(Qualcomm)骁龙系列智能手机处理器内嵌的单线串行总线接口(Single-wire Serial Bus Interface)

HSI support

CONFIG_HSI

高速同步串行接口(High speed synchronous Serial Interface)是移动产业处理器接口(MIPI)联盟的高速同步接口工作组发布的一项技术规范.MIPI(Mobile Industry Processor Interface)是 2003 年由 ARM,Nokia,ST,TI 等公司成立的一个联盟,目的是把手机内部的接口(如摄像头,显示屏接口,射频/基带接口等)标准化,从而减少手机设计的复杂程度和增加设计灵活性.MIPI 联盟下面有不同的工作组,分别定义了一系列的手机内部接口标准,比如摄像头接口 CSI,显示接口 DSI,射频接口 DigRF,麦克风/扬声器接口 SLIMbus 等.统一接口标准的好处是手机厂商根据需要可以从市面上灵活选择不同的芯片和模组,更改设计和功能时更加快捷方便.目前,MIPI 联盟的董事成员包括英特尔,摩托罗拉,诺基亚,三星,意法半导体,德州仪器.

PPS support

CONFIG_PPS

秒脉冲(Pulse Per Second)驱动用来控制电流脉冲速率,可用于计时.PPS 的精度可以到纳秒级,而且没有累积误差.这通常是 GPS 天线的一项功能,用于获取 GPS 卫星的授时.

PTP clock support

CONFIG_PTP_1588_CLOCK

精密时间协议(Precision Time Protocol)是 IEEE 1588 定义的一种基于以太网的高精度时间同步协议.PTP 采用硬件与软件结合设计,可以提供比纯软件方式的 NTP(网络时间协议)高的多的精度(微秒级).与 GPS 授时相比,在提供和 GPS 相同的精度情况下,PTP 不需要为每个设备安装 GPS 那样昂贵的组件,只需要一个高精度的本地时钟和提供高精度时钟戳的部件,成本较低.一般的 PC 和服务器上没有 PTP 硬件.

Pin controllers

Pin 控制器.其下的各选项请根据实际硬件状况选择(皆为低功耗或嵌入式平台).

GPIO Support

CONFIG_GPIOLIB

每个芯片都会有至少一个引脚(PIN),像 CPU 或者芯片组这种复杂的芯片,其引脚会有成白上千个,这些 PIN 就是芯片与外部沟通的渠道,每个 PIN 都会有它特定的功能.GPIO(General Purpose I/O)就是芯片上的一种通用功能的引脚,其功能可由使用者通过编程的方式自定义(所谓"可编程引脚"),比如使用两

条 PIN 就可以组成 I2C,使用 4 条 PIN 就可以组成 SPI.嵌入式系统经常需要控制结构简单但数量众多的外部设备(比如 LED 的亮与灭),使用传统的串口或者并口就太"大炮打蚊子",而 GPIO 则非常适合用于控制此类数量众多的简单设备.GPIO 在嵌入式设备中使用广泛,但 PC 平台的芯片组南桥大多也集成有 GPIO 引脚(但只有 BIOS 才知道如何使用他们),以支持某些特殊的定制硬件.详情参见"Documentation/gpio/gpio.txt"文档.不确定的选"N".

Debug GPIO calls

CONFIG_DEBUG_GPIO

仅供调试使用

/sys/class/gpio/... (sysfs interface)

CONFIG_GPIO_SYSFS

为 GPIO 设备添加 sysfs 接口.主要用于调试和问题排查.不确定的选"N".

Generic memory-mapped GPIO controller support (MMIO platform device)

CONFIG_GPIO_GENERIC_PLATFORM

这是最简单的 GPIO 控制器驱动(platform 总线驱动),仅支持单独一个"data"寄存器,用于读/写 GPIO 的状态.不确定的选"Y".

{这里被省略的部分,按主板上实际集成的芯片选择即可}

Dallas's 1-wire support

CONFIG_W1

Dallas 公司发明的单总线是比 I2C 更简单的总线,仅使用一个引脚(1-wire),使用 Master-Slave 结构,用于连接慢速的单引脚设备,比如 iButton 和热传感器.主要用于嵌入式系统.

Power supply class support

CONFIG_POWER_SUPPLY

允许用户空间程序通过 sysfs/uevent 接口对电源(电池,交流电,USB)进行监控.主要用于笔记本与嵌入式设备.

Power supply debug

CONFIG_POWER_SUPPLY_DEBUG

仅供调试使用

Generic PDA/phone power driver

CONFIG_PDA_POWER

通用的 PDA/phone 电源切换驱动.用于在内部电池和外部电源(AC/USB)之间进行切换.

Generic battery support using IIO

CONFIG_GENERIC_ADC_BATTERY

为使用 IIO 总线(CONFIG_IIO)的电池提供的通用驱动

Test power driver

CONFIG_TEST_POWER

仅供测试使用

SBS Compliant gas gauge

CONFIG_BATTERY_SBS

与智能电池系统(Smart Battery System)规范兼容的气压计(集成在电池组中)支持.

GPIO charger

CONFIG_CHARGER_GPIO

支持充电器通过 GPIO 引脚报告其在线状态.

Board level reset or power off

CONFIG_POWER_RESET

允许通过操作板载的主电源,关闭或重启整个系统.仅用于嵌入式系统.

{这里被省略的部分,按实际电池控制芯片选择即可}

Adaptive Voltage Scaling class support

CONFIG_POWER_AVS

自适应电压调节(Adaptive Voltage Scaling)技术能够动态的对设备工作电压进行精细的调整,拥有比 DVFS 更佳的电力利用效率,是一种降低功耗与优化性能并举的电源与性能管理技

术.AVS 在 OMAP 设备上也被称为"SmartReflex".目前仅用于嵌入式领域.

Hardware Monitoring support

CONFIG_HWMON

当前主板大多都有一个监控硬件温度/电压/风扇转速等状况的设备,请按照主板实际使用的芯片选择相应的子项.如果你不知道究竟需要使用哪个驱动,可以使用 Superiotool 和 sensors-detect 工具进行检测.另外,某些子项可能还需要 CONFIG_I2C 的支持.更多详情参见"Documentation/hwmon/userspace-tools"文档.

Hardware Monitoring Chip debugging messages

CONFIG_HWMON_DEBUG_CHIP

在系统日志中输出大量的 I2C 调试信息,仅用于故障调试

{被省略的部分,按实际的硬件监控芯片选择即可}

GPIO fan

CONFIG_SENSORS_GPIO_FAN

连接在 GPIO 引脚上的风扇

PMBus support

CONFIG_PMBUS

电源管理总线(Power Management Bus)是一种基于 SMBus(CONFIG_I2C)的开放标准的数字电源管理协议,可以用于

配置/监控/操作电源变换器,目前全球有超过 40 个 IC 厂商提供满足 PMBus 标准的产品.最新的 PMBus+ 1.3 标准增加 AVS(CONFIG_POWER_AVS)支持,可以动态控制设备的工作电压.根据你的实际硬件状况选择子项.

ACPI 4.0 power meter

CONFIG_SENSORS_ACPI_POWER

将 ACPI 4.0(2009 年 6 月发布)中定义的瓦特表(用于测量功耗)当做硬件监控设备导出到用户空间.需要固件支持 ACPI 4.0 规范,并且有一个瓦特表.不确定的选"N".

ASUS ATK0110

CONFIG_SENSORS_ATK0110

许多华硕主板都有这种 ACPI 硬件监控接口.此驱动可以通过主板固件读取风扇/电压/温度信息.

Generic Thermal sysfs driver

CONFIG_THERMAL

为 ACPI 规范中定义的"thermal"(发热控制)提供一个通用的 sysfs 接口,以方便与诸如温度传感器和风扇之类的设备通信.由于目前所有 PC 和服务器都已支持 ACPI,并且发热控制也越来越重要,所以建议选"Y".详情参见"Documentation/thermal/sysfs-api.txt"文档.

Expose thermal sensors as hwmon device

CONFIG_THERMAL_HWMON

将温度传感器同时注册为一个硬件监控设备,从而让温度传感器同样在 sysfs 中拥有 hwmon 接口.

Enable writable trip points

CONFIG_THERMAL_WRITABLE_TRIPS

允许用户空间程序更改温度报警阀值(trip temperature).

Default Thermal governor

选择默认的热调节器,建议选"step_wise".

Fair-share thermal governor

CONFIG_THERMAL_GOV_FAIR_SHARE

此调节器根据设备对所属区域的"贡献"(contribution)进行调节.

Step_wise thermal governor

CONFIG_THERMAL_GOV_STEP_WISE

此调节器以线性方式进行调节,也就是每次调节都只在紧邻的两档之间进行切换.

Bang Bang thermal governor

CONFIG_THERMAL_GOV_BANG_BANG

此调节器仅能让散热风扇处于开/关两种状态(根据温度阀值)而不能调节风扇的速度.某些 Acer 笔记本风扇驱动(acerhdf)依赖于此调节器.不能将此调节器设为默认调节器.

User_space thermal governor

CONFIG_THERMAL_GOV_USER_SPACE

此调节器让用户空间程序去决定如何调节

Power allocator thermal governor

CONFIG_THERMAL_GOV_POWER_ALLOCATOR

此调节器可对特定的设备动态分配和限制能量的使用.不确定的选"N".

generic cpu cooling support

CONFIG_CPU_THERMAL

通用的 CPU 降温机制(通过降低频率来实现,而不是通过 ACPI 接口).显然通过 ACPI 接口是更好的机制,所以建议选"N".

Thermal emulation mode support

CONFIG_THERMAL_EMULATION

"Thermal"模拟.仅供调试使用,切勿用于生产系统!!

Intel PowerClamp idle injection driver

CONFIG_INTEL_POWERCLAMP

Intel PowerClamp 驱动通过利用 Nehalem 之后的 CPU 支持的"package-level C-state"特性,强制为在线的 CPU 注入"idle"指令(通过"/sys/class/thermal/"接口设定"idle"百分比),以确保 CPU 的功耗不会超过特定的阈值(发热量也就不会超过特定的阈值).这样刻意的降低系统性能峰值还有一个好处,那就是相对于传统的动态频率调节技术而言,能够达到更高的每瓦特性能.详见 Documentation/thermal/intel_powerclamp.txt 文档.如果你对节能和限制发热量特别在意,同时又不在乎系统峰值性能的降低,可以选"Y".

X86 package temperature thermal driver

CONFIG_X86_PKG_TEMP_THERMAL

所谓"CPU 温度"实际上是个多重概念:(1)Socket 温度,是指 CPU 插座里的测温二极管探测到的温度,相当于 CPU 外表面的温度;(2)Package 温度,是封装在 CPU 内部的测温二极管探测到的温度,是真正的 CPU 内部的温度,此温度永远比 Socket 温度高;(3)Core 温度,是每个 CPU 核心内嵌的温度传感器检测到的温度,有多少个核心就有多少个 Core 温度;此选项提供了检测 Package 温度的驱动.并将此温度用于监控 CPU 温度(对于拥有多颗物理 CPU 的服务器来说每颗物理 CPU 对应一个 Package 温

度).同时,选中此项后,温度报警阀值也变为可以设置两个严重级别不同的值.建议选"Y".

Intel SoCs DTS thermal driver

CONFIG_INTEL_SOC_DTS_THERMAL

专用于 Intel SoC(BayTrail 等)平台的 CPU 内嵌温度传感器驱动.

ACPI INT340X thermal drivers

CONFIG_INT340X_THERMAL

除了 CPU/SOC 内置的温度传感器之外,新式笔记本或平板电脑还经常包含探测整机不同位置温度的传感器,这些温度传感器以 INT3400 ACPI 设备作为主设备(master),并以 INT3401~INT340B ACPI 设备为从设备(slave),此选项提供了对此类温度传感器的驱动支持.

Intel PCH Thermal Reporting Driver

CONFIG_INTEL_PCH_THERMAL

专用于 Intel PCH 芯片组内置温度传感器的驱动.

Watchdog Timer Support

CONFIG_WATCHDOG

选"Y"并选中下面相应的驱动之后,再创建一个主/次设备号为 10/130 的字符设备"/dev/watchdog",即可拥有一只看门狗.其

工作原理是:当/dev/watchdog 设备被打开后,如果喂狗守护进程超过 60 秒没有喂狗(写入"/dev/watchdog"),那么底层的看门狗硬件将会触发整个机器硬重启(相当于按下面板上的"RESET"按钮).这对于提高服务器的在线率来说意义重大.详情参见"Documentation/watchdog/watchdog-api.txt"文档.

WatchDog Timer Driver Core

CONFIG_WATCHDOG_CORE

看门狗核心驱动,它为所有特定于具体硬件的看门狗驱动提供了统一的框架和"/dev/watchdog"接口(未来还会包括 sysfs 接口).使用看门狗的必选.

Disable watchdog shutdown on close

CONFIG_WATCHDOG_NOWAYOUT

默认情况下(此项="N")如果喂狗进程关闭"/dev/watchdog"文件,那么表示停止看门狗功能.开启此项后,看门狗一旦启用就不能被停止(即使关闭"/dev/watchdog"文件也不会停止).

Software watchdog

CONFIG_SOFT_WATCHDOG

内核提供的"软看门狗".使用它不需要有任何硬件的支持,但可靠性不如硬件看门狗,仅能应对喂狗进程的崩溃,不能应对内核本身的崩溃.在某些情况下(例如 Oracle 数据库),CONFIG_HANGCHECK_TIMER 是比"软看门狗"更好的选择.

{此处省略的看门狗硬件请按照实际使用的芯片进行选择}

Sonics Silicon Backplane support

CONFIG_SSB

SSB(Sonics Silicon Backplane)是一种仅在嵌入式环境中使用的总线.

Broadcom specific AMBA

CONFIG_BCMA

Broadcom 特有的 AMBA(Advanced Microcontroller Bus Architecture)总线支持.仅用于嵌入式环境

Multifunction device drivers

MFD(多功能设备)的含义是"在单个芯片上集成多个功能(GPIO,触摸屏,键盘,电流调节,电源管理...)".此种芯片通常通过一个或多个 IRQ 线和低速数据总线(SPI/I2C/GPIO)与主 CPU 进行通信.对于主系统来说,它们通过数据总线显示为一个单独的 MFD 设备.但透过 MFD 框架,又可以拥有多个相互独立的子设备(子功能).

Intel ICH LPC

CONFIG_LPC_ICH

LPC(Low Pin Count)总线是 Intel 于 1998 年发布的一个旨在取代传统 ISA 总线的接口规范,用于连接南桥和 Super I/O 芯

片(用于连接低速外设:串口,并口,PS/2 键鼠,软盘控制器,TPM(可信平台模块),温度传感器,风扇速度监测器)以及 Flash 芯片(BIOS).以往南桥必须保留 ISA 总线,以连接老旧的 ISA 插槽和 Super I/O 芯片(可以使用 Superiotool 和 sensors-detect 工具检测)以及 Flash 芯片.但是 ISA 需要占用大量针脚,主板的线路设计也比较复杂.随着 ISA 插槽的消失,LPC 就顺理成章的出现了,它与 ISA 在软件层面是类似的,同时 LPC 工作速率由 PCI 总线速率同步驱动,但是引脚数大大降低,以方便在拥挤的现代主板上布局,这也是取名"Low Pin Count"的原因.此选项支持几乎所有 Intel 芯片组的 LPC 总线,以方便其他驱动控制 MFD(目前仅有 GPIO 和 watchdog).具体支持的芯片可以查看"drivers/mfd/lpc_ich.c"文件.不确定的选"Y".

Intel SCH LPC

CONFIG_LPC_SCH

用于 Intel Atom 处理器的 Intel SCH(System Controller Hub) LPC 总线支持.目前仅支持 SMBus 和 GPIO.

{此处省略的硬件请按照实际使用的芯片进行选择}

Voltage and Current Regulator Support

CONFIG_REGULATOR

通用的电压与电流调节器框架.除了提供通用的电压与电流调节接口外,还能通过 sysfs 向用户空间提供电压与电流的状态信

息.目的在于通过动态调节电压和电流,降低能耗,延长电池寿命.主要用于嵌入式环境.

Multimedia support

CONFIG_MEDIA_SUPPORT

多媒体设备:摄像头,视频采集,模拟电视,数字电视,机顶盒,收音机,遥控器,数字视频广播(DVB)...内核多媒体子系统由LinuxTV项目负责维护.

Cameras/video grabbers support

CONFIG_MEDIA_CAMERA_SUPPORT

摄像头,视频采集卡

Analog TV support

CONFIG_MEDIA_ANALOG_TV_SUPPORT

模拟电视信号接收器,包括那些既能接收模拟信号又能接收数字信号的电视卡

Digital TV support

CONFIG_MEDIA_DIGITAL_TV_SUPPORT

数字电视信号接收器,包括那些既能接收模拟信号又能接收数字信号的电视卡

AM/FM radio receivers/transmitters support

CONFIG_MEDIA_RADIO_SUPPORT

AM/FM 无线电接收机和发射机,包括那些带有收音机功能的电视卡

Remote Controller support

CONFIG_MEDIA_RC_SUPPORT

基于红外线/射频的遥控器,用于控制视频采集卡或者电视卡.大多数电视卡和视频采集卡都需要它的支持,即使这些卡实际并不需要遥控器.

Media Controller API

CONFIG_MEDIA_CONTROLLER

此 API 用于查询多媒体设备内部的拓扑结构,并进行动态配置.主要用于嵌入式环境中的摄像头配置.

V4L2 sub-device userspace API

CONFIG_VIDEO_V4L2_SUBDEV_API

此 API 用于配置视频的格式/尺寸/帧率.主要用于嵌入式环境中的摄像头配置.

Enable advanced debug functionality on V4L2 drivers

CONFIG_VIDEO_ADV_DEBUG

开启 V4L2 驱动程序的高级调试特性,不确定的选"N".

Enable old-style fixed minor ranges on drivers/video devices

CONFIG_VIDEO_FIXED_MINOR_RANGES

仅在你使用 mknod 而不是 udev 进行设备管理时才需要开启.不确定的选"N".

V4L2 int device (DEPRECATED)

CONFIG_VIDEO_V4L2_INT_DEVICE

仅用于旧式的图像传感器驱动(omap24xxcam 和 tcm825x),反对使用此项.选"N".

DVB Network Support

CONFIG_DVB_NET

DVB(数字视频广播)是一系列国际公认的数字电视标准.此项提供了 DVB 网络(DVB 标准的一部分)支持,可用于数字机顶盒(Set-Top-Box)的自动固件升级以及通过 DVB 卡访问互联网.

maximum number of DVB/ATSC adapters

CONFIG_DVB_MAX_ADAPTERS

最大允许的 DVB/ATSC 电视卡数量.取值范围是[1,255],但经过测试的范围是[4,32].不确定的请保持默认值"8".

Dynamic DVB minor allocation

CONFIG_DVB_DYNAMIC_MINORS

为 DVB 设备节点动态分配次设备号,这样每张 DVB 卡就可以拥有最多 4 个同类型的设备(例如 demux(分离器)和 frontend(前端)).此特性需要 udev 的支持.

Compile Remote Controller keymap modules

CONFIG_RC_MAP

将各种遥控器的 keymap 表编译进内核.这些表都很小,但是如果你不打算使用遥控器,或者更喜欢使用 v4l-utils 包内的 ir-keytable 工具从用户空间加载这些表,可以选"N".

Remote controller decoders

CONFIG_RC_DECODERS

遥控器解码器.其下的子项是各种不同的遥控通信协议.

Remote Controller devices

CONFIG_RC_DEVICES

各种遥控器产品.其下子项按实际的厂商和型号选择即可.

Media USB Adapters

CONFIG_MEDIA_USB_SUPPORT

各种 USB 总线的多媒体设备

USB Video Class (UVC)

CONFIG_USB_VIDEO_CLASS

UVC(USB Video Class)是一个开放的通用 USB 视频捕获标准.目前大多数摄像头都是 UVC 摄像头,也就是俗称的"免驱摄像头".所有符合 UVC 规格的硬件都可以使用通用 UVC 驱动程序,而无需再使用专用驱动.

UVC input events device support

CONFIG_USB_VIDEO_CLASS_INPUT_EVDEV

某些 UVC 摄像头上会带有按钮(常用于开关 LED 灯/拍照),此选项可以将此按钮注册为一个输入设备,以用于报告按钮事件.

GSPCA based webcams

CONFIG_USB_GSPCA

基于 GSPCA 框架的摄像头驱动(依赖于 CONFIG_VIDEO_V4L2),这是一位 60 岁的法国医生的杰作.该驱动适用于大多数常见的非 UVC 摄像头.具体支持的芯片列表可以查看"Documentation/video4linux/gspca.txt"文档.

{此处省略的非 GSPCA 摄像头,电视卡,电视棒等其他硬件请按照实际使用的芯片进行选择}

Media PCI Adapters

CONFIG_MEDIA_PCI_SUPPORT

各种 PCI/PCIe 总线的多媒体设备

V4L platform devices

CONFIG_V4L_PLATFORM_DRIVERS

特定于平台的 V4L(Video For Linux)设备,这些设备不是通过 USB/PCI 这样的总线连接的.一般用于单片机之类的嵌入式环境.

SoC camera support

CONFIG_SOC_CAMERA

所谓"SoC Camera"是指那些不通过 PCI 或 USB 总线连接的摄像头(例如通过 I2C 直接与 SoC 数据总线连接).此选项为这类摄像头提供了通用的支持.

platform camera support

CONFIG_SOC_CAMERA_PLATFORM

仅用于调试目的

{此处省略的部分请按照实际使用的芯片进行选择}

Memory-to-memory multimedia devices

CONFIG_V4L_MEM2MEM_DRIVERS

使用系统内存作为源和目标缓存(Memory-to-memory)的多媒体设备.一般的采集输出驱动仅将系统内存用于源或目标缓存之一.不确定的选"N".

Media test drivers

CONFIG_V4L_TEST_DRIVERS

仅用于调试目的.

Siano SMS1xxx based MDTV via SDIO interface

CONFIG_SMS_SDIO_DRV

使用 SDIO 接口的一种移动数字电视(MDTV)卡,基于 Siano SMS1xxx 芯片.主要用于嵌入式设备

ISA and parallel port

devices

CONFIG_MEDIA_PARPORT_SUPPORT

使用 ISA 或并口的多媒体设备,古董级别的设备

Radio Adapters

CONFIG_RADIO_ADAPTERS

AM/FM 无线电广播接收设备

FireDTV and FloppyDTV

CONFIG_DVB_FIREDTV

Digital Everywhere 生产的 FireWire(IEEE 1394)接口的 DVB 电视接收卡

Cypress

firmware helper routines

CONFIG_CYPRESS_FIRMWARE

Cypress(赛普拉斯)多媒体产品的固件加载帮助程序.

Enable Remote Controller support for

Siano devices CONFIG_SMS_SIANO_RC

Siano 多媒体设备遥控器

Enable debugfs for rsmsdvb

CONFIG_SMS_SIANO_DEBUGFS

仅供调试使用,当前仅可用于 Siano USB 设备

Autoselec

t
a
n
c
i
l
l
a
r
y
d
r
i
v
e
r
s
(
t
u
n
e

rs,sensors,i2c,frontend

d

s

)

C

O

N

F

I

G

_

M

E

D

I

A

_

S

U

B

D

R

V

_AUTOSELECT

为多媒体设备驱动自动选择所有相关的辅助驱动(tuner[调谐器],sensor[传感器],视频编/解码器以及前端),以免去手动选择的麻烦.通常这是个好主意,建议选"Y".但是对于某些嵌入式环境来说,却希望去掉部分有用的辅助驱动以保持内核尽可能短小,这时应该选"N".

I2Cmod

u
l
e
f
o
r
I
R
C
O
N
F
I
G
_
V
I
D
E
O
_
I

R_I2C

大多数板子都通过 GPIO 总线连接红外线芯片,但少数板子却使用 I2C 总线连接.此项即是对 I2C 总线连接的红外线芯片提供支持.

Encoders,dececod

e

r

s

,

s

e

n

s

o

r

s

a

n

d

o

t

h

e

r

h

e

l

perchips

编码器,解码器,传感器,混频器...等辅助芯片

Sensorsused on

soc_camera_driver

用于"SoC Camera"(CONFIG_SOC_CAMERA)的各种传感器

Custo

mizeTVtuners

各种专用的电视调谐器

Customise

DVB Frontends

各种专用的数字电视前端

Graphics support

图形设备/显卡支持.对于不需要使用图形界面的服务器环境来说,必须的最小选项集取决于平台(BIOS/UEFI)和引导程序(GRUB/LILO/GRUB4DOS)的设置(全选"N"则屏幕将无任何显示).具体如下:(1)以 UEFI 方式启动的,一律都必须"CONFIG_FB=y,CONFIG_FB_EFI=y,CONFIG_FRAMEBUFFER_CONSOLE=y"[补充说明:对于 3.12 或以上版本,则应该是"CONFIG_X86_SYSFB=y,CONFIG_FB=y,CONFIG_FB_SIMPLE=y,FRAMEBUFFER_CONSOLE=y"];(2)以 BIOS+GRUB2 启动,且在

'grub.cfg'中明确将"gfxpayload"变量设置为非'text'值或者内核引导参数中存在'vga=...',那么必须"CONFIG_FB=y,CONFIG_FB_VESA=y,CONFIG_FRAMEBUFFER_CONSOLE=y"[补充说明:对于 3.12 或以上版本,则应该是"CONFIG_X86_SYSFB=y,CONFIG_FB=y,CONFIG_FB_SIMPLE=y,FRAMEBUFFER_CONSOLE=y"];(3)以 BIOS 方式启动的其他情况,必须"CONFIG_VGA_CONSOLE=y"

/dev/agpgart (AGP Support)

CONFIG_AGP

GART(图形地址重映射表)可以看做一种被各种显卡(不只是 AGP 显卡,还包括 PCI-E 显卡与集成显卡以及核心显卡)使用的"伪 IOMMU"(参见 CONFIG_GART_IOMMU 选项),它将物理地址不连续的系统内存映射成连续的"显存"供 GPU 使用.当物理显存容量不够时(大多数集成显卡甚至根本没有物理显存),GART 允许通过 DMA(直接内存访问)方式将这部分"显存"用于纹理贴图,Z 轴缓冲,ALPHA 混合,多边形网格生成...等各种 3D 操作.如果没有 GART 支持,OpenGL 直接渲染将会变得特别慢.GLX 与 DRI(CONFIG_DRM)也依赖于此.简而言之,需要使用图形化界面的人都应该选"Y".不需要图形界面的用户应该选"N".[注意]对于使用 304 或更老版本的 nVidia 闭源驱动的用户,如果使用的是某些老旧的芯片组,那么此处应该选"N".因为在这些特定的芯片组上,闭源驱动自己的 agpgart 实现(NvAGP)是更好的选

择.[提示]老版本的 AMD/ATI/NVIDIA 闭源驱动都曾经有自己的 agpgart 实现,但在新版本中都已经被移除.而 Intel 显卡一直使用的都是内核的 agpgart 实现.

AMD Opteron/Athlon64 on-CPU GART support

CONFIG_AGP_AMD64

该项仅适用于如下 AMD 处理器:(1)AMD K8 微架构 CPU[cpu family : 15] (2)AMD K10 微架构 CPU[cpu family : 16] (3)AMD 推土机微架构 CPU[cpu family : 21 并且 model: 小于 15].[提示]除了前面列出的三种 CPU 外,其他 AMD 处理器(例如 A4/A6/A8 系列 APU)并不需要此选项.具体支持的 CPU 可以查看"arch/x86/kernel/amd_nb.c"文件中的"AMD_NB_GART"常量的使用.

Intel 440LX/BX/GX, I8xx and E7x05 chipset support

CONFIG_AGP_INTEL

该项仅适用于:(1)某些 Intel 芯片组(440LX/BX/GX, 8xx 系列, E7205/E7505/E7221, 9xx 系列, 所有 3/4 系列). (2)全部 Intel 集成显卡(包括 Intel 核心显卡).具体支持的芯片组和集显可以查看"drivers/char/agp/intel-*"系列文件.[提示]如果你使用的既不是此处所列的芯片组(例如 Xeon 芯片组或者 5/6/7/8/9 系列芯片组)也不是 Intel 的集成显卡(例如 AMD/nVidia 独立显卡),那么应该选"N".

SiS chipset support

CONFIG_AGP_SIS

该项仅适用于 SiS 芯片组,但不包括上世纪古董级的 SiS 5591/5592 芯片组.

VIA chipset support

CONFIG_AGP_VIA

该项仅适用于 VIA 芯片组.具体支持的芯片组型号可以查看"drivers/char/agp/via-agp.c"中的"via_agp_device_ids"数组.

VGA Arbitration

CONFIG_VGA_ARB

图形设备是通过 I/O 或内存的特定地址范围进行访问的.大多数现代的显卡都允许对这个范围进行重新定位,但是某些基于 PCI 的"传统"VGA 设备仍然使用"硬编码"的地址范围,无法对其进行重新定位.如果系统上有多个这样的"传统"VGA 设备,就会造成地址冲突,这时候就需要进行 VGA 仲裁.此选项主要用于处理多个显卡(比如集成显卡和独立显卡)之间的切换.如果你的系统上有多个显卡,可以选"Y",否则应该选"N".详情参见"Documentation/vgaarbiter.txt"文档.

Maximum number of GPUs

CONFIG_VGA_ARB_MAX_GPUS

最多允许支持多少个显卡

Laptop Hybrid Graphics - GPU switching support

CONFIG_VGA_SWITCHEROO

支持多个显卡之间的切换(通常是在集显和独显之间),这项技术有多个不同的名称:"Hybrid Graphics","PowerXpress","HybridPower".这项特性主要用于笔记本,台式机一般不支持.

Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)

CONFIG_DRM

DRI(Direct Rendering Infrastructure)允许应用程序以高效安全的方式直接访问 GPU,主要用于硬件 3D 加速.不需要图形界面的用户应该选"N".桌面用户建议选"Y".[提示]KMS+DRI2+GEM+UXA+Wayland 是 Linux 图形革命的基石,这里还有两篇需要越墙的文章可以帮助加深对 Linux 图形技术的理解:(1)关于 Xorg 的一些整理,(2)厘清了 xorg 里的一些概念.

Enable legacy fbdev support for your modesetting driver

CONFIG_DRM_FBDEV_EMULATION

为传统的 fbdev 设备提供支持,由于 CONFIG_FRAMEBUFFER_CONSOLE 依赖于 fbdev 设备,因此务必选"Y",否则将会遭遇控制台黑屏.

Allow to specify an EDID data set instead of probing for it CONFIG_DRM_LOAD_EDID_FIRMWARE

EDID(扩展显示器识别数据)是一种 VESA(视频电子标准协会)制定的标准数据格式,其中包含有关显示器的各种参数:供应商信息,最大图像尺寸,颜色设置,厂商预设置,频率范围,显示器名,序列号字符串等等.EDID 保存在显示器的 PROM 或 EEPROM 中,显卡或 read-edid 一般通过 I2C 总线使用 DDC 协议进行读取.但是在某些脑残的硬件(显示器或显卡)上却不能正确获取 EDID 数据.此选项就是为了解决这个问题而设置的.此选项可以允许从"/lib/firmware/"目录加载 EDID 数据,或者将 EDID 数据在编译时直接嵌入内核.不确定的选"N".详情参见"Documentation/EDID/HOWTO.txt"文档.[提示]EDID 的继任者是 DisplayID,但目前 DisplayID 尚未被广泛应用.

I2C encoder or helper chips

I2C 编码器或辅助芯片

Chrontel ch7006 TV encoder CONFIG_DRM_I2C_CH7006

Chrontel ch7006 电视解码器支持.某些 nVidia 显卡上有这个芯片.此选项仅对 NVIDIA 显卡开源驱动(nouveau)有意义.

Silicon Image sil164 TMDS transmitter

CONFIG_DRM_I2C_SIL164

Silicon Image SIL164 最小化传输差分信号(TMDS)发送器,用于实现 DVI 信号的合成和发送.TMDS 不如 LVDS 应用广泛,仅在某些 nVidia 显卡上有出现.

NXP Semiconductors TDA998X HDMI encoder

CONFIG_DRM_I2C_NXP_TDA998X

NXP(恩智浦) TDA998X HDMI 发射器.用于实现 HDMI 信号的合成和发送.

3dfx Banshee/Voodoo3+

CONFIG_DRM_TDFX

3dfx Banshee/Voodoo3+ 系列古董显卡

ATI Rage 128

CONFIG_DRM_R128

ATI Rage 128 系列古董显卡

ATI Radeon

CONFIG_DRM_RADEON

Radeon 系列显卡开源驱动(radeon).[提示]R600 及更新的 GPU 需要额外的固件/微代码(radeon-ucode)的帮助才能使用开源

驱动.如果你打算在 R600 及更新的 GPU 上使用此开源驱动,那么建议选"M"而不是"Y".因为"Y"需要将微代码(通常位于"/lib/firmware/radeon/"目录)一起编译进内核(使用 CONFIG_EXTRA_FIRMWARE="radeon/GPU-MODEL.bin").[注意]如果你打算使用目前尚不支持 KMS 的 Radeon 闭源驱动(Catalyst/fglrx),那么此项必须选"N".

Enable userspace modesetting on radeon (DEPRECATED)

CONFIG_DRM_RADEON_UMS

禁用 KMS 支持.目的是为了兼容远古版本的 DDX 驱动.除非你有充足的理由,否则请选"N".

Nouveau (nVidia) cards

CONFIG_DRM_NOUVEAU

nVidia 系列显卡开源驱动(nouveau).[注意]如果你打算使用也许永远不会支持 KMS 的 nVidia 闭源驱动(nvidia-drivers),那么此项必须选"N".

Maximum debug level

CONFIG_NOUVEAU_DEBUG

最大调试级别,也就是最高允许显示的调试信息详细程度.取值范围是[0,7],数字越大,在内核中编入的调试信息就越多,建议设为"4",过大的级别会导致驱动运行缓慢.

Default debug level

CONFIG_NOUVEAU_DEBUG_DEFAULT

默认调试级别,必须小于等于 CONFIG_NOUVEAU_DEBUG 的值.数字越大,输出的调试信息就越详细,建议设为"2",过大的级别会导致驱动运行缓慢.

Support for backlight control

CONFIG_DRM_NOUVEAU_BACKLIGHT

允许调整显示器背光亮度,主要用于液晶显示屏等使用背光技术的显示器.建议选"Y".

Intel I810

CONFIG_DRM_I810

专为古董级 Intel 集成显卡(i810/i815)准备的驱动

Intel 8xx/9xx/G3x/G4x/HD Graphics

CONFIG_DRM_I915

Intel GMA(芯片组集成显卡)与 HD Graphics(核心集成显卡)开源驱动(intel).除了个别老古董(i810/i815)与基于 PowerVR 的芯片(Atom z5xx)之外,此驱动支持所有 Intel 集成显卡(包括 Atom 中的集成显卡).[提示]三大主流显卡厂商对 Linux 的驱动支持,Intel 是最彻底的,官方只提供开源驱动.

Enable modesetting on intel by default

CONFIG_DRM_I915_KMS

默认开启 KMS(Kernel Mode Setting)特性,作用是可以在内核级别(而不是用户级别)设置显示分辨率和颜色深度.KMS 使用了更新的技术,可以减少失真,增强 3D 性能,甚至可以使用内核的节能功能.KMS 是大势所趋,只要用户层软件不太旧(2010 年之后),都建议开启.[注意]开启此项后,应该:(1)关闭 CONFIG_FB_INTEL 选项并禁止加载任何 framebuffer 驱动(包括 CONFIG_FB_UVESA),(2)取消内核引导参数"vga=xxx"和"video=xxx",(3)必须开启 CONFIG_FRAMEBUFFER_CONSOLE 选项.

Enable legacy fbdev support for the modesetting intel driver

CONFIG_DRM_I915_FBDEV

使用此驱动为传统的 fbdev 设备提供支持,由于 CONFIG_FRAMEBUFFER_CONSOLE 依赖于 fbdev 设备,因此务必选"Y",否则将会遭遇控制台黑屏.

Enable preliminary support for prerelease Intel hardware by default

CONFIG_DRM_I915_PRELIMINARY_HW_SUPPORT

为尚未正式发布的显卡提供支持,相当于设置"i915.preliminary_hw_support=1"引导参数.一般应该选"N"

Enable userspace modesetting on Intel hardware (DEPRECATED)

CONFIG_DRM_I915_UMS

为古董级的 DDX 驱动提供用户空间模式设置支持.选"N".

Matrox g200/g400

CONFIG_DRM_MGA

Matrox G200, G400, G450 系列古董显卡

SiS video cards

CONFIG_DRM_SIS

SiS 630 系列古董显卡

Via unichrome video cards

CONFIG_DRM_VIA

Via unichrome 系列古董显卡

Savage video cards

CONFIG_DRM_SAVAGE

Savage3D/4/SuperSavage/Pro/Twister 系列古董显卡

DRM driver for VMware Virtual GPU

CONFIG_DRM_VMWGFX

VMware SVGA2 虚拟显卡驱动.支持 3D 加速,支持 KMS.如果你打算在 VMware 内使用图形化界面,建议选"Y".

Enable framebuffer console under vmwgfx by default

CONFIG_DRM_VMWGFX_FBCON

如果你使用的 VMware Tools 不太旧就选"Y".

Intel GMA5/600 KMS Framebuffer

CONFIG_DRM_GMA500

Intel 基于 Poulsbo 架构的集成显卡实验性支持.此类显卡并不常见,仅用于 Atom z5xx 系列处理器.

Display Link

CONFIG_ DRM_UDL

DisplayLink 是一个通过 USB 接口实现显示器连接到电脑的连接技术,可以非常简单的连接电脑和多个显示设备,常用于通过 USB 接口扩展虚拟的电脑的桌面.目前 DisplayLink 技术最多可以支持 6 台显示器同时显示 32 位色彩的任意分辨率画面.

AST server chips

CONFIG_DRM_AST

AST 系列显卡实验性支持.此种显卡仅出现在服务器环境.

Kernel mode setting driv

er

for

MGA

G200

serv

er

engi

nes

CONF

IG_D

RM_M

GAG2

00

MGA G200 系列服务器显卡芯片的 KMS 支持.[注意]仅用于服务器芯片,不要用于桌面芯片!此驱动需要 v0.3.0 版本的用户空间 modesetting 驱动.

Ci

rr

us

dr

iv

er

for

QEMU emulated device CONFIG_DRM_CIRRUS_QEMU

这是 QEMU 虚拟的 cirrus 显卡 KMS 驱动,仅可用于客户机中.千万不要用于真正的物理 cirrus 显卡.

QXLvirtualGPUCONFIG_

DRM_QXL

用于 SPICE 的虚拟桌面的 QXL 虚拟显卡(cirrus)KMS 支持.

Lowlevelvideoo

u

t

p

u

t

s

w

i

t

c

h

c

o

n

t

r

o

l

s

C

O

N

FIG_VIDEO_OUTPUT_CONTR

底层视频输出开关控制(通过 sysfs 接口).这是 ACPI 视频控制所依赖的功能,也被许多显卡驱动所依赖.可以选"N",若有其它驱动需要它,会被自动选中.

framebuffer_FBCONFIG_devices

帧缓冲(framebuffer)设备(/dev/fb*)是一种对图形硬件的抽象,它把屏幕上的所有像素点都直接映射到一段线性的内存空

间,这样就为软件提供了访问图形硬件的统一接口,这些软件不需要了解硬件的底层细节(例如寄存器),只要简单的改变相应内存位置的值,就能改变屏幕上显示的内容(颜色/亮度等).Xorg 的高度可移植性也就根源于此.图形界面用户必选.CJKTTY 补丁也依赖于它.

Enable firmware EDID

CONFIG_FIRMWARE_EDID

EDID(扩展显示器识别数据)保存在显示器的 PROM 或 EEPROM 中,显卡或 read-edid 一般通过 I2C 总线使用 DDC 协议进行读取.但是在某些脑残的硬件(显示器或显卡)上却不能正确获取 EDID 数据.此选项就是为了解决这个问题而设置的(参见 CONFIG_DRM_LOAD_EDID_FIRMWARE).开启此项后,将允许三种驱动(nvidiafb,i810fb,savagefb)通过 Video BIOS 获取 EDID.建议选"N",仅在你确实遭遇 EDID 读取失败,并且确实需要使用 nvidiafb/i810fb/savagefb 驱动的时候才需要考虑选"Y".[提示]EDID 的继任者是 DisplayID,但目前 DisplayID 尚未被广泛应用.

Framebuffer foreign endianness support

CONFIG_FB_FOREIGN_ENDIAN

如果你想混合使用不同字节序的主板和显卡(在 Little-Endian 主板上使用 Big-Endian 显卡,或者相反),可以选"Y".绝大多数人都应该选"N".除非你确实知道自己在做什么.

Enable Video Mode Handling Helpers

CONFIG_FB_MODE_HELPERS

使用 GTF 和 EDID 解析程序来帮助处理显示模式,建议选"N",若有其他选项依赖于它时,会自动选上.

Enable Tile Blitting Support

CONFIG_FB_TILEBLITTING

此项仅对 matroxfb 驱动有意义,建议选"N",若有其他选项依赖于它时,会自动选上

{虽然此处省略的各种 Framebuffer 驱动提供了对图形硬件的统一抽象,让 Xorg 不必与硬件直接对话,但这些 Framebuffer 驱动仅能提供 2D 功能,在如今 3D 硬件加速和视频硬件解码早已铺天盖地的情况下,让 Xorg 直接与 GPU 硬件对话才更符合潮流,而传统的 Framebuffer 驱动(CONFIG_FB_*)反而成为了绊脚石,不但没有必要与新的 DRI 驱动(CONFIG_DRM_*)共存,而且还会相互冲突.所以切勿选中这里省略的任何 Framebuffer 驱动.除非你确实知道自己在做什么.}

sVideodriversupportCON

FIG_EXYNOS_VIDEO

三星基于 ARM 构的 EXYNOS 处理器内置显卡

Back1

CONFIG_BACKLIGHT_LCD_SUP

P

P

O

R

T

背光与液晶支持.

Lowlevel LCD controls

CONFIG_LCD_CLASS_DEVICE

液晶(LCD)底层控制框架.用于控制对比度和 LCD 开关(而不是背光亮度).这些 LCD 硬件目前仅用于智能手机/平板电脑等嵌入式环境.

Lowlevel Backlight controls

CONFIG_BACKLIGHT_CLASS_DEVICE

背光(Backlight)底层控制框架.用于控制背光源的亮度和开关.选中此项后还需要从子项中选择特定于硬件的驱动.

Generic PWM based Backlight Driver

CONFIG_BACKLIGHT_PWM

液晶显示器(包括台式机和笔记本)的背光亮度调整方式有两种:(1)PWM 调光,(2)非 PWM 调光.目前主流的液晶显示器基本上都是 PWM 调光,仅有少数是非 PWM 调光型号(而且越来越少).

Apple Backlight Driver

CONFIG_BACKLIGHT_APPLE

基于 Intel 处理器的苹果 Macbook 笔记本和 iMac 台式机显示器背光控制

{其它省略的驱动仅用于智能手机/平板电脑等嵌入式环境}

Consoledisplaydr

控制台显示驱动.每个人都需要.下面的"VGA text"与"Framebuffer"至少应该选中一个.

VGA text console

CONFIG_VGA_CONSOLE

VGA 文本模式控制台.建议选"N".仅某些服务器环境可以考虑选"Y".

Enable Scrollback Buffer in System RAM

CONFIG_VGACON_SOFT_SCROLLBACK

标准的 VGA 控制台回滚缓冲区位于 VGA RAM 中,但是其空间非常小,并且是固定的.开启此项后,就可以在内存中开辟更大的屏幕回滚缓冲区,这将允许你回滚更多的屏幕(Shift+PageUp),但是

控制台的速度会略有下降.经常使用文本控制台的可以选"Y",不确定的选"N".

Scrollback Buffer Size (in KB)

CONFIG_VGACON_SOFT_SCROLLBACK_SIZE

在内存中开辟的屏幕回滚缓冲区大小.每个 80x25 屏幕需要 4KB 内存

Framebuffer Console support

CONFIG_FRAMEBUFFER_CONSOLE

基于 Framebuffer 的图形模式控制台.KMS 特性依赖于它.CJKTTY 补丁也依赖于它.桌面用户必选"Y"(使用了 CONFIG_DRM_*的用户必须开启),服务器以 UEFI 方式启动的也必选"Y".

Map the console to the primary display device

CONFIG_FRAMEBUFFER_CONSOLE_DETECT_PRIMARY

选"Y"表示自动将控制台映射到"主"显卡,选"N"表示自动将控制台映射到第一个加载的显卡驱动.无论是否选中此项,都可以通过"fbcon=map:N"内核引导参数更改映射关系.仅在系统拥有多个显卡时此选项才有意义.参见"Documentation/fb/fbcon.txt"文档.

Framebuffer Console Rotation

CONFIG_FRAMEBUFFER_CONSOLE_ROTATION

显示画面旋转,由于是纯软件方式实现,所以会大大降低显示速度,除非你确实需要,否则建议选"N".若使用了 CJKTTY 补丁,则必须选"N".

Support for the Framebuffer Console Decorations

CONFIG_FB_CON_DECOR

允许在控制台上显示背景图像,例如在系统启动时,在一堆滚动的字符背后显示漂亮的背景图像.当然,要实现这个功能,还需要用户空间程序的帮助.详见"Documentation/fb/fbcondecor.txt"文档以及 fbsplash 的 wiki 页.

启动时显示 linux 的 logo(一幅企鹅图像),企鹅的数量表示内核检测到的 CPU 数目,喜欢炫一下的就选吧.子项是三种不同质量的图片,分别是黑白,16 色,224 色.按需选择.

Sound card support

CONFIG_SOUND

声卡支持

Preclaim OSS device numbers

CONFIG_SOUND_OSS_CORE_PRECLAIM

开启此项后,只要 OSS 支持被开启,无论相应的模块是否被加载,内核都会预先声明所有 OSS 设备号.当其中一个设备被打开时,将会尝试使用"sound-slot/service-*"与"char-major-*"两种别名去加载相应的模块.关闭此项后, 内核将仅声明实际使用

中的 OSS 设备号.当打开一个不存在的设备时,将会仅尝试使用标准的"char-major-*"别名去加载相应的模块.由于"sound-slot/service-*"将会在未来移除,此选项仅是一个为了兼容性而保留的过渡选项,未来会被移除(相当于设为"N").

Advanced Linux Sound Architecture

CONFIG_SND

ALSA(高级 Linux 声音架构)是内核默认的声音子系统.ALSA 除了提供了声音设备的驱动,还提供了一个用户空间的函数库,这样用户空间程序就可以通过统一的 API 使用驱动功能,而不必直接与内核驱动交互.[吐槽]Linux 音频系统,比意大利面条更混乱的系统!

Sequencer support

CONFIG_SND_SEQUENCER

MIDI 音序器支持,如果你是 MIDI 玩家,请选"Y",但如果你不知道 MIDI 是什么,请选"N".

Sequencer dummy client

CONFIG_SND_SEQ_DUMMY

除非你要同时连接到多个 MIDI 设备或应用程序,否则请不要选中

OSS Mixer API

CONFIG_SND_MIXER_OSS

模拟 OSS 混音器 API(/dev/mixer*),某些老旧的程序仍然使用它,建议不选

OSS PCM (digital audio) API

CONFIG_SND_PCM_OSS

模拟 OSS 数字音频(PCM)API(/dev/dsp*),某些老旧的程序仍然使用它,建议不选.

OSS PCM (digital audio) API - Include plugin system

CONFIG_SND_PCM_OSS_PLUGINS

让 ALSA 模拟的 OSS PCM API 支持 channel/format/rate 的转换.选"N",除非你确实知道为什么要选"Y".

OSS Sequencer API

CONFIG_SND_SEQUENCER_OSS

模拟 OSS 音序器(/dev/sequencer,/dev/music),某些老旧的程序仍然使用它,建议不选

HR-timer backend support

CONFIG_SND_HRTIMER

允许将高精度定时器(CONFIG_HIGH_RES_TIMERS)用作 ALSA 高精度时间源,建议选中

Use HR-timer as default sequencer timer

CONFIG_SND_SEQ_HRTIMER_DEFAULT

将高精度定时器(HR-timer)当作默认的时序脉冲发生器时间源,建议选中

Dynamic device file minor numbers

CONFIG_SND_DYNAMIC_MINORS

动态分配 ALSA 设备的次设备号.如果你有 8 个以上的声卡,可以选"Y",否则应该选"N".

Support old ALSA API

CONFIG_SND_SUPPORT_OLD_API

支持已被废弃的老旧版本的 ALSA PCM API,选"N".

Verbose procfs contents

CONFIG_SND_VERBOSE_PROCFS

仅供调试使用

Verbose printk

CONFIG_SND_VERBOSE_PRINTK

仅供调试使用

Debug

CONFIG_SND_DEBUG

仅供调试使用

Generic sound devices

CONFIG_SND_DRIVERS

通用声音设备

PC-Speaker support (READ HELP!)

CONFIG_SND_PCSP

如果你有声卡,务必选"N".如果你的系统没有声卡,仅在认真阅读了帮助之后,确实知道自己在干什么的情况下,才可以开启此项.

Dummy (/dev/null) soundcard

CONFIG_SND_DUMMY

仅供调试使用

Generic loopback driver (PCM)

CONFIG_SND_ALOOP

PCM 环回(loopback)设备非常类似于网卡的环回接口(127.0.0.1),它会将输入的音频流原封不动的返回给用户空间.PCM 回环设备常用来将 A 程序输出的音频流作为 B 程序的输入(A 以写模式打开环回设备,而 B 以读模式打开),比如用 B 程序记录 A 程序的输出,或做进一步的处理.

Virtual MIDI soundcard

CONFIG_SND_VIRMIDI

虚拟 MIDI 驱动,允许将使用原始 MIDI 设备的应用程序连接到音序器客户端,如果你不知道 MIDI 是什么就选"N".

{此处省略几种 MIDI 设备(事实上大部分人都没有这些设备)}

AC97 Power-Saving Mode

CONFIG_SND_AC97_POWER_SAVE

AC97(Audio Codec 97)自动节能模式支持.在此模式下,如果音频设备闲置超过"/sys/module/snd_ac97_codec/parameters/power_save"设定的秒数("0"表示关闭节能模式),那么驱动程序将会关闭音频设备以节约电力.建议选"Y".详见"Documentation/sound/alsa/powersave.txt"文档.

Default time-out for AC97 power-save mode

CONFIG_SND_AC97_POWER_SAVE_DEFAULT

默认的超时秒数,也就是"/sys/module/snd_ac97_codec/parameters/power_save"的默认值."0"表示关闭节能模式.建议设为"10"这个久经考验的合理数字.

ISA sound devices

CONFIG_SND_ISA

基于 ISA 总线的声卡,已经绝种了.

PCI sound devices

CONFIG_SND_PCI

基于 PCI 总线的声卡,绝大多数声卡都是 PCI 接口

{此处省略的声卡按实际情况选择即可(都是些比较旧的 AC97 声卡)}

Intel HD Audio

CONFIG_SND_HDA_INTEL

符合 Intel HD Audio 规范的声卡是目前的主流声卡.如果选"M",那么下面的每个驱动也都会被编译成模块,如果选"Y",那么下面的每个驱动也都会直接编译进内核.

Pre-allocated buffer size for HD-audio driver

CONFIG_SND_HDA_PREALLOC_SIZE

为 HD-audio 驱动程序预先分配的缓冲区大小(kB),较大的值拥有更好的性能,例如对于使用 PulseAudio 声音服务器的系统来说,推荐使用"4096".默认值"64"仅仅是为了历史兼容的原因.[提示]ALSA+PulseAudio 是目前的主流搭配.

Build hwdep interface for HD-audio driver

CONFIG_SND_HDA_HWDEP

为 HD-audio 驱动添加 hwdep 接口.仅用于调试目的

Support digital beep via input layer

CONFIG_SND_HDA_INPUT_BEEP

为 HD-audio 驱动添加数字蜂鸣(beep)接口.如果你的主板没有蜂鸣器(不是能够播放音乐的扬声器),可以考虑选"Y".

Digital beep registration mode (0=off, 1=on)

CONFIG_SND_HDA_INPUT_BEEP_MODE

设为"0"表示默认禁用数字蜂鸣接口,设为"1"表示默认启用数字蜂鸣接口.

Support jack plugging notification via input layer

CONFIG_SND_HDA_INPUT_JACK

通过输入层支持 JACK 插件通知.JACK 是一个比 PulseAudio 更专业的声音服务器,重点是低延迟,是专业音频软件(例如:Ardour,Rezound,LinuxSampler)首选的音频服务器.如果你打算使用 JACK,可以选"Y".[提示]如果要将 JACK 和 PulseAudio 一起使用,需要安装 PulseAudio 的 JACK 支持模块.

Support initialization patch loading for HD-audio

CONFIG_SND_HDA_PATCH_LOADER

仅用于调试目的

{此处省略的 HD-audio 声卡按实际情况选择即可.[提示]如果 CONFIG_SND_HDA_INTEL 被编译为模块,这里的每一个驱动也都会被编译成模块.}

Build HDMI/DisplayPort HD-audio codec support

CONFIG_SND_HDA_CODEC_HDMI

在 HD-audio 驱动中添加 HDMI 和 DisplayPort 支持.如果你需要使用 HDMI/DisplayPort 接口,可以选"Y".

Enable generic HD-audio codec parser

CONFIG_SND_HDA_GENERIC

通用 HD-audio 编解码器(codec)支持,必选.

Default time-out for HD-audio power-save mode

CONFIG_SND_HDA_POWER_SAVE_DEFAULT

HD-audio 自动节能模式默认的超时秒数."0"表示关闭节能模式.建议设为"10"这个久经考验的合理数字.详见"Documentation/sound/alsa/powersave.txt"文档与 CONFIG_SND_AC97_POWER_SAVE_DEFAULT 选项.

SPI sound devices

CONFIG_SND_SPI

基于 SPI 总线的声卡,仅出现在嵌入式设备上

USB sound devices

CONFIG_SND_USB

基于 USB 总线的声卡,主要是外接声卡,并不常用

FireWire sound devices
CONFIG_SND_FIREWIRE

基于 IEEE-1394/FireWire/iLink 总线的声卡,主要用于苹果的产品

PCMCIA sound devices
CONFIG_SND_PCMCIA

基于 PCMCIA 接口的声卡,主要是外接声卡,并不常用

ALSA for SoC audio support
CONFIG_SND_SOC

SoC 系统音频设备支持,重点是节能支持.仅用于嵌入式设备

Open Sound System (DEPRECATED)
CONFIG_SOUND_PRIME

OSS 早已被废弃(已被 ALSA 取代).选"N".

HID support

HID(人机接口设备)是一种定义计算机如何与人类交互的规范,常与 USB 或蓝牙搭配使用,常见的设备有:键盘,鼠标,触摸板,游戏杆,遥控器,蓝牙耳机,游戏手柄,手写板,等等.不过 HID 设备不一定要有人机接口,只要符合 HID 规范,就是 HID 设备.

HID bus support

CONFIG_HID

HID(human interface device)总线及通用 HID 层.要使用 HID 设备就必须开启.[提示]PS/2 接口的鼠标和键盘不是 HID 设备,USB 或蓝牙接口的才是 HID 设备.

Battery level reporting for HID devices

CONFIG_HID_BATTERY_STRENGTH

为那些支持 power_supply 类的 HID 电池,向用户空间报告电池的剩余电量(可以通过 upower 工具显示).

/dev/hidraw raw HID device support

CONFIG_HIDRAW

如果你想支持那些严格说来并不属于人机交互设备的硬件(使用额外的/dev/hidraw 接口),例如显示控制装置(monitor control)或不间断电源(UPS)以及某些罗技的无线鼠标接收器,可以选"Y".与 CONFIG_USB_HIDDEV 选项(/dev/hiddev)相比,/dev/hidraw 设备直接无视一切 hid 事件(既不解析也不查找),这样就允许应用程序直接处理和操作原始的 hid 事件,从而

避免使用用户层 libhid/libusb 库.详见"Documentation/hid/hidraw.txt"文档.

User-space I/O driver support for HID subsystem

CONFIG_UHID

HID 子系统需要两种驱动:(1)"HID I/O Driver"是特定于硬件的驱动,直接与底层总线交互,并向"HID Device Driver"提供了一致接口用于收发 HID 数据.(2)"HID Device Driver"是硬件无关的通用驱动,其任务是按照 HID 规范解析和处理来自于"HID I/O Driver"的 HID 数据,并将组装好的数据通过"HID I/O Driver"提供的统一接口发送给底层硬件.开启此项后,将允许在用户空间实现"HID I/O Driver".不确定的选"N".详见"Documentation/hid/uhid.txt"文档.

Generic HID driver

CONFIG_HID_GENERIC

HID 总线通用驱动,也就是前面说的"HID Device Driver".它实现了对各种常见 HID 协议的支持:键盘,鼠标,游戏杆,手写板,数字画板.不确定的选"Y".

Special HID drivers

各种不严格遵守 HID 协议的"HID Device Driver"

{此处省略的硬件按实际情况选择即可}

Lenovo ThinkPad USB Keyboard with TrackPoint

CONFIG_HID_LENOVO_TPKBD

带有"小红帽(TrackPoint)"的联想(Lenovo)ThinkPad USB 键盘.

Logitech devices

CONFIG_HID_LOGITECH

某些并不完全遵从 HID 标准的罗技(Logitech)外设

HID Multitouch panels

CONFIG_HID_MULTITOUCH

HID 多点触控(Multitouch)板的通用支持

HID Sensors framework support

CONFIG_HID_SENSOR_HUB

HID 传感器支持框架.详见"Documentation/hid/hid-sensor.txt"文档

USB HID support

基于 USB 接口的 HID 设备,这是目前最常见的 HID 设备

USB HID transport layer

CONFIG_USB_HID

特定于 USB 接口的"HID I/O Driver".用于和 USB 总线上的硬件进行交互.只要你想使用任何基于 USB 接口的 HID 设备(键盘,鼠标,游戏杆,手写板,手绘板,不间断电源(UPS),显示控制装置

(monitor control),等等),就必须选"Y".[例外]在嵌入式环境中使用的 HIDBP(HID Boot Protocol)键盘和鼠标不在此列,而且两者也不能共存.

PID device support

CONFIG_HID_PID

PID 兼容的力反馈设备,例如:Microsoft Sidewinder Force Feedback 2

/dev/hiddev raw HID device support

CONFIG_USB_HIDDEV

如果你想支持那些严格说来并不属于人机交互设备的硬件(使用额外的/dev/usb/hiddevX[char 180:96~111]接口),例如显示控制装置(monitor control)或不间断电源(UPS)以及某些罗技的无线鼠标接收器,可以选"Y".参见 CONFIG_HIDRAW 选项.

USB HID Boot Protocol drivers

如果你有绝对的把握确信不为自己的键盘和鼠标使用常规的 HID 驱动,而要使用 Boot Protocol 模式的 HID 驱动(常见于嵌入式环境)就选吧

I2C HID support

基于 I2C 总线的 HID 设备

HID over I2C transport layer

CONFIG_I2C_HID

特定于 I2C 总线的"HID I/O Driver".用于和 I2C 总线上的硬件进行交互.只要你想使用任何基于 I2C 总线的 HID 设备(键盘,触摸板,触摸屏,等等),就必须选"Y".I2C-HID 主要用于嵌入式设备.

USB support

CONFIG_USB_SUPPORT

通用串行总线(Universal Serial Bus)的目标是统一电脑的外设接口.目前几乎找不到没有 USB 接口的电脑,而且各种智能设备也大多带有 USB 接口.不要犹豫,选"Y".

Support for Host-side USB

CONFIG_USB

主机端(Host-side)USB 支持.通用串行总线(USB)是一个串行总线子系统规范,它比传统的串口速度更快并且特性更丰富(供电,热插拔,最多可接 127 个设备等),其目标是统一 PC 外设接口.USB 总体上呈现一种树型结构,USB 的"Host"(主设备)被称为"根"(也可以理解为是主板上的 USB 控制器),USB 的"Slave"(从设备)被称为"叶子",而内部的节点则称为"hub"(集线器).只要使用任何 USB 设备都必须选中此项.另外,你还需要从下面选中至少一个 HCD(Host Controller Driver),比如适用于 USB1.1 的"UHCI HCD support"或"OHCI HCD support",适用于 USB2.0 的"EHCI HCD (USB 2.0) support".如果你拿不准的话把他们都选中一般也不会出问题.

如果你的系统有设备端的 USB 接口(也就是你的系统可以作为"叶子"使用),请到"USB Gadget"中进行选择.

USB verbose debug messages

CONFIG_USB_DEBUG

仅供调试使用

USB announce new devices

CONFIG_USB_ANNOUNCE_NEW_DEVICES

在 syslog 中记录每个新接入系统的 USB 设备的详细标识信息(idVendor,idProduct,Manufacturer,Product,SerialNumber),主要用于系统调试.不确定的选"N".

Enable USB persist by default

CONFIG_USB_DEFAULT_PERSIST

根据 USB 规范,当 USB 总线被挂起(休眠)后,它必须继续提供挂起电流(1-5 毫安),以确保 USB 设备能保持其内部状态,并且 USB 集线器(HUB)能够检测连接变化(设备插入和拔出).这在技术上被称为"电力会话"(power session).如果一个 USB 设备的电力会话被中断,那么系统必须按照该设备已经被拔出进行处理,这是一种保守的做法,因为没有挂起电流,计算机不可能知道外围设备究竟发生了什么变化:也许依然保持连接,也许已经被拔出并在同一端口上插入了一个新设备.系统必须做最坏的打算.默认情况下,Linux 的行为符合 USB 规范的要求.当整个电脑进入休眠状态(例如挂起到硬盘)时,包括 USB 总线在内所有总线

都将掉电,然后当系统被唤醒,所有 USB 设备都会被当做在休眠前就已经被拔出来处理.这样做始终是安全的,并且也是"官方正确"的做法.对于大多数 USB 设备来说,这样做没有任何问题,但是对于 USB 存储设备(例如移动硬盘/U 盘)来说,如果在休眠前有尚未卸载的文件系统(特别是根文件系统),当系统被唤醒之后,由于无法访问该文件系统,系统可能会立即崩溃!其实不只有掉电,只要"power session"被中断(例如 BIOS 在唤醒过程中重置了 USB 控制器),都会导致这种故障.此选项(USB-persist)就是为了解决这个问题而设置的,虽然解决的不甚完美(参见"Documentation/usb/persist.txt"),但是依然推荐选"Y",除非你确实有选"N"的理由.当然,最保险的做法是在休眠之前先卸载所有 USB 设备上的文件系统,而如果根文件系统位于 USB 设备上,就根本不使用任何休眠功能(不论是挂起到硬盘还是挂起到内存).

Dynamic USB minor allocation

CONFIG_USB_DYNAMIC_MINORS

动态分配 USB 设备的次设备号(仅限于主设备号为 180 的字符设备[通常位于"/dev/usb/"目录下]).除非你有超过 16 个同类型(仅限:打印机,鼠标,扫描仪)的 USB 设备,否则应选"N".[提示]即使你有 100 个 U 盘或者 USB 移动硬盘,也不需要开启此项,因为他们不是"主设备号为 180 的字符设备".

OTG support

CONFIG_USB_OTG

传统上,码照相机,手机,打印机,播放器,移动硬盘等设备之间要交换数据,都要作为 PC 的外围设备,在 PC 的控制下进行数据交换.一旦离开了 PC,由于没有一个设备能够充当 PC 的"Host"角色,所以无法直接通信.USB-OTG(On-The-Go)就是为了解决这个问题而诞生的,它是 USB2.0 规格的补充标准,支持"双角色"设备(既可以当 Host,也可以当 Slave),从而实现外围设备之间的数据传送.例如,将数码相机直接连接到打印机上将相片打印出来.仅在你的主板上有 Mini-AB/Micro-AB 接口(目前仅用于嵌入式设备)时才需要选"Y".

Rely on OTG and EH Targeted Peripherals List

CONFIG_USB_OTG_WHITELIST

将"otg_whitelist.h"文件用作"OTG Targeted Peripherals List"(外设白名单),白名单之外的 USB 外设将按照 OTG 规范的要求不被枚举(也就是初始化).同样,"Embedded Host"也只支持限定的外设.如果选"N",那么白名单之外的外设也同样会被枚举(但会产生一个警告),这将大大方便嵌入式产品的开发.

Disable external hubs

CONFIG_USB_OTG_BLACKLIST_HUB

选"Y"将禁止枚举(也就是初始化)外部 USB 集线器(HUB).这样,OTG 主机就可以通过省去对外部集线器的支持,降低系统软硬件的成本.不确定的选"N".

USB ULPI PHY interface support

CONFIG_USB_ULPI_BUS

ULPI(UTMI+ Low Pin Interface)是一种 2005 年开始兴起的通用 USB 2.0 PHY 接口.可有效地减少主机/外设/On-The-Go(OTG)USB 收发器的针脚数量(从 32 个减少到 12 个).仅用于嵌入式设备.

USB Monitor

CONFIG_USB_MON

选"Y"后,将可以捕获特定 USB 外设与 USB 主控器之间的数据流量,usbdump 和 usbmon 工具依赖于此项.详见"Documentation/usb/usbmon.txt"文档.

Enable Wireless USB extensions

CONFIG_USB_WUSB

主机端的 WUSB(无线 USB)支持.

Support WUSB Cable Based Association (CBA)

CONFIG_USB_WUSB_CBAF

WUSB CBA(Cable Based Association)是一项保障主机和WUSB 设备之间通信安全的技术.如果你的 WUSB 设备在建立无线连接前必须先建立有线连接,可以选"Y".

Enable CBA debug messages

CONFIG_USB_WUSB_CBAF_DEBUG

仅供调试使用

Cypress C67x00 HCD support

CONFIG_USB_C67X00_HCD

Cypress C67x00 (EZ-Host/EZ-OTG) USB 1.1 "双角色"控制器

xHCI HCD (USB 3.0) support

CONFIG_USB_XHCI_HCD

xHCI(eXtensible Host Controller Interface)就是当下大红大紫的 USB3.0(SuperSpeed USB)主机控制器规范.[提示]因为 xHCI 移除了 EHCI 中为兼容 USB1.1 而引入的"Companion"模式,所以仅用一个单独的 xHCI 驱动就可以兼容所有 USB3.0/2.0/1.1 外设.也就是说,对于蓝色的 USB3.0 接口来说,开启此项之后,就不需要再额外开启 EHCI/OHCI/UHCI 选项了.[注意]如果你的电脑上除了蓝色的 USB3.0 接口,还存在黑色的 USB2.0 接口,那么你仍然需要开启 EHCI/OHCI/UHCI 选项,除非你不想使用这些黑色的 USB2.0 接口.

Debugging for the xHCI host controller

CONFIG_USB_XHCI_HCD_DEBUGGING

仅供调试使用

Generic xHCI driver for a platform device

CONFIG_USB_XHCI_PLATFORM

通用 platform 设备的 xHCI 驱动.仅用于嵌入式环境.不确定的选"N".

EHCI HCD (USB 2.0) support

CONFIG_USB_EHCI_HCD

EHCI(Enhanced Host Controller Interface)就是渐成昨日黄花的 USB2.0(HighSpeed USB)主机控制器规范.[提示]因为 EHCI 通过"Companion"模式来支持 USB1.1 设备,所以一般还需要额外再开启 OHCI 或 UHCI 选项(除非你不想兼容任何 USB1.1 设备).详见"Documentation/usb/ehci.txt"文档.

Root Hub Transaction Translators

CONFIG_USB_EHCI_ROOT_HUB_TT

带有 USB2.0 接口的主板上都有一个"根集线器"(Root Hub)以允许在无需额外购买 hub 的情况下就可以提供多个 USB 插口,而大多数主板还在其中集成了事务转换(Transaction Translator)功能,这样就不需要再额外使用一个 OHCI 或 UHCI

控制器来兼容 USB1.1,建议选"Y",除非你不想兼容任何 USB1.1 设备.

Improved Transaction Translator scheduling

CONFIG_USB_EHCI_TT_NEWSCHED

改变周期性调度代码的工作方式,当多个 USB1.1 设备连接在同一个 USB2.0 集线器上时,可以获得更高的运行速度.建议选"Y"

Generic EHCI driver for a platform device

CONFIG_USB_EHCI_HCD_PLATFORM

通用 platform 设备的 EHCI 驱动.仅用于嵌入式环境.不确定的选"N".

OHCI HCD support

CONFIG_USB_OHCI_HCD

OHCI(Open Host Controller Interface)是主要用于嵌入式环境的 USB1.1(LowSpeed/FullSpeed USB)主机控制器规范.但也存在于某些老旧的 SiS 芯片组的 PC 上.

UHCI HCD (most Intel and VIA) support

CONFIG_USB_UHCI_HCD

UHCI(Universal Host Controller Interface)是主要用于PC环境的USB1.1(LowSpeed/FullSpeed USB)主机控制器规范.

{此处省略的USB控制器请按照实际硬件状况选择(基本上都仅用于嵌入式环境)}

Wireless USB Host Controller Interface (WHCI) driver

CONFIG_USB_WHCI_HCD

WHCI(Wireless USB Host Controller Interface)是无线USB主机控制器规范.目前市场上带有WUSB主控器的主板很少.不确定的选"N".

Host Wire Adapter (HWA) driver

CONFIG_USB_HWA_HCD

USB 接口的 3G/4G 无线上网卡(通常需要搭配 SIM 卡使用),常见制式有:WCDMA/LTE/HSPA 等.

BCMA usb host driver

CONFIG_USB_HCD_BCMA

BCMA(Broadcom specific AMBA)总线上的 EHCI/OCHI 主机控制器支持.仅用于嵌入式环境.

SSB usb host driver

CONFIG_USB_HCD_SSB

BCMA(Broadcom specific AMBA)总线上的 EHCI/OCHI 主机控制器支持.仅用于嵌入式环境.

Inventra Highspeed Dual Role Controller (TI, ADI, ...)

CONFIG_USB_MUSB_HDRC

一系列基于 Mentor Graphics 公司 silicon IP 核的 USB 控制器.仅用于嵌入式环境.

Renesas USBHS controller

CONFIG_USB_RENESAS_USBHS

一系列基于 Renesas 公司 USBHS IP 核的 USB 控制器.仅用于嵌入式环境.

USB Modem (CDC ACM) support

CONFIG_USB_ACM

USB 接口的猫或 ISDN 适配器,基本没人用的东西.

USB Printer support CONFIG_USB_PRINTER

USB 接口的打印机,这是主流的打印机

USB Wireless Device Management support CONFIG_U

SB_W

DM

为符合 CDC(Communication Device CIass)和 WMC(Wireless Mobile Communication)标准的手机提供 WMC 设备管理支持,这样你可以在这些手机上使用 AT 命令(被所有调制解调器制造商采用的一个调制解调器命令语言).

US

B

Te

st

an

d

Me

as

ur

em

en

t

Cl

as

s

su

pp

or

t

CO

NF

IG

_U

SB

_T

MC

USBTMC(USB Test and Measurement Class)协议支持.主要适用于测试仪器的 USB 通信开发.不确定的选"N".

U

S

B

M

a

s

s

S

t

o

r
a
g
e
s
u
p
p
o
r
t
C
O
N
F
I
G
_
U
S
B
_

S

T

O

R

A

G

E

USB 存储设备(U 盘,USB 硬盘,USB 软盘,USB 光盘,USB 磁带,记忆棒,数码相机,读卡器[包括某些笔记本内置的 SD 卡读卡器]等等).该选项依赖于 CONFIG_SCSI 和 CONFIG_BLK_DEV_SD 选项.选"Y",除非你确实知道自己在干什么.

USB Mass Storage verbose debug

CONFIG_USB_STORAGE_DEBUG

仅供调试使用

{省略的部分请按照自己实际使用的硬件选择(事实上大部分人都没有这些设备)}

U

S

B

M

u

s

tekMDC8000Digital Camera

s
u
p
p
o
r
t
C
O
N
F
I
G
_
U
S
B
_
M
D
C
8

rsupport CONFIG_USB_MIC

R

O

T

E

K

几款上世纪生产的扫描仪

DesignWare USB3 DRD Core Support

CONFIG_USB_DWC3

基于 DesignWare USB3 IP 核的 USB3.0 控制器.仅用于嵌入式环境.

ChipIdea Highspeed Dual Role Controller

CONFIG_USB_CHIPIDEA

基于 ChipIdea silicon IP 核的 USB2.0 控制器.仅用于嵌入式环境.

USS720 parport driver

CONFIG_USB_USS720

一种 USB 转并口的转换设备.不确定的选"N".

USB Serial Converter support

CONFIG_USB_SERIAL

USB-串口转换支持:(1)USB-串口转换器,(2)连接在 USB 口上的串口设备.详情参见"Documentation/usb/usb-serial.txt"文档.不确定的选"N".

{此处省略的各种杂七杂八的 USB 设备,要么是老古董,要么是很罕见,不确定的可以全部选"N"}

USB testing driver

CONFIG_USB_TEST

仅供调试使用

USB DSL modem support

CONFIG_USB_ATM

USB DSL modem 已经是绝迹的古董猫了

USB Physical Layer drivers

CONFIG_USB_PHY

这类设备仅在嵌入式系统上存在

USB Gadget Support

CONFIG_USB_GADGET

USB 是一个主/从协议,一个主机最多控制 127 个外设,其结构是非对称的,所以你无法把一个"到主机"的插头连接到外设上.Linux 既可以在 USB 主机上运行,也可以在 USB 外设上运行.外设 USB 控制器可以是单独的芯片,也可以是集成在 CPU 中的微控制器,而常见的主机端控制器通常集成在芯片组的南桥中

(xHCI/EHCI/OHCI/UHCI).如果你打算在外设中运行 Linux,那么就必须开启此项,然后还需要为外设段的总线控制器配置一个硬件驱动,以及一个用于外设协议的"配件驱动".不过对于大多数人来说,并不将 Linux 运行于外设端,因此可以放心的选"N".仅那些嵌入式设备(例如智能手机)上运行的 Linux 才可能由此需求.

Ultra Wideband devices

CONFIG_UWB

UWB(Ultra Wideband)是一种高带宽,低能耗,点对点,抗干扰性能强的无载波通信技术.UWB 在较宽的频谱(3.1-10.6GHz)上,使用极低的功率(约为蓝牙的 1/20),以时间间隔极短(小于 1ns)的脉冲信号进行通信.UWB 主要应用于室内通信(2 米范围内实现 480Mbps 速率,10 米范围内实现 110Mbps 速率),例如作为 WUSB(Wireless USB)协议的传输层.如果你有 UWB 无线控制器,可以选"Y",不确定的选"N".详见"Documentation/usb/WUSB-Design-overview.txt"文档.

MMC/SD/SDIO card support

CONFIG_MMC

MMC(MultiMediaCard)/SD(Secure Digital)/SDIO(Secure Digital I/O)主机控制器。[提示]虽然许多笔记本上有 SD 卡插槽,但其实它们大多使用的是 CONFIG_USB_STORAGE 驱动,而不是这里的驱动.

MMC debugging

CONFIG_MMC_DEBUG

仅供调试使用

Assume MMC/SD cards are non-removable

(DANGEROUS)

CONFIG_MMC_UNSAFE_RESUME

假定在系统休眠的过程中,所有 MMC/SD/SDIO 卡依然插在各自的插槽上没有变动.也许只有嵌入式系统才可以做这样的假定.不确定的选"N".参见 CONFIG_USB_DEFAULT_PERSIST 选项.

MMC host clock gating

CONFIG_MMC_CLKGATE

尝试激进的"gate the clock to the MMC card"(啥意思?).这样当 MMC 卡不使用的时候,就可以进入节电状态.主机控制器必须支持此特性.不确定的选"N".

MMC block device driver

CONFIG_MMC_BLOCK

MMC 块设备驱动.基本上 MMC 卡都是作为块设备(就像 U 盘一样)使用.所以只要使用 MMC 卡就应该开启.

Number of minors per block device

CONFIG_MMC_BLOCK_MINORS

为每个 MMC 块设备保留的次设备号数量.取值范围是[4,256].这里设置的值应该等于"最大可能的分区数+1".因为总的次设备号只有 256 个,所以最大能支持的 MMC 块设备数量就等于 256 除以此处设置的值.默认值"8"可以保证最大的向后兼容性.不确定的请保持默认值.

Use bounce buffer for simple hosts

CONFIG_MMC_BLOCK_BOUNCE

为 SD/MMC 控制器提供更多的缓存(最大 64KB),从而可以大幅提升其性能.建议选"Y".

SDIO UART/GPS class support

CONFIG_SDIO_UART

实现了 UART 类的 SDIO 卡支持.包括那些表现的像 UART 一样的 GPS 类支持.主要用于嵌入式设备.不确定的选"N".

MMC host test driver

CONFIG_MMC_TEST

仅供调试使用

Secure Digital Host Controller Interface support

CONFIG_MMC_SDHCI

通用 SD 主控支持.笔记本电脑上用的 SD 主控(TI(德州仪器)/Ricoh(理光)/Toshiba(东芝)等厂商)基本上都是这个驱

动.选中此项后,还需要选中相应的总线驱动(见下,通常是CONFIG_MMC_SDHCI_PCI).

SDHCI support on PCI bus

CONFIG_MMC_SDHCI_PCI

PCI 总线的 SD 主控支持,目前的笔记本的 SD 主控基本都接在 PCI 总线上.

Ricoh MMC Controller Disabler

CONFIG_MMC_RICOH_MMC

用于修正 Ricoh(理光)MMC 主控的 bug,如果你需要使用 Ricoh 主控,就选"Y".

SDHCI support for ACPI enumerated SDHCI controllers

CONFIG_MMC_SDHCI_ACPI

专用于"ACPI Compatibility ID"等于"PNP0D40"的 SD 主控,以及"ACPI Hardware ID"等于"INT33C6,INT33BB,80860F14"的 SD 主控.

SDHCI platform and OF driver helper

CONFIG_MMC_SDHCI_PLTFM

基于 platform 总线和 OpenFirmware 的 SD 主控.

{此处省略的 SD 主控请按照实际使用的芯片进行选择}

MMC/SD/SDIO over SPI

CONFIG_MMC_SPI

基于 SPI 总线的 MMC/SD/SDIO 主控.仅用于嵌入式环境.

Sony MemoryStick card support

CONFIG_MEMSTICK

Sony 记忆棒是一种 Sony 专用的存储设备.

MemoryStick debugging

CONFIG_MEMSTICK_DEBUG

仅供调试使用

Allow unsafe resume (DANGEROUS)

CONFIG_MEMSTICK_UNSAFE_RESUME

假定在系统休眠的过程中,所有记忆棒依然插在各自的插槽上没有变动.也许只有嵌入式系统才可以做这样的假定.不确定的选"N".参见 CONFIG_USB_DEFAULT_PERSIST 选项.

MemoryStick Pro block device driver

CONFIG_MSPRO_BLOCK

"Memory Stick PRO"是 SONY 从 2003 开始引入的升级版标准,早已成为主流,所以应该选"Y"(除非你仍在使用十年前的老 VAIO 笔记本).

{此处省略的 MemoryStick 主控请按照实际使用的芯片进行选择}

LED Support

CONFIG_NEW_LEDS

发光二级管(LED)支持.[提示]标准键盘上的 LED 灯不在此列(由 input 子系统控制)

Accessibility support

CONFIG_ACCESSIBILITY

无障碍(Accessibility)支持.各种帮助残疾人使用计算机的软硬件技术.例如:盲文设备,语音合成,键盘映射,等等.

InfiniBand support

CONFIG_INFINIBAND

InfiniBand 是一种低延迟/高带宽数据中心互联架构,采用远程直接内存存取(RDMA)实现高性能处理器间通信(IPC),同时对虚拟化技术也提供了良好的支持.主要用于服务器集群与高性能计算(HPC)领域.

EDAC (Error Detection And Correction) reporting

CONFIG_EDAC

在电磁环境比较恶劣的情况下,一些大规模集成电路常常会受到干扰,特别是像 RAM 这种利用双稳态进行存储的器件,往往会在强干扰下发生翻转,使原来存储的"0"变为"1",或者"1"变为

"0",造成严重的后果(例如控制程序跑飞,关键数据出错).随着芯片集成度的增加,发生错误的可能性也在增大,这已经成为一个不能忽视的问题.错误检测与纠正(EDAC)技术的目标就是发现并报告甚至纠正在计算机系统中发生的错误,这些错误是由 CPU 或芯片组报告的底层错误(内存错误/缓存错误/PCI 错误/温度过高,等等),建议选"Y".如果这些代码报告了一个错误,请到 http://bluesmoke.sourceforge.net/和 http://buttersideup.com/edacwiki 查看更多信息.详见"Documentation/edac.txt"文档.

EDAC legacy sysfs

CONFIG_EDAC_LEGACY_SYSFS

仅在你需要使用老版本 edac-utils 的情况下才需要选"Y".

Debugging

CONFIG_EDAC_DEBUG

仅供调试使用

Decode MCEs in human-readable form (only on AMD for now)

CONFIG_EDAC_DECODE_MCE

将 MCE(Machine Check Exception)解码为人类可读的形式(目前仅支持 AMD).建议选"Y".

Simple MCE injection interface over /sysfs

CONFIG_EDAC_MCE_INJ

仅供调试使用

Main Memory EDAC (Error Detection And Correction) reporting

CONFIG_EDAC_MM_EDAC

一些系统能够检测和修正主内存中的错误,EDAC 能够报告这些信息(EDAC 自己检测到的或者根据 ECC 得到的).EDAC 还会尽量检测这些错误发生在哪里以便于替换损坏的内存.建议选"Y"并按照你实际硬件状况选取子项

Output ACPI APEI/GHES BIOS detected errors via EDAC

CONFIG_EDAC_GHES

并不是所有机器都提供基于硬件的 EDAC 技术,有部分机器提供的是基于 ACPI BIOS 的报告机制(使用 CONFIG_ACPI_APEI_GHES 驱动).开启此项后,如果检测到 GHES BIOS,那么 CONFIG_ACPI_APEI_GHES 驱动提供的错误报告将会通过 EDAC API 发送到用户空间,同时,硬件 EDAC 也会被禁用,也就是进入"固件优先"模式.注意:GHES BIOS 和硬件 EDAC 两者不能共存,因为 BIOS 和操作系统在读取 error 寄存器时会相互竞争.所以如果你不想使用"固件优先"模式,应该选"N",或者使用"ghes.disable=1"内核引导参数.否则应该选"Y".

{此处省略的芯片请按照实际情况选择}

Real Time Clock

CONFIG_RTC_CLASS

通用 RTC(实时时钟)类支持.所有的 PC 机主板都包含一个电池动力的实时时钟芯片,以便在断电后仍然能够继续保持时间,RTC 通常与 CMOS 集成在一起,因此 BIOS 可以从中读取当前时间(精度一般是秒级).选中此项后你就可以在操作系统中使用一个或多个 RTC 设备(你还必须从下面启用一个或多个 RTC 接口).[注意]Clock 与 Timer 没有任何关系,Timer 是定时器(用于计量时长),Clock 是时钟(用于记录当前的时刻"年-月-日 时:分:秒")

Set system time from RTC on startup and resume

CONFIG_RTC_HCTOSYS

系统启动时从指定的 RTC 设备(CONFIG_RTC_HCTOSYS_DEVICE)中读取时间,以设定系统时间,这将有助于避免时间不准导致的麻烦(例如不必要的文件系统检测(fsck)以及网络故障).建议选"Y".

Set the RTC time based on NTP synchronization

CONFIG_RTC_SYSTOHC

如果用户空间报告了"NTP 已同步",那么每隔大约 11 分钟,内核将会自动把系统时间写入指定的 RTC 设备(CONFIG_RTC_HCTOSYS_DEVICE)中.建议选"Y".

RTC used to set the system time

CONFIG_RTC_HCTOSYS_DEVICE

默认的 RTC 设备(通常是"rtc0").该设备的驱动必须静态编译进内核(而不能作为模块加载).

RTC debug support

CONFIG_RTC_DEBUG

仅供调试使用

/sys/class/rtc/rtcN (sysfs)

CONFIG_RTC_INTF_SYSFS

允许通过 sysfs 接口使用 RTC,允许多个 RTC 设备,也就是/sys/class/rtc/rtc0~N

/proc/driver/rtc (procfs for rtcN)

CONFIG_RTC_INTF_PROC

允许通过 proc 接口使用 RTC,仅允许一个 RTC 设备,也就是/proc/driver/rtc(若有多个 RTC 设备,则其将对应"rtc0")

/dev/rtcN (character devices)

CONFIG_RTC_INTF_DEV

允许通过 dev 接口使用 RTC,允许多个 RTC 设备,也就是/dev/rtc0~N,某些程序(比如 hwclock)需要使用/dev/rtc(这是个软连接,udev 会自动将其指向默认的 RTC 设备)

RTC UIE emulation on dev interface

CONFIG_RTC_INTF_DEV_UIE_EMUL

如果底层 rtc 芯片驱动没有提供 RTC_UIE 就仿真一个 RTC_UIE.选"N",除非你确实知道自己在做什么.

Test driver/device

CONFIG_RTC_DRV_TEST

仅供调试使用

{此处省略的其他 RTC 设备一般仅用于非 PC 环境}

PC-style 'CMOS'

CONFIG_RTC_DRV_CMOS

这是所有 PC 和基于 ACPI 的系统通用的 RTC 驱动.必须选"Y"(不能选"M"),除非你是嵌入式系统.

DMA Engine support

CONFIG_DMADEVICES

DMA 引擎(DMA Engine)可以看做是传统 DMA 控制器(DMA controller)的新生.在 DMA 引擎的协助下,CPU 只需初始化一个传输动作,其余的动作就可以由 DMA 引擎独立完成(完成后以中断的方式通知 CPU),这对于高速传输大量数据以及"分散-收集"操作大有益处,可以节约大量的 CPU 资源(有时也可节约大量的内存操作).目前,DMA 引擎有两个用途:(1)卸载高速网络栈中

的内存 COPY 操作,(2)加速 CONFIG_MD_RAID456 驱动中的 RAID 操作."DMA 引擎"只是一个统称,在不同场合对应着不同的技术,例如 Intel I/OAT(PC 平台)和 AHB(嵌入式).[提示]历史上,ISA 架构的电脑都有一个专用的"DMA 控制器"(最常见的是 Intel 8237),但是到了 PCI 架构,由于每一个 PCI 设备都可以控制 PCI 总线(成为"bus master")并直接读写系统内存,所以虽然 DMA 的操作方式依然存在,但是"DMA 控制器"却消失了.现在新生的"DMA 引擎"目前仍然主要出现在 Intel 的高端芯片上.

DMA Engine debugging

CONFIG_DMADEVICES_DEBUG

仅供调试使用

Intel MID DMA support for Peripheral DMA controllers

CONFIG_INTEL_MID_DMAC

Intel MID DMA 引擎,搭配 Atom 处理器使用.

Intel integrated DMA 64-bit support

CONFIG_INTEL_IDMA64

为 Intel 低功耗子系统(Intel Low Power Subsystem)中整合的 64 位 DMA 提供支持.Intel Skylake PCH 就是一个支持 Intel 低功耗子系统的例子.

Intel I/OAT DMA support

CONFIG_INTEL_IOATDMA

带有英特尔 I/O 加速技术(I/O Acceleration Technology)的至强芯片组 DMA 引擎

Synopsys DesignWare AHB DMA support

CONFIG_DW_DMAC

基于 Synopsys DesignWare IP 核的 AHB 总线 DMA 引擎.例如 Atmel AT32ap7000 中就整合了这个引擎.

Timberdale FPGA DMA support

CONFIG_TIMB_DMA

Timberdale FPGA DMA 引擎."Timberdale FPGA"是一个多功能设备,出现在基于 Intel Atom 的车载信息娱乐系统 IVI(In-Vehicle Infotainment)上.

Intel EG20T PCH / LAPIS Semicon

IOH(ML7213/ML7223/ML7831) DMA

CONFIG_PCH_DMA

所有与 Intel EG20T PCH 兼容的芯片的 DMA 引擎支持,具体型号可以查看内核帮助.都是嵌入式芯片.

Network: TCP receive copy offload

CONFIG_NET_DMA

通过在网络栈中利用 DMA 引擎来减少接收数据包时的 copy-to-user 操作以释放 CPU 资源,仅在 CONFIG_INTEL_IOATDMA 开启的前提下才有意义.

Async_tx: Offload support for the async_tx api

CONFIG_ASYNC_TX_DMA

如果你开启了 CONFIG_MD_RAID456,同时你的硬件又支持 DMA 引擎,那么开启此项后可以加速 RAID 操作.

DMA Test client

CONFIG_DMATEST

仅供调试使用

Auxiliary Display support

CONFIG_AUXDISPLAY

辅助显示设备.例如基于 KS0108 控制器的 Crystalfontz CFAG12864B 单色液晶屏(分辨率:128x64).仅用于嵌入式系统.

Userspace I/O drivers

CONFIG_UIO

UIO(Userspace I/O)是运行在用户空间的 I/O 技术,它为开发用户空间的驱动提供了一个简单的架构(/dev/uioN).使用 uio 的设备一般都属于嵌入式系统.不确定的选"N".[提示]lsuio 工具可以列出所有 UIO 的模块和其映射的内存地址.

VFIO Non-Privileged userspace driver framework

CONFIG_VFIO

VFIO 是一套无特权用户空间 I/O 驱动框架,需要有 IOMMU 虚拟化硬件支持(AMD-Vi/Intel VT-d).此选项仅用于宿主机内核.VFIO 的目标是在 IOMMU 硬件的帮助下,取代 CONFIG_UIO 和 CONFIG_KVM_DEVICE_ASSIGNMENT.VFIO 主要用于编写高效的用户态驱动,以及在虚拟化环境的属主机中高效的实现设备直通(passthrough)且无须 root 特权,可用于详见"Documentation/vfio.txt"文档.[提示]QEMU 1.3 以上版本才能利用 VFIO 特性.不玩 KVM 虚拟化的选"N".

VFIO support for PCI devices

CONFIG_VFIO_PCI

允许 PCI 设备使用 VFIO 框架.这是 VFIO 框架当前的主要用途.选"Y".

VFIO PCI support for VGA devices

CONFIG_VFIO_PCI_VGA

让 VFIO PCI 支持 VGA 扩展,此选项依赖于 CONFIG_VGA_ARB,建议选"Y".

Virtualization drivers

CONFIG_VIRT_DRIVERS

这个选项仅对 PowerPC 架构有意义

Virtio drivers

仅可用于客户机内核的 Virtio 驱动.Virtio 的目标是为各种半虚拟化的虚拟机管理程序(特别是 KVM)提供一组通用的模拟设备,目前已实现:network/block/balloon/console/hw_random,未来还会实现更多.下列驱动仅可用在基于 lguest 或 QEMU 的半虚拟化客户机中(一般是 KVM 或 XEN).

PCI driver for virtio devices

CONFIG_VIRTIO_PCI

半虚拟化 PCI 设备驱动.VMM(虚拟机管理程序)必须要有相应的 "PCI virtio backend".基于 QEMU 的 VMM(KVM,Xen)一般都支持该驱动.[提示]由于目前的 ABI 尚不稳定,建议使用时注意版本匹配.

Virtio balloon driver

CONFIG_VIRTIO_BALLOON

balloon 驱动支持按需扩展或减少 KVM 客户机内的内存大小.

Platform bus driver for memory mapped virtio devices

CONFIG_VIRTIO_MMIO

支持使用内存映射机制的 virtio 设备驱动

Memory mapped virtio devices parameter parsing

CONFIG_VIRTIO_MMIO_CMDLINE_DEVICES

允许通过"virtio_mmio.device"内核模块参数实例化 virtio-mmio 设备.注意,错误的参数(特别是"baseaddr"错误)会导致系统崩溃.详见"Documentation/kernel-parameters.txt"文档中对"virtio_mmio.device"的说明.

Microsoft Hyper-V guest support

仅在将此 Linux 内核作为微软 Hyper-V 虚拟机的来宾操作系统运行时,才需要开启这里的选项.

Microsoft Hyper-V client drivers

CONFIG_HYPERV

将 Linux 内核作为 Hyper-V 的来宾操作系统运行

Microsoft Hyper-V Utilities driver

CONFIG_HYPERV_UTILS

Hyper-V 管理工具驱动.

Microsoft Hyper-V Balloon driver

CONFIG_HYPERV_BALLOON

Hyper-V Balloon 驱动

Xen driver support

仅在将此 Linux 内核作为半虚拟化模式的 XEN 虚拟机的来宾操作系统运行时,才需要开启这里的选项.由于 KVM 的高歌猛进,特

别是 Ubuntu 和 Redhat 的力挺,与昔日的辉煌相比,XEN 现在已经没落许多了.

Staging drivers

CONFIG_STAGING

尚在开发中或尚未完成的,目前尚不完善的驱动,切勿用于生产环境.仅供测试人员或者开发者试用.

X86 Platform Specific Device Drivers

CONFIG_X86_PLATFORM_DEVICES

特定于 X86 平台的设备驱动.例如很多笔记本厂商的专有硬件和特色功能.大多数笔记本用户都应该进去看看(简单易懂,一看即知).

{此处省略所有特定于笔记本厂商的驱动}

Thermal Management driver for Intel menlow platform

CONFIG_INTEL_MENLOW

此驱动专用于 Intel Menlow 平台(搭配 Atom 处理器),提供了增强的 ACPI 热量管理能力.

WMI

CONFIG_ACPI_WMI

ACPI-WMI(Windows 管理规范)映射设备(find /sys/devices/ -name "PNP0C14*")支持.WMI 是微软对 ACPI 规范的专有扩展,

可将部分 ACPI 固件内容通过 PNP0C14 设备映射到用户空间,以方便用户空间调用 ACPI 固件的功能.本选项仅为那些依赖于 WMI 的驱动提供支持(并不真正导出到用户空间),例如 CONFIG_DRM_NOUVEAU 驱动以及专用于 Acer/Asus/Dell/MSI/HP 等品牌笔记本的 WMI 驱动.

Intel Intelligent Power Sharing

CONFIG_INTEL_IPS

Intel Calpella 平台支持 Intel 的智能电源共享(Intelligent Power Sharing)技术,可以在保持功耗不变的前提下,在 CPU 和 GPU 之间智能分配电力.开启此项和 CONFIG_CPU_FREQ 以及 CONFIG_DRM_I915 之后,即可实现此功能.

WMI support for MXM Laptop Graphics

CONFIG_MXM_WMI

MXM 接口笔记本显卡的 WMI 支持.目前主要用于 nvidia 显卡.

Intel Oaktrail Platform Extras

CONFIG_INTEL_OAKTRAIL

Intel Oak Trail 嵌入式平台需要此驱动来开关 WiFi/相机/蓝牙等设备.

Intel Rapid Start Technology Driver

CONFIG_INTEL_RST

英特尔快速启动技术(Intel Rapid Start Technology)通过在 SSD 上建立与本机内存容量一致的休眠分区,在休眠时将内存数据保存到 SSD 休眠分区上,从而实现快速唤醒.

Intel Smart Connect disabling driver

CONFIG_INTEL_SMARTCONNECT

禁用英特尔智能连接技术(Intel Smart Connect Technology)的驱动.此技术旨在通过定期将处于睡眠/待机状态的 PC 唤醒一小段时间,联网一段时间让应用程序(如电子邮件)更新,但是不打开屏幕,然后又自动回到睡眠状态,以此循环重复.这意味着,当您唤醒 PC 时,程序数据已经处于最新状态.主要针对超级本用户且需要 BIOS 支持,必须在 BIOS 中将 Intel Smart Connect Technology 设为"Enabled".如果你的超级本支持此技术且同时安装了 Windows 系统,那么务必选"Y".

pvpanic device support

CONFIG_PVPANIC

pvpanic 是 QEMU 提供的一种准虚拟化(paravirtualized)设备,允许客户机向宿主机报告 panic 事件.此驱动仅可用于客户机内核.

Intel PMC IPC Driver

CONFIG_INTEL_PMC_IPC

为某些 Intel 平台的 PMC 控制器提供支持.这里的"PMC"是一个 RISC 架构的 32 位 ARC 处理器(Argonaut RISC Core processor).不确定的选"N".

Platform support for Chrome hardware

CONFIG_CHROME_PLATFORMS

专用于 Google 公司的 Chromebook 笔记本/Chromebox 迷你机的设备驱动.

Common Clock Framework

CCF(Common Clock Framework)是从 3.4 内核开始引入的新时钟框架,用于取代原有的"Clock Framework".详见"Documentation/clk.txt"文档.这里还有一个讲解 CCF 的 PDF.

DebugFS representation of clock tree

CONFIG_COMMON_CLK_DEBUG

仅供调试使用

{此处省略的时钟设备请按实际情况选择(主要用于嵌入式设备)}

Hardware Spinlock drivers

硬件自旋锁驱动.目前仅出现在嵌入式处理器上.自旋锁是保护共享资源的一种锁机制,与互斥锁比较类似,都是为了解决对某项资源的互斥使用.无论是互斥锁,还是自旋锁,在任何时刻,最

多只能有一个持有者.也就是说,在任何时刻最多只能有一个执行单元获得锁.但是两者在调度机制上略有不同,对于互斥锁,如果资源已经被占用,资源申请者只能进入睡眠状态.但是自旋锁不会引起申请者睡眠,如果自旋锁已经被别的执行单元保持,调用者就一直在循环中"忙等"(占用 CPU 但无事可做),直到该自旋锁被释放."自旋"一词就是因此而得名.自旋锁的使用非常方便,但仅适用于需要极短时间锁定的场合(例如 1 毫秒),以避免消耗太多的 CPU 空等时间.

Clock Source drivers

"时钟源"驱动.主要面向嵌入式设备.

Mailbox Hardware Support

CONFIG_MAILBOX

Mailbox 硬件支持.这里的"Mailbox"是一个框架,通过消息队列和中断驱动信号,控制芯片上的多个处理器之间的通信.仅用于嵌入式环境.

IOMMU Hardware Support

CONFIG_IOMMU_SUPPORT

IOMMU 硬件主要出现在带有 I/O 虚拟化技术的硬件上,例如带有 AMD-Vi 或 VT-d 技术的芯片.IOMMU 主要作用:(1)内存地址转换(例如 DMA 地址转换,scatter-gather),(2)中断重映射,(3)对设备读取和写入的进行权限检查.这对于提高虚拟化性能和安全性,以及在 64 位系统上更好的使用 32 位设备,意义重大.[提示]

此选项仅对宿主机有意义,如果此内核要作为来宾操作系统运行,请选"N".

AMD IOMMU support

CONFIG_AMD_IOMMU

AMD IOMMU 硬件支持.一般还需要在 BIOS 中开启相应选项.

Export AMD IOMMU statistics to debugfs

CONFIG_AMD_IOMMU_STATS

仅供调试使用

AMD IOMMU Version 2 driver

CONFIG_AMD_IOMMU_V2

新一代的 AMD IOMMUv2 硬件(支持 PCI PRI 和 PASID 接口)支持.首次出现在 AMD Opteron 4000/6000 系列平台上.

Support for Intel IOMMU using DMA Remapping Devices

CONFIG_INTEL_IOMMU

让 Intel IOMMU 支持 DMA 重映射,这是 IOMMU 的主要用途,只要你的芯片支持 VT-d,就选"Y".

Support for Shared Virtual Memory with Intel IOMMU

CONFIG_INTEL_IOMMU_SVM

共享虚拟内存(SVM, Shared Virtual Memory)允许不同设备(例如 CPU 与 GPU)以 PASID(Process Address Space ID)为各自的身份标识,统一使用 CPU 地址空间访问系统内存的 DMA 资源,也就是内存统一寻址,以提升异构计算性能.传说 Skylake 会支持 SVM.

Enable Intel DMA Remapping Devices by default

CONFIG_INTEL_IOMMU_DEFAULT_ON

默认开启 DMA 重映射支持,相当于设置"intel_iommu=on"内核引导参数.选"Y".

Support for Interrupt Remapping

CONFIG_IRQ_REMAP

支持对 IO-APIC 和 MSI 设备开启中断重映射,这也是 IOMMU 的主要用途,只要你的芯片支持 VT-d,就选"Y".

Remoteproc drivers

现代的 SoC 芯片一般都会以 AMP(非对称多处理器)方式集成多个不同的处理器(例如 OMAP5432 就在单个芯片上集成了 2 个 Cortex-A15 处理器,2 个 Cortex-M4 处理器,1 个 C64x DSP),这样就可在不同的处理器上分别运行多个不同的操作系统实例(例如,在 2 个 Cortex-A9 处理器上以 SMP(对称多处理器)方式运行 Linux,在 2 个 Cortex-M3 和 1 个 C64x 上分别运行不同的实时操作系统).而 Remoteproc 驱动对此种场合下的处理器间通信非常有用.详见"Documentation/remoteproc.txt"和

"Documentation/rpmsg.txt"文档.目前仅对嵌入式系统有意义.不确定的选"N".

Rpmsg drivers

此项和上面的 Remoteproc 紧密相关,目前其下尚无子项可选.

SOC (System On Chip) specific Drivers

专用于单片机的设备驱动.皆为嵌入式系统.

Generic Dynamic Voltage and Frequency Scaling (DVFS) support

CONFIG_PM_DEVFREQ

DVFS(动态电压与频率调整)可以根据系统负载动态调节设备的运行频率和电压(对于同一芯片,频率越高,需要的电压也越高),从而达到节能目的.此选项提供了一个类似 CPUfreq(CONFIG_CPU_FREQ)的通用 DVFS 框架(devfreq).目前 DVFS 技术进在嵌入式设备(例如 Exynos4/Exynos5)上普遍存在.不确定的选"N".

External Connector Class (extcon) support

CONFIG_EXTCON

extcon(外部连接器类)允许用户空间通过 sysfs 和 uevent 监控外部连接器,同时也支持多状态外部连接器(也就是拥有多个连接线缆的外部连接器).例如,一端连接到主机 USB 端口的多状态外部连接器,另一端可以同时连接一条 HDMI 线缆和一个 AC 适

配器.30 针的 PDMI 连接器也是多状态外部连接器的常见例子.不确定的选"N".

Memory Controller drivers

CONFIG_MEMORY

内存控制器驱动.这里所说的"内存控制器"仅指嵌入式 SoC 系统中的各种控制器.不确定的选"N".

Industrial I/O support

CONFIG_IIO

IIO 子系统为各种不同总线(i2c,spi,等)的嵌入式传感器驱动提供了一个统一的框架.例如:(1)模数转换器,(2)加速度传感器,(3)陀螺仪,(4)惯性测量仪,(5)电容-数字转换器,(6)压力/温度/光线传感器,等等.主要用于工业领域和嵌入式领域.不确定的选"N".

Non-Transparent Bridge support

CONFIG_NTB

PCI-E 非透明桥是一个点对点 PCI-E 总线,用于连接两条对等的 PCI-E 总线.例如英特尔 Atom S1200 处理器, Xeon C5500/C3500 嵌入式处理器等.具体支持的设备号(PCI_DEVICE_ID)可以查看"drivers/ntb/ntb_hw.h"文件或 NTB 驱动数据库.不确定的选"N".

VME bridge support

CONFIG_VME_BUS

VME(VersaModule Eurocard)总线是一种通用的计算机总线，主要用于工业控制/军用系统/航空航天/交通运输/医疗等嵌入式领域.而 VME 桥则是其他总线(例如 PCI/PCI-E)到 VME 总线之间的转换芯片.不确定的选"N".

Pulse-Width Modulation (PWM) Support

CONFIG_PWM

PWM(脉宽调制)是将模拟信号转换为脉波的一种技术.在计算机领域,这项技术常被用于控制风扇转速和背光显示器的亮度.很多微型处理器内部都包含有 PWM 控制器,此选项为所有 PWM 控制器驱动提供了一个统一的框架.不确定的选"N".

Intel LPSS PWM support

CONFIG_PWM_LPSS

适用于"Intel Low Power Subsystem" PWM 控制器的通用框架驱动.适用于 Atom Z36xxx/Z37xxx 系列处理器.

NXP PCA9685 PWM driver

CONFIG_PWM_PCA9685

适用于 NXP PCA9685 LED 亮度控制器.

IndustryPack bus support

CONFIG_IPACK_BUS

IndustryPack 是工业控制领域常用的一种总线.不确定的选"N".

Reset Controller Support

CONFIG_RESET_CONTROLLER

为 GPIO 总线或者芯片内置的重启控制器提供通用支持.仅用于嵌入式设备.不确定的选"N".

FMC support

CONFIG_FMC

FMC(FPGA Mezzanine Carrier)是一个定义如何将 FPGA 夹层卡(FPGA Mezzanine Card)连接到主机电路板的接口标准,仅用于嵌入式环境.

PHY Subsystem

PHY 子系统.

PHY Core

CONFIG_GENERIC_PHY

为内核中的所有 PHY 设备提供一个通用的 PHY 框架.不确定的选"N".内核中若有其他部分依赖它,会自动选上.

{此处省略的部分请按照硬件的实际情况选择}

Generic powercap sysfs driver

CONFIG_POWERCAP

"power capping"的意思是允许用户把设备的总功耗限定在指定的范围内.此选项允许内核子系统将"power capping"的设置以 sysfs 的方式导出到用户空间.不确定的选"N".

MCB support

CONFIG_MCB

MCB(MEN Chameleon Bus)是专用于德国 MEN Mikro Elektronik 公司的 FPGA 设备的总线.MEN Mikro Elektronik 公司的嵌入式计算主要是为航空/航海/铁路及陆地车辆的应用,以及自动化/电力/能源和医疗用途.不确定的选"N".

Performance monitor support

专用于 ARM 架构的 CPU 性能监控框架.

Reliability, Availability and Serviceability (RAS) features

CONFIG_RAS

RAS(可靠,可用,可维护)是一个计算机硬件术语,可靠性描述系统能够持续正确工作多长时间,可用性描述系统能够正确工作的时间百分比,可维护性描述系统从错误恢复到正常需要多长时间.具有高等级 RAS 的硬件会有一系列额外的技术保障数据的可靠性与正确性.

Thunderbolt support for Apple devices

CONFIG_THUNDERBOLT

苹果公司的雷电接口支持.

Android Drivers

CONFIG_ANDROID

安卓平台专用驱动

NVDIMM (Non-Volatile Memory Device) Support

CONFIG_LIBNVDIMM

NVDIMM(非易失性内存)支持.

NVMEM Support

CONFIG_NVMEM

NVMEM(非易失性存储器)设备支持.包括:EEPROM,EFUSES...不确定的选"N".

System Trace Module devices

CONFIG_STM

仅供调试使用

Dummy STM driver

CONFIG_STM_DUMMY

仅供调试使用

Kernel console over STM devices

CONFIG_STM_SOURCE_CONSOLE

仅供调试使用

Intel(R) Trace Hub controller

CONFIG_INTEL_TH

仅供调试使用

FPGA Configuration Framework

CONFIG_FPGA

FPGA 配置框架支持.仅用于嵌入式系统.

Firmware Drivers

固件(Firmware)驱动

BIOS Enhanced Disk Drive calls determine boot disk

CONFIG_EDD

这是一个实验性选项,支持实模式 BIOS 中的增强磁盘服务(EDD),从而允许内核从 BIOS 中获取第一启动盘(可以从 sysfs 中查看具体是哪个硬盘),大多数 BIOS 提供商都没有实现这个特性.不确定的选"N".

Sets default behavior for EDD detection to off

CONFIG_EDD_OFF

选"Y"相当于使用"edd=off"内核引导参数(禁用 EDD),选"N"相当于使用"edd=on"内核引导参数(启用 EDD).语法:edd={on|skipmbr|off}.

Add firmware-provided memory map to sysfs

CONFIG_FIRMWARE_MEMMAP

将原始的固件内容映射到"/sys/firmware/memmap"文件.主要用于调试目的,以及 kexec 为下一个内核设置参数.详见"Documentation/ABI/testing/sysfs-firmware-memmap"文档.不确定的选"N".

BIOS update support for DELL systems via sysfs

CONFIG_DELL_RBU

允许 Dell OpenManage 或 DUP(Dell Update Packages)工具通过 sysfs 更新 DELL 服务器主板的 BIOS.详见"Documentation/dell_rbu.txt"文档.即使你确实需要此功能,也建议选"M"而不是"Y".毕竟刷 BIOS 不能当做家常便饭.留着刷 BIOS 的接口,总是件危险的事.

Dell Systems Management Base Driver

CONFIG_DCDBAS

该驱动为 DELL 服务器专用的系统管理软件(Dell OpenManage)提供了 sysfs 接口.详见"Documentation/dcdbas.txt"文档.

Export DMI identification via sysfs to userspace

CONFIG_DMIID

将 SMBIOS(System Management BIOS)/DMI(Desktop Management Interface)中的系统识别信息(序列号,制造商,型号,等等)导出到用户空间(/sys/class/dmi/id/).开启此项后,dmidecode 工具就可以显示与制造商相关的信息.此外,基于

DMI 的模块的自动加载也依赖于此项.建议选"Y",因为某些 Udev 规则和虚拟化检测需要它.

DMI table support in sysfs

CONFIG_DMI_SYSFS

将 SMBIOS/DMI 中的原始数据(包含大量的系统硬件信息)导出到用户空间(/sys/firmware/dmi/).以允许 dmidecode 工具无需访问/dev/mem 设备(需要 root 权限)即可将这些信息显示出来.

iSCSI Boot Firmware Table Attributes

CONFIG_ISCSI_IBFT_FIND

使内核能定位 iBFT(iSCSI Boot Firmware Table)在内存中的位置.目的是为了能够从 iSCSI 驱动器中启动操作系统.不确定的选"N".

iSCSI Boot Firmware Table Attributes module

CONFIG_ISCSI_IBFT

将 iBFT(iSCSI Boot Firmware Table)的内容通过 sysfs 导出到用户空间.如果你想在系统引导过程中动态检测 iSCSI 引导参数,可以选"Y",否则应选"N".

Google Firmware Drivers

CONFIG_GOOGLE_FIRMWARE

仅用于 Google 自家的服务器

EFI (Extensible Firmware Interface) Support

EFI/UEFI(统一可扩展固件接口)支持.2010 年之后,UEFI 已经基本全线取代了 BIOS.

EFI Variable Support via sysfs

CONFIG_EFI_VARS

通过 sysfs 接口操作(读/写/新建/删除)EFI 变量的老旧方式,并且可能与新方式(CONFIG_EFIVAR_FS)冲突.应该选"N".

Register efivars backend for pstore

CONFIG_EFI_VARS_PSTORE

将 efivars 模块(CONFIG_EFI_VARS)用作 pstore 文件系统(CONFIG_PSTORE)的后端.这样就可以向 EFI 变量中写入各种 pstore 支持的信息,例如,控制台消息,崩溃转储,等等.

Disable using efivars as a pstore backend by default

CONFIG_EFI_VARS_PSTORE_DEFAULT_DISABLE

禁止默认将 efivars 模块(CONFIG_EFI_VARS)用作 pstore 文件系统(CONFIG_PSTORE)的后端.

Enable EFI fake memory map

CONFIG_EFI_FAKE_MEMMAP

选"Y"等价于使用"efi_fake_mem"内核引导参数,仅供调试使用.

File systems

文件系统

Second extended fs support

CONFIG_EXT2_FS

Ext2 文件系统,无日志.详见"Documentation/filesystems/ext2.txt"文档.

Ext2 extended attributes

CONFIG_EXT2_FS_XATTR

Ext2 文件系统扩展属性(与 inode 关联的 name:value 对)支持.详见 attr 手册.不确定的选"N".

Ext2 POSIX Access Control Lists

CONFIG_EXT2_FS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见 acl 手册.不确定的选"N".

Ext2 Security Labels

CONFIG_EXT2_FS_SECURITY

"安全标签"允许选择使用不同安全模块(如 SELinux)实现的访问控制模型,如果你没有使用需要扩展属性的安全模块,可以选"N".

Ext2 execute in place support

CONFIG_EXT2_FS_XIP

芯片内执行(execute in place)的意思是程序在写入存储介质时就已经分配好运行时的地址,因此不需要载入内存即可在芯片内执行,一般仅在嵌入式系统上才使用这种技术.

Ext3 journalling file system support

CONFIG_EXT3_FS

Ext3 日志型文件系统.详见"Documentation/filesystems/ext3.txt"文档.

Default to 'data=ordered' in ext3

CONFIG_EXT3_DEFAULTS_TO_ORDERED

选"Y"表示将默认的日志模式设为"data=ordered"(更安全),选"N"表示将默认的日志模式设为"data=writeback"(更危险).选"Y",仅在你确实明白"data=writeback"的风险,以及确实有充足的理由的时候,才能选"N".

Ext3 extended attributes

CONFIG_EXT3_FS_XATTR

Ext3 文件系统扩展属性(与 inode 关联的 name:value 对)支持.详见 attr 手册.不确定的选"N".

Ext3 POSIX Access Control Lists

CONFIG_EXT3_FS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见 acl 手册.不确定的选"N".

Ext3 Security Labels

CONFIG_EXT3_FS_SECURITY

"安全标签"允许选择使用不同安全模块(如 SELinux)实现的访问控制模型,如果你没有使用需要扩展属性的安全模块,可以选"N".

The Extended 4 (ext4) filesystem

CONFIG_EXT4_FS

Ext4 日志型文件系统.详见"Documentation/filesystems/ext4.txt"文档.

Use ext4 for ext2/ext3 file systems

CONFIG_EXT4_USE_FOR_EXT23

在 ext2/ext3 文件系统上使用 ext4 驱动.这样可以对 ext2/ext3/ext4 三种文件系统只使用同一个驱动.主要目的是减少内核尺寸.

Ext4 POSIX Access Control Lists

CONFIG_EXT4_FS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见 acl 手册.不确定的选"N".

Ext4 Security Labels

CONFIG_EXT4_FS_SECURITY

"安全标签"允许选择使用不同安全模块(如 SELinux)实现的访问控制模型,如果你没有使用需要扩展属性的安全模块,可以选"N".

EXT4 debugging support

CONFIG_EXT4_DEBUG

仅供调试使用

JBD (ext3) debugging support

CONFIG_JBD_DEBUG

仅供调试使用

JBD2 (ext4) debugging support

CONFIG_JBD2_DEBUG

仅供调试使用

Reiserfs support

CONFIG_REISERFS_FS

曾经的明星文件系统,特别擅长处理大量小文件的场合,由于其创始人入狱,前景不明.

Enable reiserfs debug mode

CONFIG_REISERFS_CHECK

仅供调试使用

Stats in /proc/fs/reiserfs

CONFIG_REISERFS_PROC_INFO

在/proc/fs/reiserfs 文件中显示 Reiserfs 文件系统的状态，仅供调试使用

ReiserFS extended attributes

CONFIG_REISERFS_FS_XATTR

ReiserFS 文件系统扩展属性(与 inode 关联的 name:value 对)支持.详见 attr 手册.不确定的选"N".

ReiserFS POSIX Access Control Lists

CONFIG_REISERFS_FS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见 acl 手册.不确定的选"N".

ReiserFS Security Labels

CONFIG_REISERFS_FS_SECURITY

"安全标签"允许选择使用不同安全模块(如 SELinux)实现的访问控制模型,如果你没有使用需要扩展属性的安全模块,可以选"N".

JFS filesystem support

CONFIG_JFS_FS

JFS 日志型文件系统.详见

"Documentation/filesystems/jfs.txt"文档.

JFS POSIX Access Control Lists

CONFIG_JFS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越

"owner/group/world"的权限管理方式,可以更精细的针对每个

用户进行访问控制.详见 acl 手册.不确定的选"N".

JFS Security Labels

CONFIG_JFS_SECURITY

"安全标签"允许选择使用不同安全模块(如 SELinux)实现的访

问控制模型,如果你没有使用需要扩展属性的安全模块,可以选

"N".

JFS debugging

CONFIG_JFS_DEBUG

仅供调试使用

JFS statistics

CONFIG_JFS_STATISTICS

在/proc/fs/jfs/目录中显示 JFS 文件系统的统计信息

XFS filesystem support

CONFIG_XFS_FS

XFS日志型文件系统是一个高性能的文件系统(笔者的最爱),擅长大文件和多线程.详见"Documentation/filesystems/"目录中"xfs*.txt"系列文档.

XFS Quota support

CONFIG_XFS_QUOTA

XFS磁盘配额(使用专用的xfs_quota工具)比通用磁盘配额模块(CONFIG_QUOTA)拥有更高级的特性,它不但能够控制用户或组的磁盘用量,还能控制项目(文件夹)的磁盘用量(无论哪个用户在项目的文件夹中创建文件),但是不能同时使用组配额和项目配额.此外,对XFS来说,配额数据记录在文件系统元数据中,而不是像CONFIG_QUOTA那样记录在aquota.user和aquota.group文件中.最后,XFS配额和通用磁盘配额是两个相互独立的系统,可以同时并存.

XFS POSIX ACL support

CONFIG_XFS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见acl手册.不确定的选"N".

XFS Realtime subvolume support

CONFIG_XFS_RT

"实时子卷"是专门存储文件数据的卷,可以允许将日志与数据分开在不同的磁盘上,例如将大块头的流媒体文件存储在高速磁盘组成的实时子卷上.详见 xfs 手册页.

XFS Verbose Warnings

CONFIG_XFS_WARN

仅供调试使用

XFS Debugging support

CONFIG_XFS_DEBUG

仅供调试使用

GFS2 file system support

CONFIG_GFS2_FS

GFS2 可用于搭建高可用集群文件系统,由红帽公司开发,允许所有集群节点并行访问,同时又能够完美的保持文件系统的一致性:一个节点对文件系统的任何修改都立即对所有其他节点可见.详见"Documentation/filesystems/"目录中"gfs*.txt"系列文档.

GFS2 DLM locking

CONFIG_GFS2_FS_LOCKING_DLM

GFS2 分布式锁管理器(DLM).务必选"Y",除非你知道自己在做什么.

OCFS2 file system support

CONFIG_OCFS2_FS

OCFS2(Oracle 集群文件系统)的目标是成为一种通用文件系统.OCFS2 能使集群中的所有节点并发的通过标准文件系统接口来访问存储备.要使用 OCFS2 还需要 ocfs2-tools 的帮助.详见"Documentation/filesystems/ocfs2.txt"文档.

O2CB Kernelspace Clustering

CONFIG_OCFS2_FS_O2CB

O2CB(OCFS2 Cluster Base)是位于内核空间的集群服务结构.具体包括:NM(节点管理器,监控所有节点),HB(心跳服务),TCP(控制节点间的通讯),DLM(分布式锁管理器),CONFIGFS(用户配置文件系统驱动,挂载点是/config),DLMFS(用户空间和内核空间 DLM 的接口).开启此项后,将只需要很少量的用户空间组件(也就是 ocfs2-tools),OCFS2 就可以转起来了.但它只能玩转 OCFS2 自身,玩不了其他集群.

OCFS2 Userspace Clustering

CONFIG_OCFS2_FS_USERSPACE_CLUSTER

为用户空间的集群服务提供支持.目的是为了配合 CONFIG_DLM 模块一起使用.

OCFS2 statistics

CONFIG_OCFS2_FS_STATS

允许对 OCFS2 的使用状况进行一些统计.开启后会增加内存占用.

OCFS2 logging support

CONFIG_OCFS2_DEBUG_MASKLOG

仅供调试使用

OCFS2 expensive checks

CONFIG_OCFS2_DEBUG_FS

以性能为代价提供了存储一致性检测,仅供调试使用

Btrfs filesystem support

CONFIG_BTRFS_FS

Btrfs 是由 Oracle 于 2007 年宣布的支持写时复制(COW)的文件系统.拥有众多抢眼球的特性:软 RAID 管理,卷管理,克隆/快照,压缩功能,支持跨多块磁盘动态增大或收缩卷.其目标是成为下一代 Linux 标准文件系统.详见"Documentation/filesystems/btrfs.txt"文档.

Btrfs POSIX Access Control Lists

CONFIG_BTRFS_FS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个用户进行访问控制.详见 acl 手册.不确定的选"N".

Btrfs with integrity check tool compiled in (DANGEROUS)

CONFIG_BTRFS_FS_CHECK_INTEGRITY

仅供调试使用

Btrfs will run sanity tests upon loading

CONFIG_BTRFS_FS_RUN_SANITY_TESTS

仅供调试使用

Btrfs debugging support

仅供调试使用

NILFS2 file system support

CONFIG_NILFS2_FS

NILFS2 是一种非常前卫的"log-structured"文件系统,是 Linux 下一代文件系统的有力竞争者.NILFS2 将底层设备当作一种只能追加写(append)的设备,文件系统的任何修改都只以顺序追加的方式写入磁盘(而是不覆盖旧数据),从而避免耗时的寻道(seek)操作,从而大幅提升写入性能(因为文件系统的整体效率主要由写操作的效率决定).此种思路带来了一系列靓瞎眼的特性:自动不间断快照(可以迅速恢复被删除的文件或者回到先前某个特定的时间点),快速崩溃恢复(比大多数日志型文件系统还要快),高性能(在 SSD 上更有绝对优势).但也带来了一个新问题:需要垃圾收集机制以清理旧数据,造成垃圾收集时的性能降低(可以通过合理安排垃圾收集时间来避免).NILFS2 目前不支

持如下功能:atime(访问时间),POSIX ACL,扩展属性.不过考虑到 SSD(固态硬盘)即将成为主流,假以时日,前途大大的啊!.详见"Documentation/filesystems/nilfs2.txt"文档.

F2FS filesystem support

CONFIG_F2FS_FS

F2FS 也是一种非常前卫的"log-structured"文件系统,针对基于 NAND 闪存的存储设备进行了特别设计,使之更加适应新的存储介质(也就是闪存),据称它是目前 SSD 上性能最好的文件系统.

Direct Access (DAX) support

CONFIG_FS_DAX

DAX(Direct Access)的含义是绕过内存缓冲直接访问块设备.选"Y"后,就可以用"mount -o dax"方式挂载块设备以避免使用 pagecache 作为 I/O 缓冲.例如将持续性非易失内存(断电内容不丢失)作磁盘用时,或者挂载内存盘时,使用此特性便恰到好处.还有一种用法是在虚拟机内使用"mount -o dax"方式挂载已经缓存在宿主机内存中的块设备文件(或者倒过来也可以).

Enable POSIX file locking API

CONFIG_FILE_LOCKING

POSIX 标准文件锁定 API 支持.NFS 之类的网络文件系统和给文件加锁与解锁的 flock()系统调用需要它.不确定的选"Y".

Dnotify support

CONFIG_DNOTIFY

旧式的基于目录的文件变化的通知机制(已被 Inotify 取代),目前仅有 NFSv4 以及少量古董程序依赖它.不确定的选"N".

Inotify support for userspace

CONFIG_INOTIFY_USER

用户空间的 Inotify 支持.Inotify 是替代 Dnotify 的文件系统变化通知机制.建议选"Y".[提示]如果你使用了 systemd 作为 init,那就必须选"Y".

Filesystem wide access notification

CONFIG_FANOTIFY

fanotify 是一种打算取代 Inotify 的文件系统变化通知机制,不过,由于目前 Fanotify 比 inotify 支持的文件系统事件类型少很多,完全取代 Inotify 还不现实.Udev 的"Predictable Network Interface Names"功能依赖于它.建议选"Y".[提示]如果你使用了 systemd 作为 init,那就必须选"Y".

fanotify permissions checking

CONFIG_FANOTIFY_ACCESS_PERMISSIONS

允许 fanotify 的监听器(listener)对文件系统事件进行权限检查.这样,监听器就可以在系统访问某个文件之前,首先扫描此文件.某些防病毒程序以及分级存储管理系统可能需要此特性.不确定的选"N".

Quota support

CONFIG_QUOTA

通用的磁盘配额支持(限制某个用户或者某组用户的磁盘占用空间).需要配合 quota-tools 工具使用.

Report quota messages through netlink interface

CONFIG_QUOTA_NETLINK_INTERFACE

通过 netlink 接口报告 QUOTA 的警告信息(例如"到达限额").不确定的选"Y".

Print quota warnings to console (OBSOLETE)

CONFIG_PRINT_QUOTA_WARNING

将 QUOTA 的警告信息直接显示在控制台上.反对使用,未来会移除此项.选"N".

Additional quota sanity checks

CONFIG_QUOTA_DEBUG

对 quota 内部结构进行额外的完整性检查.主要用于调试目的.不确定的选"N".

Old quota format support

CONFIG_QFMT_V1

老旧的 v1 版配额格式(linux-2.4.22 之前使用的格式)支持.选"N".

Quota format vfsv0 and vfsv1 support

CONFIG_QFMT_V2

vfsv0/vfsv1 配额格式支持.两者都支持 32 位的 UID/GID,而 vfsv1 还支持 64 位的 inode/block 配额.建议开启.

Kernel automounter version 4 support (also supports v3)

CONFIG_AUTOFS4_FS

第四版内核按需自动挂载文件系统的支持(也支持 v3).此特性需要配合用户空间工具(autofs 或 Systemd)使用,以实现仅在某个文件系统挂载点真正被访问到的时候才触发挂载操作.[提示]使用了 systemd 的系统建议选"Y".

FUSE (Filesystem in Userspace) support

CONFIG_FUSE_FS

FUSE 允许在用户空间实现一个全功能的文件系统,还有一个与之对应的 libfuse2 库和相关工具.详见"Documentation/filesystems/fuse.txt"文档.如果你打算开发一个自己的文件系统或者使用一个基于 FUSE 的文件系统(例如 NTFS-3G 或 ZFS-FUSE 或 GlusterFS),可以选"Y".

Character device in Userspace support

CONFIG_CUSE

这是一个 FUSE 扩展,用于在用户空间实现字符设备支持.

Overlay filesystem support

CONFIG_OVERLAY_FS

overlay 的含意是以层叠的方式组合上下两个文件系统层,常和容器技术配合使用.详见 Documentation/filesystems/overlayfs.txt 文档.

Caches

文件系统缓存

General filesystem local caching manager

CONFIG_FSCACHE

通用文件系统本地缓存管理器.它为各种不同的文件系统(例如网络文件系统)提供了统一的本地缓存框架.这样各种缓存实现可以作为插件添加进来.详见 "Documentation/filesystems/caching/fscache.txt"文档.

Gather statistical information on local caching

CONFIG_FSCACHE_STATS

收集本地缓存的统计信息(这会增加系统运行负载),并通过 /proc/fs/fscache/stats 文件导出到用户空间.主要用于调试目的.

Gather latency information on local caching

CONFIG_FSCACHE_HISTOGRAM

收集本地缓存的延迟信息(这会增加系统运行负载),并通过/proc/fs/fscache/histogram 文件导出到用户空间.主要用于调试目的.

Debug FS-Cache

CONFIG_FSCACHE_DEBUG

仅供调试使用

Maintain global object list for debugging purposes

CONFIG_FSCACHE_OBJECT_LIST

在/proc/fs/fscache/objects 文件中维护一个活动缓存对象的全局列表.仅用于调试目的.

Filesystem caching on files

CONFIG_CACHEFILES

将一个已挂载的文件系统用作另一个文件系统的缓存.例如将一个本地磁盘分区挂载为一个远程网络文件系统的缓存,或者将一个高速设备(例如 SSD)用作一个低速设备(例如普通硬盘)的缓存.

Debug CacheFiles

CONFIG_CACHEFILES_DEBUG

仅供调试使用

Gather latency information on CacheFiles

CONFIG_CACHEFILES_HISTOGRAM

收集本地缓存的延迟信息(这会增加系统运行负载),并通过 /proc/fs/cachefiles/histogram 文件导出到用户空间.主要用于调试目的.

CD-ROM/DVD Filesystems

CD-ROM/DVD 光盘文件系统

ISO 9660 CDROM file system support

CONFIG_ISO9660_FS

ISO9660 是所有 CD/DVD 光盘通用的标准文件系统.建议选"Y".

详见"Documentation/filesystems/isofs.txt"文档.

Microsoft Joliet CDROM extensions

CONFIG_JOLIET

Microsoft 对 ISO9660 文件系统的 Joliet 扩展,允许在文件名中使用 Unicode 字符,也允许长文件名.建议选"Y".

Transparent decompression extension

CONFIG_ZISOFS

Linux 对 ISO9660 文件系统的扩展,允许将数据透明的压缩存储在 CD 上.使用并不广泛,不确定的可以选"N".

UDF file system support

CONFIG_UDF_FS

UDF 被设计为可擦写格式(其实质是管理增量写入),其目标是取代 ISO9660,现已经广泛地用于大容量 DVD 光盘上(特别是刻录盘).建议选"Y".详见"Documentation/filesystems/udf.txt"文档.

DOS/FAT/NT Filesystems

DOS/FAT/NTFS 文件系统

MSDOS fs support

CONFIG_MSDOS_FS

古老的 MSDOS 文件系统(FAT16),基本绝种了

VFAT (Windows-95) fs support

CONFIG_VFAT_FS

从 Win95 开始使用的 VFAT 文件系统(FAT32).如果你要使用基于 UEFI 平台的电脑,并且使用 GPT 磁盘分区,则必须选"Y".详见"Documentation/filesystems/vfat.txt"文档.

Default codepage for FAT

CONFIG_FAT_DEFAULT_CODEPAGE

在 FAT 系列文件系统上,"8.3"格式的短文件名以特定的代码页进行存储(可以通过 chcp 命令查看),但长文件名却以 Unicode 进行存储.此选项的作用就是指定将长文件名转换为短文件名时使用的默认代码页.可以通过"codepage"挂载选项进行修改.简体中文通常使用"936",繁体中文通常使用"950".

Default iocharset for FAT

CONFIG_FAT_DEFAULT_IOCHARSET

指定默认以什么字符集显示文件名,这个值可以通过"iocharset"挂载选项修改.但必须与系统的 locale 设置保持一致.例如在"zh_CN.UTF-8"或"en_US.UTF-8"的情况下应该使用"utf8".[注意]应谨慎使用"iocharset=utf8",因为它会导致 FAT 文件系统上的文件名变得大小写敏感.

exFAT fs support

CONFIG_EXFAT_FS

exfat-nofuse 开源项目提供的内核级 exFAT,FAT12,FAT16,FAT32(vfat)文件系统支持补丁.此补丁是取代前面两个内核自带的 FAT 驱动的更优秀替代品.

enable discard support

CONFIG_EXFAT_DISCARD

"discard"挂载选项支持.在 U 盘或者 SSD 上使用此文件系统的用户必选"Y".

enable delayed sync

CONFIG_EXFAT_DELAYED_SYNC

延迟刷写磁盘脏数据,可提高文件系统性能.有电池的笔记本或者有 UPS 的台式机建议选"Y".

enable kernel debug features via ioctl

CONFIG_EXFAT_KERNEL_DEBUG

仅供调试使用.

print debug messages

CONFIG_EXFAT_DEBUG_MSG

仅供调试使用

Default codepage for exFAT

CONFIG_EXFAT_DEFAULT_CODEPAGE

在 FAT 系列文件系统上,"8.3"格式的短文件名以特定的代码页进行存储(可以通过 chcp 命令查看),但长文件名却以 Unicode 进行存储.此选项的作用就是指定将长文件名转换为短文件名时使用的默认代码页.可以通过"codepage"挂载选项进行修改.简体中文通常使用"936",繁体中文通常使用"950".

Default iocharset for exFAT

CONFIG_EXFAT_DEFAULT_IOCHARSET

指定默认以什么字符集显示文件名,这个值可以通过"iocharset"挂载选项修改.但必须与系统的 locale 设置保持一致.例如在"zh_CN.UTF-8"或"en_US.UTF-8"的情况下应该使用"utf8".[注意]应谨慎使用"iocharset=utf8",因为它会导致 FAT 文件系统上的文件名变得大小写敏感.

NTFS file system support

CONFIG_NTFS_FS

NTFS 文件系统.仅选中此项表示仅支持只读(不支持 NTFS 压缩或加密文件),不支持写入.详见"Documentation/filesystems/ntfs.txt"文档.

NTFS debugging support

CONFIG_NTFS_DEBUG

仅供调试使用

NTFS write support

CONFIG_NTFS_RW

由于微软没有公开 NTFS 的技术标准,所以内核只能支持非常残缺的写入功能:仅能覆盖已存在的文件但不能改变其长度,不能创建文件或目录.建议选"N".[提示]在 Linux 环境下写入 NTFS 始终是一件危险的事情,即使对于 NTFS-3G 也是如此,除了不支持压缩或加密文件之外,网上还有不少血的教训,有兴趣可以搜搜.

Pseudo filesystems

伪文件系统

/proc file system support

CONFIG_PROC_FS

显示系统状态的虚拟文件系统(进程信息,irq 设置,内存使用,设备驱动,网络状态等),通常挂载到"/proc"目录.许多程序依赖于它,systemd 也依赖于它.选"Y",除非你知道自己在做什么.详见"Documentation/filesystems/proc.txt"文档.

/proc/kcore support

CONFIG_PROC_KCORE

系统物理内存的映象.建议选"N".

/proc/vmcore support

CONFIG_PROC_VMCORE

以 ELF 格式转储的已崩溃内核镜像,仅供调试使用

Sysctl support (/proc/sys)

CONFIG_PROC_SYSCTL

显示各种不同的内核调节参数,并让 root 用户能通过/proc/sys/目录交互地更改其中的某些内容.必选"Y",除非你是嵌入式系统并且知道自己在做什么.详见"Documentation/sysctl/"目录中的文档.

Enable /proc page monitoring

CONFIG_PROC_PAGE_MONITOR

用于监视进程内存占用的接口(/proc/<pid>/{smaps,clear_refs,pagemap}和/proc/{kpagecount,kpageflags}).建议选"Y".

sysfs file system support

CONFIG_SYSFS

导出内核内部对象及其属性和对象之间的相互关系的文件系统,通常挂载到"/sys"目录,sysfs 把连接在系统上的设备和总线以及驱动程序等组织成为一个分级的文件,并允许通过该文件系统调整某些内核子系统以及设备的参数.内核启动时依靠它挂载类似"/dev/sda1"这样形式的根分区,禁用 sysfs 后必须在内核引导参数中使用设备号指定根分区(类似"root=03:01"这样).systemd 依赖于它.选"Y",除非你知道自己在做什么.详见"Documentation/filesystems/"目录中"sysfs*.txt"系列文档.

Tmpfs virtual memory file system support

(former shm fs)

CONFIG_TMPFS

tmpfs 文件系统(以前叫 shm[共享内存]文件系统),大多数系统的正常运行都依赖于它(例如 Udev 使用的"/dev/"目录通常挂载为 tmpfs).选"Y",除非你知道自己在做什么.详见"Documentation/filesystems/tmpfs.txt"文档.

Tmpfs POSIX Access Control Lists

CONFIG_TMPFS_POSIX_ACL

POSIX ACL(访问控制列表)支持,这是一种超越"owner/group/world"的权限管理方式,可以更精细的针对每个

用户进行访问控制.详见 acl 手册.许多发行版都要求/dev/目录支持 ACL(例如让 ALSA 相关的文件可以正常工作),并且 systemd 也建议开启它.不确定的选"Y".

Tmpfs extended attributes

CONFIG_TMPFS_XATTR

TMPFS 文件系统扩展属性(与 inode 关联的 name:value 对)支持(仅支持 trusted.*和 security.*命名空间).详见 attr 手册.由于它被 CONFIG_TMPFS_POSIX_ACL 依赖,并且 systemd 也建议开启它.建议选"Y".

HugeTLB file system support

CONFIG_HUGETLBFS

这是使用大内存页的传统方式,需要专门进行配置以及应用程序的特别支持.推荐使用较新的透明大内存页(CONFIG_TRANSPARENT_HUGEPAGE).选"N".

Userspace-driven configuration filesystem

CONFIG_CONFIGFS_FS

configfs 是一个基于内存的虚拟文件系统,与 sysfs 类似但又有不同:configfs 用于从用户空间查看/修改/创建/删除内核对象,而 sysfs 仅能查看/修改由内核负责创建和删除的对象.通常挂载到"/config"目录.详见"Documentation/configfs/"目录中的文档.不确定的选"N".

Miscellaneous filesystems

CONFIG_MISC_FILESYSTEMS

各种非主流的杂项文件系统,有些是专用于嵌入式系统,有些是来自于其他操作系统,还有些专用于某些特定场合.

{此处省略哪些非常非主流的文件系统}

eCrypt filesystem layer support

CONFIG_ECRYPT_FS

eCryptfs 是一个符合 POSIX 标准的企业级文件系统加密栈(加密/解密转换层),工作在 VFS(虚拟文件系统)层,可以在各种普通文件系统上使用(需要 ecryptfs-utils 工具).eCryptfs 将加密元数据保存在每个文件的首部,从而允许文件在不同主机之间任意移动,同时又能确保仅在内核密钥环中拥有正确密钥的时候才能解密文件的内容.此外,eCryptfs 还支持高级密匙管理和配置策略.[提示]使用 eCryptfs 之后,读操作性能最大可下降 1/3 左右,写操作性能则普遍下降一个数量级.

Enable notifications for userspace key wrap/unwrap

CONFIG_ECRYPT_FS_MESSAGING

允许 ecryptfsd 守护进程操作/dev/ecryptfs 设备.这将允许用户空间使用其他后端(例如 OpenSSL)加密/解密 FEK(file encryption key).不确定的选"N".

SquashFS 4.0 - Squashed file system support

CONFIG_SQUASHFS

SquashFS 是一种高压缩率的只读文件系统,可以使用多种压缩算法(例如 zlib,xz,lzo).SquashFS 常用于嵌入式设备和 LiveCD 系统.

Squashfs XATTR support

CONFIG_SQUASHFS_XATTR

Squashfs 文件系统扩展属性(与 inode 关联的 name:value 对)支持.详见 attr 手册.不确定的选"N".

Include support for ZLIB compressed file systems

CONFIG_SQUASHFS_ZLIB

ZLIB 是 Squashfs 默认的标准压缩算法.在压缩率和性能之间达到了最佳的平衡.

Include support for LZO compressed file systems

CONFIG_SQUASHFS_LZO

LZO 是性能最佳的压缩算法(CPU 和内存占用都很低),但是压缩率确是最差的.常用于资源有限的嵌入式系统.

Include support for XZ compressed file systems

CONFIG_SQUASHFS_XZ

XZ 是压缩率最佳的压缩算法,但其 CPU 和内存占用都最高.可用于 PC 环境.

Use 4K device block size?

CONFIG_SQUASHFS_4K_DEVBLK_SIZE

出于降低潜伏时间的考虑,Squashfs 默认使用 1K 大小的块.但是在 MTD NAND 设备上,使用 4K 大小的块才可以获得最佳性能.此外,在大多数设备上,使用 4K 大小的块才能获得最佳连续读取性能.如果你的 Squashfs 位于闪存设备上,建议选"Y".否则建议选"N".

Additional option for memory-constrained systems

CONFIG_SQUASHFS_EMBEDDED

允许强制指定缓存大小.不确定的选"N".

Number of fragments cached

CONFIG_SQUASHFS_FRAGMENT_CACHE_SIZE

SquashFS 默认缓存最后 3 个从文件系统上读取的片段.降低此值(最小值是"1",不能设为"0")可以降低内存的占用,但是会增加底层物理设备的读取次数.增加此值则正好相反.[提示]按一般经验,大于"3"的值并不能带来显著的性能提升.

EFI Variable filesystem

CONFIG_EFIVAR_FS

efivarfs 是访问"EFI 变量"的新方式,意在取代传统的 sysfs(CONFIG_EFI_VARS)方式,其主要优点是可以突破 sysfs 中变量值不能超出 1024 字节的限制.开启后可支持各种操作 EFI 变量的工具:efivar,efibootmgr,vathpela/efibootmgr,uefivars,efitools,fwts(Firmware Test Suite).systemd 依赖于它.建议选"Y".

Network File Systems

CONFIG_NETWORK_FILESYSTEMS

网络文件系统

NFS client support

CONFIG_NFS_FS

NFS(Network File System)客户端支持,这样就可以使用 nfs-utils 包中的 mount.nfs 工具挂载远程服务器提供的 NFS 文件系统.详见 nfs 手册页.

NFS client support for NFS version 2

CONFIG_NFS_V2

NFSv2(RFC 1094)版本协议支持

NFS client support for NFS version 3

CONFIG_NFS_V3

NFSv3(RFC 1813)版本协议支持

NFS client support for the NFSv3 ACL protocol extension

CONFIG_NFS_V3_ACL

为 NFSv3 添加 POSIX ACL 支持(Solaris NFSv3 ACL).大多数 NFS 服务器都不支持这个扩展.不确定的选"N".

NFS client support for NFS version 4

CONFIG_NFS_V4

NFSv4(RFC 3530)版本协议支持

Provide swap over NFS support

CONFIG_NFS_SWAP

允许将 NFS 文件系统用做 swap 分区.

NFS client support for NFSv4.1

CONFIG_NFS_V4_1

NFSv4.1(RFC 5661)版本协议客户端支持,这样就可以使用 nfs-utils 包中的 mount.nfs 工具挂载远程服务器提供的 NFS 文件系统.详见 nfs 手册页以及 NFS 各个版本之间的比较.

NFSv4.1 Implementation ID Domain

CONFIG_NFS_V4_1_IMPLEMENTATION_ID_DOMAIN

NFSv4.1 规范新引入了会话机制,该选项定义在建立会话过程中使用在 EXCHANGE_ID 指令中的"domain"部分的值.这个值必须是个标准的 DNS 域名格式.如果你没有修改内核的 NFS 客户端代码,那么请保持默认值"kernel.org".

Root file system on NFS

CONFIG_ROOT_NFS

允许将 NFS 挂载为根文件系统(root=/dev/nfs),通常用于没有本地存储设备的无盘工作站(还必须开启 CONFIG_IP_PNP 以及至少一个子项).详见"Documentation/filesystems/nfs/nfsroot.txt"文档.

Provide NFS client caching support

CONFIG_NFS_FSCACHE

为 NFS 提供本地缓存支持,也就是利用 CONFIG_FSCACHE 选项的功能.

Use the legacy NFS DNS resolver

CONFIG_NFS_USE_LEGACY_DNS

内核现在有自己的 DNS 解析实现,如果你依然想使用老式的 DNS 解析脚本,可以选"Y".不确定的选"N".

NFS server support

CONFIG_NFSD

NFS 服务器端支持.要实现此功能,还需要 nfs-utils 软件包的支持.详见 nfs 手册页.这里也有一个 NFS 文章系列可以看看.此选项内嵌了 NFSv2 协议支持.

NFS server support for NFS version 3

CONFIG_NFSD_V3

NFSv3(RFC 1813)版本协议支持

NFS server support for the NFSv3 ACL protocol extension

CONFIG_NFSD_V3_ACL

为 NFSv3 添加 POSIX ACL 支持(Solaris NFSv3 ACL).此扩展并不属于 NFSv3 协议的官方内容.

NFS server support for NFS version 4

CONFIG_NFSD_V4

NFSv4(RFC 3530)版本协议支持

NFS server manual fault injection

CONFIG_NFSD_FAULT_INJECTION

仅供调试使用

Secure RPC: Kerberos V mechanism

CONFIG_RPCSEC_GSS_KRB5

为使用 Kerberos V5 GSS-API 身份验证机制(RFC1964)的安全 RPC 提供支持.要实现此功能,还需要 nfs-utils 软件包以及用户空间的 Kerberos 支持.

RPC: Enable dprintk debugging

CONFIG_SUNRPC_DEBUG

允许使用 rpcdebug 工具调试 RPC 故障,如果选"N"会让故障调试特别困难.

Ceph distributed file system

CONFIG_CEPH_FS

允许挂载 Ceph 分布式文件系统.不确定的选"N".详见"Documentation/filesystems/ceph.txt"文档.

CIFS support (advanced network filesystem, SMBFS successor)

CONFIG_CIFS

CIFS(Common Internet File System)协议客户端支持.CIFS 主要用于 Linux 与 Windows 之间共享文件系统.如果你打算挂载 Windows 的共享文件夹,或者由 Samba 提供的文件系统,就选"Y".详见"Documentation/filesystems/cifs.txt"文档.

CIFS statistics

CONFIG_CIFS_STATS

在/proc/fs/cifs/Stats 文件中显示每个被挂载的 CIFS 文件系统的统计信息

Extended statistics

CONFIG_CIFS_STATS2

在/proc/fs/cifs/目录下显示更详细的统计信息.对运行性能和内存占用都有些影响.不确定的选"N".

Support legacy servers which use weaker LANMAN security

CONFIG_CIFS_WEAK_PW_HASH

选"N",除非你确实知道自己在干什么.

Kerberos/SPNEGO advanced session setup

CONFIG_CIFS_UPCALL

Kerberos/SPNEGO 高级会话支持.不确定的选"N".

CIFS extended attributes

CONFIG_CIFS_XATTR

CIFS 文件系统扩展属性(与 inode 关联的 name:value 对)支持.不确定的选"N".

CIFS POSIX Extensions

CONFIG_CIFS_POSIX

CIFS POSIX 扩展.不确定的选"N".

Provide CIFS ACL support

CONFIG_CIFS_ACL

允许从服务器抓取 CIFS/NTFS ACL.不确定的选"N".

Enable CIFS debugging routines

CONFIG_CIFS_DEBUG

仅供调试使用

DFS feature support

CONFIG_CIFS_DFS_UPCALL

DFS(Distributed File System)支持.不确定的选"N".

SMB2 network file system support

CONFIG_CIFS_SMB2

仅供开发与调试使用

Provide CIFS client caching support

CONFIG_CIFS_FSCACHE

为 CIFS 提供本地缓存支持,也就是利用 CONFIG_FSCACHE 选项的功能.

NCP file system support (to mount NetWare volumes)

CONFIG_NCP_FS

NCP(NetWare Core Protocol)协议支持.这东西早就销声匿迹了,选"N".

Coda file system support (advanced network fs)
CONFIG_CODA_FS

Coda 是一种比 NFS 更先进的分布式集群文件系统.LVS(Linux Virtual Server)就采用了 Coda 分布式文件系统.详见"Documentation/filesystems/coda.txt"文档.

Andrew File System support
CONFIG_AFS_FS

AFS(Andrew File System)文件系统的实验性支持,目前仅支持只读访问.详见"Documentation/filesystems/afs.txt"文档.

Provide AFS client caching support
CONFIG_AFS_FSCACHE

为 AFS 提供本地缓存支持,也就是利用 CONFIG_FSCACHE 选项的功能.

Plan 9 Resource Sharing Support (9P2000)

CONFIG_9P_FS

9P2000 协议是 Plan 9 概念网络操作系统上使用的资源共享协议.不确定的选"N".

Native language support

CONFIG_NLS

本地语言支持.仅在你使用 FAT/NTFS/JOLIET 文件系统的情况下才需要这个东西.

Default NLS Option

CONFIG_NLS_DEFAULT

挂载文件系统时,控制台的默认本地语言(不是文件系统用于存储文件名的语言),建议设为"utf8"(因为控制台的默认编码是"utf8":vt.default_utf8=1).

{此处省略的各种字符集请按需选择}

Distributed Lock Manager (DLM)

CONFIG_DLM

通用的分布式锁管理器(DLM).用于为各种分布式文件系统提供通用的锁定支持.集群强烈依赖于这个驱动.

DLM debugging

CONFIG_DLM_DEBUG

仅供开发与调试使用

Kernel hacking

内核 hack 选项

Show timing information on printks

CONFIG_PRINTK_TIME

在控制台和 syslog()系统调用的输出中包含 printk()消息的时间戳,以便于直接显示内核启动过程中各步骤所用的时间.注意:无论此项是否开启,时间戳总会被记录在/dev/kmsg 中,开启此项仅相当于使用"printk.time=1"内核引导参数.

Default message log level (1-7)

CONFIG_DEFAULT_MESSAGE_LOGLEVEL

printk()内核消息日志的默认级别,取值范围是[1,7].任何由 printk 显示的字符串通常记录在/var/log/messages 文件里.数值越大显示的消息就越详细:1=ALERT,2=CRIT,3=ERR,4=WARNING,5=NOTICE,6=INFO,7=DEBUG.

Enable __deprecated logic

CONFIG_ENABLE_WARN_DEPRECATED

编译时开启"反对使用"逻辑检查,关闭此项将不会显示类似"warning: 'foo' is deprecated (declared at kernel/power/somefile.c:1234)"这样的警告消息.

Enable __must_check logic

CONFIG_ENABLE_MUST_CHECK

编译时开启"必须检查"逻辑检查,关闭此项将不会显示类似"warning: ignoring return value of 'foo', declared with attribute warn_unused_result"这样的警告消息.

Warn for stack frames larger than (needs gcc 4.4)

CONFIG_FRAME_WARN

堆栈帧大小警告阈值,设置过小会导致编译时警告太多,设为"0"可以关闭警告,需要 GCC-4.4 或更高版本

Strip assembler-generated symbols during link

CONFIG_STRIP_ASM_SYMS

连接时剥离汇编器产生的内部符号(类似'.Lxxx'),这样可以净化 get_wchan()之类的输出,同时还可以减小内核尺寸.建议开启.

Magic SysRq key

CONFIG_MAGIC_SYSRQ

开启"魔法键"(SysRq,允许用户按下 Alt+PrintScreen 后发送给内核特殊的命令)支持(可以通过"echo 0 >

/proc/sys/kernel/sysrq"关闭).由于 SysRq 会带来安全隐患(允许未经登录的操作),所以你应该仅在确实需要的场合开启.更多详情参见"Documentation/sysrq.txt"文档

Enable magic SysRq key functions by default

CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE

设置默认开启哪些魔法键.设为"1"表示开启所有魔法键,设为"0"表示禁用所有魔法键.或者按照"Documentation/sysrq.txt"文档的指引设置特定的码位.

Generate readable assembler code

CONFIG_READABLE_ASM

生成人类易读的汇编输出,以方便内核调试.这会禁用一些编译优化措施,也会降低内核的运行速度.

Enable unused/obsolete exported symbols

CONFIG_UNUSED_SYMBOLS

导出无用和废弃的符号,这将使内核不必要的增大.建议关闭.

Debug Filesystem

CONFIG_DEBUG_FS

debugfs 是内核开发者用来存储调试信息的虚拟文件系统.不搞内核开发就别选

Run 'make headers_check' when building vmlinux

CONFIG_HEADERS_CHECK

在编译内核时运行'make headers_check'命令检查内核头文件的正确性,当你修改了与用户空间相关的内核头文件后应该启用该选项

Enable full Section mismatch analysis

CONFIG_DEBUG_SECTION_MISMATCH

在编译时检查无效的引用.仅供内核开发者使用

Make section mismatch errors non-fatal

CONFIG_SECTION_MISMATCH_WARN_ONLY

若选"N",那么一旦出现"section mismatch",将会直接导致编译失败(而不是仅仅抛出警告).建议选"N".

Kernel debugging

CONFIG_DEBUG_KERNEL

仅供内核开发者使用.[提示]如果你开启了 CONFIG_EXPERT,此项会被强制选中,如果这不是你想要的,可以到内核源码树的根目录下使用"sed -i '/select DEBUG_KERNEL/d' usr/src/linux/init/Kconfig"命令去掉这个依赖.

Panic on Oops

CONFIG_PANIC_ON_OOPS

当内核 oops 时,直接 panic 掉(相当于 Windows 蓝屏死机),这样可以确保内核停止工作,避免导致无法预料的后果.等价于使用"oops=panic"内核引导参数.不确定的选"Y".

panic timeout

CONFIG_PANIC_TIMEOUT

如何处理内核崩溃(panic):(1)若设为"0"则表示无限等待,不做任何处理;(2)若设为正整数则表示等待设定的秒数之后重启;(3)若设为负整数则表示立即重启.

Enable extra timekeeping sanity checking

CONFIG_DEBUG_TIMEKEEPING

仅供内核开发者使用

Detect Hung Tasks

CONFIG_DETECT_HUNG_TASK

探测挂起的任务(进程被锁住或者冻结了,处于不可中断的"D"状态).由于仅能检测,不能做进一步的处理,所以仅用于帮助内核调试.

Collect scheduler debugging info

CONFIG_SCHED_DEBUG

提供一个"/proc/sched_debug"文件以帮助调试调度程序.仅供内核开发以及调试调度程序使用.

Collect scheduler statistics

CONFIG_SCHEDSTATS

收集调度程序的统计信息,并展示在"/proc/schedstat"文件中.可以用于调试调度程序,或者调整特定的应用程序.不确定的选"N".

Collect kernel timers statistics

CONFIG_TIMER_STATS

收集内核计时器的统计信息,并展示在"/proc/timer_stats"文件中.使用"echo 1 > /proc/timer_stats"开启统计,使用"echo 0 > /proc/timer_stats"关闭统计.不确定的选"N".

Debug object operations

CONFIG_DEBUG_OBJECTS

跟踪各种对象的生命周期(life time),并校验对这些对象的各种操作.仅供内核调试.

Debug slab memory allocations

CONFIG_DEBUG_SLAB

仅供内核开发者使用

SLUB debugging on by default

CONFIG_SLUB_DEBUG_ON

默认开启 SLUB 内存分配器调试功能.仅供调试,切勿用于生产环境.

Enable SLUB performance statistics

CONFIG_SLUB_STATS

收集 SLUB 内存分配器的性能统计信息.仅供调试,切勿用于生产环境.

Kernel memory leak detector

CONFIG_DEBUG_KMEMLEAK

内核内存泄漏检测.仅供内核调试.

Debug preemptible kernel

CONFIG_DEBUG_PREEMPT

对内核的主动抢占特性进行调试.仅供内核开发者使用

RT Mutex debugging, deadlock detection

CONFIG_DEBUG_RT_MUTEXES

仅供内核开发者使用

Built-in scriptable tester for rt-mutexes

CONFIG_RT_MUTEX_TESTER

仅供内核开发者使用

Spinlock and rw-lock debugging: basic checks

CONFIG_DEBUG_SPINLOCK

仅供内核开发者使用

Mutex debugging: basic checks

CONFIG_DEBUG_MUTEXES

仅供内核开发者使用

Lock debugging: detect incorrect freeing of live locks

CONFIG_DEBUG_LOCK_ALLOC

仅供内核开发者使用

Lock debugging: prove locking correctness

CONFIG_PROVE_LOCKING

仅供内核开发者使用

Lock usage statistics

CONFIG_LOCK_STAT

仅供内核开发者使用

Lock dependency engine debugging

CONFIG_DEBUG_LOCKDEP

仅供内核开发者使用

Sleep inside atomic section checking

CONFIG_DEBUG_ATOMIC_SLEEP

仅供内核开发者使用

Locking API boot-time self-tests

CONFIG_DEBUG_LOCKING_API_SELFTESTS

在内核启动时运行一个简短的加锁/解锁函数(spinlocks,rwlocks,mutexes,rwsems)自我测试.仅供内核开发者使用

Stack utilization instrumentation

CONFIG_DEBUG_STACK_USAGE

仅供内核开发者使用

kobject debugging

CONFIG_DEBUG_KOBJECT

仅供内核开发者使用

Verbose BUG() reporting (adds 70K)

CONFIG_DEBUG_BUGVERBOSE

在内核 panic 时让 BUG()函数报告更详细的信息.内核将会增大 70-100K.

Compile the kernel with debug info

CONFIG_DEBUG_INFO

以调试方式编译内核(gcc -g).仅供内核开发者使用

Debug VM

CONFIG_DEBUG_VM

仅供内核开发者使用

Debug VM translations

CONFIG_DEBUG_VIRTUAL

仅供内核开发者使用

Debug filesystem writers count

CONFIG_DEBUG_WRITECOUNT

仅供内核开发者使用

Debug memory initialisation

CONFIG_DEBUG_MEMORY_INIT

在内存初始化时增加额外的合理性检查.不确定的选"Y".

Debug linked list manipulation

CONFIG_DEBUG_LIST

仅供内核开发者使用

Linked list sorting test

CONFIG_TEST_LIST_SORT

仅供内核开发者使用

Debug SG table operations

CONFIG_DEBUG_SG

仅供内核开发者使用

Debug notifier call chains

CONFIG_DEBUG_NOTIFIERS

仅供内核开发者使用

Debug credential management

CONFIG_DEBUG_CREDENTIALS

仅供内核开发者使用

Compile the kernel with frame pointers

CONFIG_FRAME_POINTER

仅供内核开发者使用

Delay each boot printk message by N milliseconds

CONFIG_BOOT_PRINTK_DELAY

仅供内核开发者使用

Stack backtrace support

CONFIG_STACKTRACE

仅供内核开发者使用.

RCU Debugging

仅供内核开发者使用.建议所有子项全选"N".

Kprobes sanity tests

CONFIG_KPROBES_SANITY_TEST

仅供内核开发者使用

Self test for the backtrace code

CONFIG_BACKTRACE_SELF_TEST

仅供内核开发者使用

Force extended block device numbers and spread them

CONFIG_DEBUG_BLOCK_EXT_DEVT

仅供内核开发者使用

Force weak per-cpu definitions

CONFIG_DEBUG_FORCE_WEAK_PER_CPU

仅供内核开发者使用

Debug access to per_cpu maps

CONFIG_DEBUG_PER_CPU_MAPS

仅供内核开发者使用

Linux Kernel Dump Test Tool Module

CONFIG_LKDTM

仅供内核开发者使用

Notifier error injection

CONFIG_NOTIFIER_ERROR_INJECTION

仅供内核开发者使用

Fault-injection framework

CONFIG_FAULT_INJECTION

仅供内核开发者使用

Latency measuring infrastructure

CONFIG_LATENCYTOP

仅供内核开发者使用

Strict user copy size checks

CONFIG_DEBUG_STRICT_USER_COPY_CHECKS

仅供内核开发者使用

Debug page memory allocations

CONFIG_DEBUG_PAGEALLOC

仅供内核开发者使用

Tracers

CONFIG_FTRACE

仅供内核开发者使用.建议选"N".

Runtime Testing

运行时自我检查.建议子项全选"N".

Red-Black tree test

CONFIG_RBTREE_TEST

仅供内核开发者使用

Interval tree test

CONFIG_INTERVAL_TREE_TEST

仅供内核开发者使用

Remote debugging over FireWire early on boot

CONFIG_PROVIDE_OHCI1394_DMA_INIT

仅供内核开发者使用

Remote debugging over FireWire with firewire-ohci

CONFIG_FIREWIRE_OHCI_REMOTE_DMA

仅供内核开发者使用

Build targets in Documentation/ tree

CONFIG_BUILD_DOCSRC

编译内核源码树下"Documentation"目录中的目标.不确定的选"N".

Enable dynamic printk() support

CONFIG_DYNAMIC_DEBUG

仅供内核开发者使用

Enable debugging of DMA-API usage

CONFIG_DMA_API_DEBUG

仅供内核开发者使用

Perform an atomic64_t self-test at boot

CONFIG_ATOMIC64_SELFTEST

仅供内核开发者使用

Self test for hardware accelerated raid6 recovery

CONFIG_ASYNC_RAID6_TEST

仅供内核开发者使用

Sample kernel code

CONFIG_SAMPLES

内核示例代码.仅供内核开发者使用

KGDB: kernel debugger

CONFIG_KGDB

仅供内核开发者使用

kmemcheck: trap use of uninitialized memory

CONFIG_KMEMCHECK

仅供内核开发者使用

Test functions located in the string_helpers module at runtime

CONFIG_TEST_STRING_HELPERS

仅供内核开发者使用

Test kstrto*() family of functions at runtime

CONFIG_TEST_KSTRTOX

仅供内核开发者使用

Filter access to /dev/mem

CONFIG_STRICT_DEVMEM

如果选"N",那么用户空间的 root 用户将可以通过/dev/mem 访问所有内存空间(包括用户空间与内核空间),以方便调试内核.如果选"Y",那么内核空间除了 PCI 和 BIOS 部分以及数据区之外,都禁止访问,以保护系统安全.不确定的选"Y".

Enable verbose x86 bootup info messages

CONFIG_X86_VERBOSE_BOOTUP

在启动时显示额外 bzimage 解压消息,显示详细的内核引导信息.建议选"N"使引导过程更安静(依然会显示错误信息).

Early printk

CONFIG_EARLY_PRINTK

将内核日志直接输出到 VGA 缓冲或串口.这有助于调试那些在控制台尚未完成初始化之前就造成系统崩溃的 bug.

Early printk via EHCI debug port

CONFIG_EARLY_PRINTK_DBGP

支持将内核日志直接通过 EHCI 调试端口输出.选"N",除非你想调试内核.

Dump the EFI pagetable

CONFIG_EFI_PGT_DUMP

仅供内核开发者使用

Check for stack overflows

CONFIG_DEBUG_STACKOVERFLOW

仅供内核开发者使用

Export kernel pagetable layout to userspace via debugfs

CONFIG_X86_PTDUMP

仅供内核开发者使用

Write protect kernel read-only data structures

CONFIG_DEBUG_RODATA

仅供内核开发者使用

Set loadable kernel module data as NX and text as RO

CONFIG_DEBUG_SET_MODULE_RONX

将内核模块的数据区标记为 NX(不可执行),文本段标记为 RO(只读),以防止不良模块(例如被植入病毒的模块)对系统的破坏,也能预防某些类型的内核入侵.这需要 CPU 支持 NX 位(CPU flags

中要含有"nx"标志).但是这也有副作用:会与运行时代码补丁冲突,还会导致动态内核跟踪失效.建议选"Y",除非你需要调试内核,或者需要为运行中的模块打补丁.

Testcase for the NX non-executable stack feature

CONFIG_DEBUG_NX_TEST

对处理器的 NX 的测试用例.仅供内核开发者使用

Set upper limit of TLB entries to flush one-by-one

CONFIG_DEBUG_TLBFLUSH

仅供内核开发者使用

Enable doublefault exception handler

CONFIG_DOUBLEFAULT

仅供内核开发者使用

Enable IOMMU debugging

CONFIG_IOMMU_DEBUG

仅供内核开发者使用

Enable IOMMU stress-test mode

CONFIG_IOMMU_STRESS

仅供内核开发者使用

IOMMU leak tracing

CONFIG_IOMMU_LEAK

仅供内核开发者使用

x86 instruction decoder selftest

CONFIG_X86_DECODER_SELFTEST

仅供内核开发者使用

IO delay type

IO 延迟方式

port 0x80 based port-IO delay

CONFIG_IO_DELAY_0X80

传统的 Linux IO 延迟方式,久经考验,也是最安全的默认值.

port 0xed based port-IO delay

CONFIG_IO_DELAY_0XED

基于 0xed 端口的 IO 延迟方式,主要是为了避免和基于 0x80 端口的主板诊断卡冲突.绝大多数人都没有主板诊断卡,除非那些专门维修主板的硬件工程师.

udelay based port-IO delay

CONFIG_IO_DELAY_UDELAY

使用内核端 udelay()函数作为延迟方法(简单的延迟 2 微秒).可以不占用任何 IO 端口空间.

no port-IO delay

CONFIG_IO_DELAY_NONE

不使用任何 port-IO 延迟机制.只要你的机器不是老古董,建议选择此项.

Debug boot parameters

CONFIG_DEBUG_BOOT_PARAMS

仅供内核开发者使用

CPA self-test code

CONFIG_CPA_DEBUG

仅供内核开发者使用

Allow gcc to uninline functions marked 'inline'

CONFIG_OPTIMIZE_INLINING

允许 GCC 将标记为内联(inline)的函数变成非内联(uninline).选"Y"后将完全无视代码中的"inline"标记,完全由 GCC 自行决定是否应该将函数内联.由于 GCC4.x 系列更新了判断函数是否应该内联的算法,选"Y"后,编译出的内核体积会减小(但运行速度未必提升).建议选"Y".

NMI Selftest

CONFIG_DEBUG_NMI_SELFTEST

对不可屏蔽中断(Non Maskable Interrupt)进行自检,仅供内核开发者使用

ATOM Punit debug driver

CONFIG_PUNIT_ATOM_DEBUG

仅供内核开发者使用

Security options

安全选项

这里的选项不明白的建议不要选,否则有可能弄巧成拙.

Enable access key retention support

CONFIG_KEYS

在内核中保留认证令牌(authentication token)和访问密钥(access key).eCryptfs(CONFIG_ECRYPT_FS)与 Docker 依赖于它.不确定的选"N".

TRUSTED KEYS

CONFIG_TRUSTED_KEYS

"TRUSTED KEY"的意思是由 TPM(可信赖平台模块)用 RSA 算法封装的一对随机数.开启此项后,内核将可以为创建/封装/解封"TRUSTED KEY"提供支持.如果引导 PCR(平台配置寄存器)和各种条件都匹配,那么 TPM 只解封密钥.用户空间永远只能看到加密过后的二进制内容.不确定的选"N".

ENCRYPTED KEYS

CONFIG_ENCRYPTED_KEYS

"ENCRYPTED KEY"的意思是由内核封装的一对随机数,该对随机数可以用一个"主密钥"使用对称加密算法进行加密和解密.开启此项后,内核将可以为创建/加密/解密"ENCRYPTED KEY"提供支

持."主密钥"既可以是"TRUSTED KEY"也可以是"user-key"(用户选择的密钥).用户空间永远只能看到/存储加密过后的二进制内容.不确定的选"N".

Enable the /proc/keys file by which keys may be viewed

CONFIG_KEYS_DEBUG_PROC_KEYS

开启"/proc/keys"文件支持,该文件中保存了系统上所有可见的密钥.注意,LSM(Linux 安全模块)安全检查仍然是必须的.不确定的选"N".

Restrict unprivileged access to the kernel syslog

CONFIG_SECURITY_DMESG_RESTRICT

禁止非特权用户访问内核日志(dmesg),相当于"echo 1 > /proc/sys/kernel/dmesg_restrict".不确定的选"N".

Enable different security models

CONFIG_SECURITY

允许内核选择不同的 LSM(Linux 安全模块),如果未选中则内核将使用默认的安全模块("Default security module").不确定的选"N".

Enable the securityfs filesystem

CONFIG_SECURITYFS

securityfs 安全文件系统支持.当前仅被 TPM bios 字符设备驱动以及 IMA(完整性提供者)使用.它与 SELinux 或 SMACK 之类没有关系.不确定的选"N".

Socket and Networking Security Hooks

CONFIG_SECURITY_NETWORK

允许安全模块通过 Security Hook 对 Socket 与 Networking 进行访问控制.不确定的选"N".

XFRM (IPSec) Networking Security Hooks

CONFIG_SECURITY_NETWORK_XFRM

为 XFRM(IPSec)启用安全 Hook.这样安全模块可以通过这些 hook,根据 IPSec 策略标签,实现针对每个网络包的访问控制.非 IPSec 通信则被当做"无标签"处理,仅允许那些被明确批准可以不使用策略标签的 socket 才能不通过 IPSec 进行通信.不确定的选"N".

Security hooks for pathname based access control

CONFIG_SECURITY_PATH

此安全钩子程序可以让各种安全模块实现基于路径的访问控制.不确定的选"N".

Enable Intel(R) Trusted Execution Technology (Intel(R) TXT)

CONFIG_INTEL_TXT

支持使用可信引导(Trusted Boot)技术引导内核(需要使用 tboot 模块).这将使用英特尔 TXT(可信任执行技术)来引导内核.在不支持 TXT 的平台上开启此项没有效果.详见 "Documentation/intel_txt.txt"文档.不确定的选"N".

Low address space for LSM to protect from user allocation

CONFIG_LSM_MMAP_MIN_ADDR

禁止用户空间分配的低位内存范围.禁止用户写入低位内存有助于降低内核 NULL 指针漏洞造成的破坏(参见 CONFIG_DEFAULT_MMAP_MIN_ADDR 选项).建议保持默认值 "65536".

NSA SELinux Support

CONFIG_SECURITY_SELINUX

SELinux(安全增强 Linux)是美国国家安全局(NSA)开发的 Linux 安全模块,它拥有一个灵活而强制性的访问控制结构,可防御未知攻击,相当于 B1 级的军事安全性能(比微软所谓的 C2 等高得多).应用 SELinux 后,可以减轻恶意攻击或恶意软件带来的灾难,对机密性和完整性有很高要求的信息,亦可提供很高的安全保障.但另一方面,如果不深入了解 SELinux 知识而盲目使用,则会弄巧成拙.不确定的选"N".

NSA SELinux boot parameter

CONFIG_SECURITY_SELINUX_BOOTPARAM

添加"selinux"内核引导参数.以允许在引导时使用'selinux=0'禁用 SELinux 或'selinux=1'启用 SELinux.

NSA SELinux boot parameter default value

CONFIG_SECURITY_SELINUX_BOOTPARAM_VALUE

"selinux"内核引导参数的默认值.

NSA SELinux runtime disable

CONFIG_SECURITY_SELINUX_DISABLE

允许在运行时禁用 SELinux.建议选"N".

NSA SELinux Development Support

CONFIG_SECURITY_SELINUX_DEVELOP

SELinux 开发支持.开启此项后,除非明确使用"enforcing=1"引导参数让内核以"强制模式"运行,否则内核将以"许可模式"运行(记录所有事件,同时允许所有操作).主要用于测试 SELinux 以及策略开发.此外,开启此项后,还可以在运行时通过"/selinux/enforce"让内核在"强制模式"与"许可模式"之间切换.

NSA SELinux AVC Statistics

CONFIG_SECURITY_SELINUX_AVC_STATS

搜集访问向量缓存(access vector cache)的统计信息并在/selinux/avc/cache_stats 中显示出来.这些信息可以用 avcstat 之类的工具查看.

NSA SELinux checkreqprot default value

CONFIG_SECURITY_SELINUX_CHECKREQPROT_VALUE

内核引导参数"checkreqprot"的默认值.设为"0"表示默认检查内核要求执行的保护策略,设为"1"表示默认检查应用程序要求执行的保护策略.此值还可以在运行时通过/selinux/checkreqprot 修改.不确定的选"1".

NSA SELinux maximum supported policy format version

CONFIG_SECURITY_SELINUX_POLICYDB_VERSION_MAX

将支持的策略格式的最高版本设置为一个特定的数值.该数值将通过/selinux/policyvers 向用户空间报告,并在加载策略时被使用.不确定的选"N".

NSA SELinux maximum supported policy format version value

支持策略格式的最高版本的数值.可以通过"checkpolicy -V"命令检查当前工具链支持的版本数值.

Simplified Mandatory Access Control Kernel Support

CONFIG_SECURITY_SMACK

Smack(简化的强制访问控制内核)内核安全模块.Smack 是一种简单而有效的强制访问控制机制,它的简单体现在安全策略的配

置很简单,它的有效体现在完全使用 LSM 作为其控制手段.不确定的选"N".

TOMOYO Linux Support

CONFIG_SECURITY_TOMOYO

TOMOYO Linux 是日本 NTT 数据公司开发的一种 Linux 安全模块.不确定的选"N".

AppArmor support

CONFIG_SECURITY_APPARMOR

AppArmor(应用盔甲)是来自 Novell 的一种 Linux 安全模块.AppArmor 使用文件路径来跟踪程序限制,是最容易配置的安全模块.不确定的选"N".

Yama support

CONFIG_SECURITY_YAMA

Yama(阎王)是 3.4 版内核新引入的一种 Linux 安全模块.不确定的选"N".

Digital signature verification using multiple keyrings

CONFIG_INTEGRITY_SIGNATURE

允许使用多个密钥环(keyring)进行数字签名验证,也就允许为多个不同的使用场合(evm,ima,module)分别使用不同的 keyring.看不懂的选"N".

Enable asymmetric keys support

CONFIG_INTEGRITY_ASYMMETRIC_KEYS

允许使用非对称密钥进行数字签名验证.

Integrity Measurement Architecture(IMA)

CONFIG_IMA

IMA(完整性度量架构)是一个在 TCG(可信计算工作组)技术规范之上提出的完整性检查技术.IMA 维护着一个系统关键文件的哈希值列表,从而可以检测这些关键文件是否被篡改.如果系统上有 TPM 安全芯片,那么 IMA 还会在 TPM 芯片内存储哈希值的集合.这样的 TPM 芯片可以提供给第三方,用于检查系统上的关键文件是否被篡改.不确定的选"N".

Enables auditing support

CONFIG_IMA_AUDIT

添加"ima_audit"内核引导参数支持.当设为"ima_audit=1"时,将允许显示完整性审计信息.

Appraise integrity measurements

CONFIG_IMA_APPRAISE

本地完整性鉴定支持.这样就可以在加载文件时检验它的完整性.这要求系统配置 EVM 支持.不确定的选"N".

EVM support

CONFIG_EVM

EVM 通过保护文件的安全扩展属性来对抗完整性攻击.

EVM HMAC version

CONFIG_EVM_HMAC_VERSION

支持的 EVM HMAC 版本:"1"表示原始版本,默认值"2"表示添加了文件系统 UUID 支持的改进版本.

Default security module

内核默认的安全模块.[提示]"Unix Discretionary Access Controls"是经典的 UNIX 基于目录的访问控制安全模型.如果没有开启任何安全模块,这将是默认值.

Cryptographic API

内核加密 API 支持

这里的加密算法被广泛的应用于驱动程序通信协议等机制中.子选项可以全不选,内核中若有其他部分依赖它,会自动选上.使用内核树外的模块时可能需要手动选择.

FIPS 200 compliance

CONFIG_CRYPTO_FIPS

"fips"内核引导参数支持.这是在 FIPS200 认证的系统中运行所必须的.选"N",除非你确实知道自己在做什么.

RSA algorithm

CONFIG_CRYPTO_RSA

"RSA"公钥加密算法.能同时用于密钥交换和数字签名.

Cryptographic algorithm manager

CONFIG_CRYPTO_MANAGER

创建默认的加密模版实例,提供了操作内核的加密特性所需的软件.

Userspace cryptographic algorithm configuration

CONFIG_CRYPTO_USER

允许用户空间配置内核加密实例.不确定的选"N".

Disable run-time self tests

CONFIG_CRYPTO_MANAGER_DISABLE_TESTS

禁止在注册算法时进行简单的自我检测.不确定的选"Y".

GF(2^128) multiplication functions

CONFIG_CRYPTO_GF128MUL

由高效表格驱动的伽罗瓦域 GF(2^128)乘法器支持.某些加密模式需要它.不确定的选"N".如果有其它模块需要此特性,会被自动选中.

Null algorithms

CONFIG_CRYPTO_NULL

NULL 加密算法(什么也不做),用于 IPsec 协议的封装安全载荷模块(ESP)

Parallel crypto engine

CONFIG_CRYPTO_PCRYPT

将任意加密算法转化成并行算法,并在内核线程中执行.

Software async crypto daemon

CONFIG_CRYPTO_CRYPTD

这是一个通用的软件异步加密守护进程,可将任意的同步软件加密算法转换成在内核线程中执行的异步算法.

Software async multi-buffer crypto daemon

CONFIG_CRYPTO_MCRYPTD

这是一个通用的软件异步加密守护进程,可将任意的多缓冲加密算法转换成在内核线程中执行的异步多缓冲加密算法.

Authenc support

CONFIG_CRYPTO_AUTHENC

用于 IPSec 组合模式的包装器:认证加密并对 IPSec 提供多重加密.

Testing module

CONFIG_CRYPTO_TEST

丑陋的加密测试模块.仅供调试使用.

CCM support

CONFIG_CRYPTO_CCM

CBC-MAC 计数器.IPsec 需要它.

GCM/GMAC support

CONFIG_CRYPTO_GCM

GCM(Galois/Counter Mode)与 GMAC(Galois Message Authentication Code)支持.IPsec 需要它.[注释]GCM 是一种对称加密算法的块密码工作模式,使用 128 位块大小.块密码工作模式可以分为加密模式,认证模式,认证加密模式.GCM 模式为认证模式的一种,提供认证和加密两种功能.

Sequence Number IV Generator

CONFIG_CRYPTO_SEQIV

序号初始向量(IV)生成器.它基于一个序号与一个盐粒子(salt)的异或值生成一个向量.此算法主要用于块密码的 CTR(计数模式).

CBC support

CONFIG_CRYPTO_CBC

块密码工作模式:密码分组链接(Cipher Block Chaining)模式.IPSec 需要它.

CTR support

CONFIG_CRYPTO_CTR

块密码工作模式:计数器(Counter)模式.IPSec 需要它.

CTS support

CONFIG_CRYPTO_CTS

块密码工作模式:密文窃取(Cipher Text Stealing)模式.Kerberos gss 机制支持的 AES 加密需要它.

ECB support

CONFIG_CRYPTO_ECB

块密码工作模式电子密码本(Electronic CodeBook)模式.这是最简单的分组密码算法,只是简单的分别加密每个块.

LRW support

CONFIG_CRYPTO_LRW

块密码工作模式:LRW(Liskov Rivest Wagner)模式.这个模式以三个人名命名.这是一种小数据块加密模式,加密后的数据保持与明文数据同样的长度,专门用于 CONFIG_DM_CRYPT 模块加密磁盘区块(使用"aes-lrw-benbi"指定).

PCBC support

CONFIG_CRYPTO_PCBC

块密码工作模式:填充密码块链接(Propagating Cipher Block Chaining)模式.CONFIG_AF_RXRPC 需要它.

XTS support

CONFIG_CRYPTO_XTS

块密码工作模式:XTS 模式.这是 IEEE1619/D16 规范制定的一种小数据块加密模式,加密后的数据保持与明文数据同样的长度,专门用于加密磁盘区块(使用"aes-xts-plain"指定).

CMAC support

CONFIG_CRYPTO_CMAC

NIST(美国国家标准与技术研究所)制定的基于密文的消息认证码(Cipher-based Message Authentication Code)

HMAC support

CONFIG_CRYPTO_HMAC

基于哈希的消息验证代码(RFC2104).在发送方和接收方共享机密密钥的前提下,HMAC 可用于确定通过不安全信道发送的消息是否被篡改.IPSec 需要它.systemd 依赖于它.

XCBC support

CONFIG_CRYPTO_XCBC

基于哈希的加密算法(RFC3566)

VMAC support

CONFIG_CRYPTO_VMAC

VMAC 是一种专用于 64 位 CPU 的高速消息认证算法

CRC32c CRC algorithm

CONFIG_CRYPTO_CRC32C

CRC32c 摘要算法是常见的 CRC32 循环冗余校验的一个变种,仅多项式常数不同,算法完全一样.常用于 iSCSI 和 SCTP 数据校验.

CRC32c INTEL hardware acceleration

CONFIG_CRYPTO_CRC32C_INTEL

利用 SSE4.2 指令集中专用的"CRC32"指令,可以提高最少 10 倍的运算速度.如果你的 CPU 支持 SSE4.2,建议选"Y".

CRC32 CRC algorithm

CONFIG_CRYPTO_CRC32

经典的 CRC32 循环冗余校验算法.

CRC32 PCLMULQDQ hardware acceleration

CONFIG_CRYPTO_CRC32_PCLMUL

使用处理器的 PCLMULQDQ 指令(又称 CLMUL 指令集,其实只有一条指令)加速 CRC32 的运算.PCLMULQDQ 是从 Intel Westmere 和 AMD Bulldozer 开始引入的指令(隶属于 AES 指令集).可以大幅提升 CRC32 的运算速度.如果你的 CPU 支持 AES 指令集(grep aes /proc/cpuinfo),建议选"Y".

CRCT10DIF algorithm

CONFIG_CRYPTO_CRCT10DIF

CRC T10 DIF(Data Integrity Field)算法可用于保障 T10/SCSI 设备的端到端的数据完整性.

CRCT10DIF PCLMULQDQ hardware acceleration

CONFIG_CRYPTO_CRCT10DIF_PCLMUL

使用处理器的 PCLMULQDQ 指令(又称 CLMUL 指令集,其实只有一条指令)加速 CRC T10 DIF 的运算.PCLMULQDQ 是从 Intel Westmere 和 AMD Bulldozer 开始引入的指令(隶属于 AES 指令集).可以大幅提升 CRC T10 DIF 的运算速度.如果你的 CPU 支持 AES 指令集(grep aes /proc/cpuinfo),建议选"Y".

GHASH digest algorithm

CONFIG_CRYPTO_GHASH

GHASH 是用于 GCM(Galois/Counter Mode)的消息摘要算法.

MD4 digest algorithm

CONFIG_CRYPTO_MD4

老旧的 MD4(RFC1320)摘要算法,已经被淘汰.

MD5 digest algorithm

CONFIG_CRYPTO_MD5

广泛使用的 MD5(RFC1321)摘要算法,128 位.已经被发现可以快速找到碰撞,正逐渐淘汰中.

Michael MIC keyed digest algorithm

CONFIG_CRYPTO_MICHAEL_MIC

Michael MIC 是仅用于 TKIP(IEEE 802.11i)的摘要算法.不能用于其它场合,因为它存在一些缺陷.

RIPEMD-128 digest algorithm

CONFIG_CRYPTO_RMD128

RIPEMD-128(ISO/IEC 10118-3:2004)128 位摘要算法.安全性不高,不建议使用.

RIPEMD-160 digest algorithm

CONFIG_CRYPTO_RMD160

RIPEMD-160(ISO/IEC 10118-3:2004)160 位摘要算法.是替代各种 128 位摘要算法(RIPEMD-128,MD5,MD4)的首选.其运算速度和 SHA1 相当,但是目前尚无已知有效的攻击方法.

RIPEMD-256 digest algorithm

CONFIG_CRYPTO_RMD256

RIPEMD-256 在本质上和 RIPEMD-128 是一样的.因为 RIPEMD 的设计者们根本就没有真正设计 256 和 320 位这两种标准,他们只是在 128 位和 160 位的基础上,修改了初始参数和 s-box 来达到输出为 256 和 320 位的目的.所以,256 位的强度和 128 相当,而 320 位的强度和 160 位相当.

RIPEMD-320 digest algorithm

CONFIG_CRYPTO_RMD320

RIPEMD-320 在本质上和 RIPEMD-160 是一样的.因为 RIPEMD 的设计者们根本就没有真正设计 256 和 320 位这两种标准,他们只是在 128 位和 160 位的基础上,修改了初始参数和 s-box 来达到输出为 256 和 320 位的目的.所以,256 位的强度和 128 相当,而 320 位的强度和 160 位相当.

SHA1 digest algorithm

CONFIG_CRYPTO_SHA1

目前使用最广泛的 SHA-1(FIPS 180-1/DFIPS 180-2)160 位摘要算法是 SHA 家族中的一员,在许多安全协议中广为使用(TLS,SSL,PGP,SSH,S/MIME,IPsec 等).SHA-1 曾被视为是 MD5 的后继者,但由于出现了针对 SHA-1 的理论上破解的方法(不等于实践中被破解),有些人已经开始改用其它的替代算法(例如 SHA-3).

SHA1 digest algorithm (SSSE3/AVX)

CONFIG_CRYPTO_SHA1_SSSE3

使用 SSSE3/AVX 指令集加速 SHA-1 的计算.如果你的 CPU 支持 SSSE3/AVX 指令集,建议选"Y".

SHA256 digest algorithm (SSSE3/AVX/AVX2)

CONFIG_CRYPTO_SHA256_SSSE3

使用 SSSE3/AVX/AVX2 指令集加速 SHA-256 的计算.

SHA512 digest algorithm (SSSE3/AVX/AVX2)

CONFIG_CRYPTO_SHA512_SSSE3

使用 SSSE3/AVX/AVX2 指令集加速 SHA-512 的计算.

SHA224 and SHA256 digest algorithm

CONFIG_CRYPTO_SHA256

SHA-224 和 SHA-256 摘要算法,速度较 SHA1 稍慢,都属于"SHA-2"系列,目前尚无已知的有效攻击方法.但并未被广泛使用.systemd 依赖于它.

SHA384 and SHA512 digest algorithms

CONFIG_CRYPTO_SHA512

SHA-384 和 SHA-512 摘要算法,速度大约只有 SHA1 的 40-50%,都属于"SHA-2"系列,目前尚无已知的有效攻击方法.但并未被广泛使用.

Tiger digest algorithms

CONFIG_CRYPTO_TGR192

Tiger 号称是最快的哈希算法,专门为 64 位机器做了优化.

Whirlpool digest algorithms

CONFIG_CRYPTO_WP512

Whirlpool 是一种 512 位的摘要算法,利用了已有的 AES 分组密码算法构造 Hash 函数,拥有相当高的安全性,已经被列入了 ISO 标准,目前最新版本为 3.0(2003 年发布).

GHASH digest algorithm (CLMUL-NI accelerated)

CONFIG_CRYPTO_GHASH_CLMUL_NI_INTEL

使用 CPU 的 CLMUL 指令集(包含在 AES 指令集中)加速 GHASH 摘要算法.

AES cipher algorithms

CONFIG_CRYPTO_AES

AES(FIPS-197)又称"Rijndael",是目前最佳的对称加密算法,快速且节省内存,可以使用 128/192/256 位密钥,是目前使用最广泛的对称加密算法.

AES cipher algorithms (x86_64)

CONFIG_CRYPTO_AES_X86_64

针对 x86_64 架构的 AES 实现.

AES cipher algorithms (AES-NI)

CONFIG_CRYPTO_AES_NI_INTEL

使用 AES 指令集加速 AES 的计算.如果你的 CPU 支持 AES 指令集(grep aes /proc/cpuinfo),建议选"Y".

Anubis cipher algorithm

CONFIG_CRYPTO_ANUBIS

Anubis 是一种分组密码算法.分组长度为 128 位,密钥长度可变(最低 128 位),圈数可变(最低 12 圈).是欧洲于 2000 年 1 月 1 日启动的 NESSIE 计划 17 个候选分组加密算法之一.

ARC4 cipher algorithm

CONFIG_CRYPTO_ARC4

一种脆弱的流对称加密算法,仅用于已经被淘汰的 WEP.

Blowfish cipher algorithm

CONFIG_CRYPTO_BLOWFISH

Blowfish 对称加密算法,一种又老又慢的对称加密算法.

Blowfish cipher algorithm (x86_64)

CONFIG_CRYPTO_BLOWFISH_X86_64

针对 x86_64 架构的 Blowfish 实现

Blowfish cipher algorithm (x86_64/AVX2)

CONFIG_CRYPTO_BLOWFISH_AVX2_X86_64

使用 AVX2 指令集加速 Blowfish 的计算.

Camellia cipher algorithms

CONFIG_CRYPTO_CAMELLIA

Camellia 是欧盟 NESSIE 项目的选定算法,也是日本 CRYPTREC 项目的推荐算法.可以使用 128/192/256 位密钥,具有与 AES 同等级的安全强度及运算速度.

Camellia cipher algorithm (x86_64)

CONFIG_CRYPTO_CAMELLIA_X86_64

针对 x86_64 架构的 Camellia 实现

Camellia cipher algorithm (x86_64/AES-NI/AVX)

CONFIG_CRYPTO_CAMELLIA_AESNI_AVX_X86_64

使用 AES 指令集/AVX 指令集加速 Camellia 的计算.

Camellia cipher algorithm (x86_64/AES-NI/AVX2)

CONFIG_CRYPTO_CAMELLIA_AESNI_AVX2_X86_64

使用 AES 指令集/AVX2 指令集加速 Camellia 的计算.

CAST5 (CAST-128) cipher algorithm

CONFIG_CRYPTO_CAST5

老旧的 CAST5(CAST-128)对称加密算法

CAST5 (CAST-128) cipher algorithm (x86_64/AVX)

CONFIG_CRYPTO_CAST5_AVX_X86_64

使用 AVX 指令集加速 CAST5 的计算.

CAST6 (CAST-256) cipher algorithm

CONFIG_CRYPTO_CAST6

老旧的 CAST6(CAST-256)对称加密算法

CAST6 (CAST-256) cipher algorithm (x86_64/AVX)

CONFIG_CRYPTO_CAST6_AVX_X86_64

使用 AVX 指令集加速 CAST6 的计算.

DES and Triple DES EDE cipher algorithms

CONFIG_CRYPTO_DES

老旧的 DES 和三重 DES 对称加密算法.

FCrypt cipher algorithm

CONFIG_CRYPTO_FCRYPT

FCrypt 对称加密算法仅用于 CONFIG_AF_RXRPC

Khazad cipher algorithm

CONFIG_CRYPTO_KHAZAD

Khazad 是一种最终进入 NESSIE 决赛的对称加密算法,专为 64 位 CPU 设计,支持 128 位密钥.

Salsa20 stream cipher algorithm

CONFIG_CRYPTO_SALSA20

Salsa20 是一种流密码算法,也是 eSTREAM 工程最终胜选算法之一.

Salsa20 stream cipher algorithm (x86_64)

CONFIG_CRYPTO_SALSA20_X86_64

针对 x86_64 架构的 Salsa20 实现

SEED cipher algorithm

CONFIG_CRYPTO_SEED

SEED(RFC4269)对称分组加密算法,采用 128 位密钥,是韩国的国家标准.

Serpent cipher algorithm

CONFIG_CRYPTO_SERPENT

Serpent 对称加密算法曾经是 AES 的最终 5 个候选算法之一,因为速度较 Rijndael 慢而最终得票数次之.目前尚未发现针对 Serpent 的有效攻击,因此被认为是一种强安全算法(甚至被认为比 Rijndael 更安全).

Serpent cipher algorithm (x86_64/SSE2)

CONFIG_CRYPTO_SERPENT_SSE2_X86_64

使用 SSE2 指令集加速 Serpent 的计算.

Serpent cipher algorithm (x86_64/AVX)

CONFIG_CRYPTO_SERPENT_AVX_X86_64

使用 AVX 指令集加速 Serpent 的计算.

Serpent cipher algorithm (x86_64/AVX2)

CONFIG_CRYPTO_SERPENT_AVX2_X86_64

使用 AVX2 指令集加速 Serpent 的计算.

TEA, XTEA and XETA cipher algorithms

CONFIG_CRYPTO_TEA

较弱的几种对称加密算法

Twofish cipher algorithm

CONFIG_CRYPTO_TWOFISH

Twofish 是派生自 Blowfish 的对称加密算法,曾经是 AES 的最终 5 个候选算法之一,最终得票数第三.

Twofish cipher algorithm (x86_64)

CONFIG_CRYPTO_TWOFISH_X86_64

针对 x86_64 架构的 Twofish 实现

Twofish cipher algorithm (x86_64, 3-way parallel)

CONFIG_CRYPTO_TWOFISH_X86_64_3WAY

针对 x86_64 架构的三路并行 Twofish 实现.能够充分利用乱序执行 CPU 的指令周期.

Twofish cipher algorithm (x86_64/AVX)

CONFIG_CRYPTO_TWOFISH_AVX_X86_64

使用 AVX 指令集加速 Twofish 的计算.

Twofish cipher algorithm (x86_64/AVX2)

CONFIG_CRYPTO_TWOFISH_AVX2_X86_64

使用 AVX2 指令集加速 Twofish 的计算.

Deflate compression algorithm

CONFIG_CRYPTO_DEFLATE

Deflate(RFC1951)无损数据压缩算法.当在 IPSec 中使用 IPCOMP 协议时才需要.

Zlib compression algorithm

CONFIG_CRYPTO_ZLIB

zlib 无损数据压缩算法是一种事实上的业界标准.被广泛应用.

LZO compression algorithm

CONFIG_CRYPTO_LZO

LZO是致力于解压速度的一种无损数据压缩算法.

Pseudo Random Number Generation for Cryptographic modules

CONFIG_CRYPTO_ANSI_CPRNG

符合ANSI(美国国家标准学会)X9.31-1998附录A.2.4所描述的伪随机数发生器(基于3DES).这是一种较老的算法,生成的随机数质量不高.

NIST SP800-90A DRBG

CONFIG_CRYPTO_DRBG_MENU

符合NIST(美国国家标准技术局)SP800-90A标准的伪随机数发生器(DRBG).SP800-90A是美国政府FIPS 140-2安全认证强制推广的加密标准,其中包含了生成强随机数的算法.这是一种较新的算法,生成的随机数质量较高.建议选"Y".选中此项后,还需选中至少一个子项.

Enable Hash DRBG

CONFIG_CRYPTO_DRBG_HASH

符合NIST SP800-90A标准的Hash DRBG变种算法(基于SHA256).建议选"Y".

Enable CTR DRBG

CONFIG_CRYPTO_DRBG_CTR

符合 NIST SP800-90A 标准的 CTR DRBG 变种算法(基于 AES).建议选"Y".

Jitterentropy Non-Deterministic Random Number Generator

CONFIG_CRYPTO_JITTERENTROPY

JitterEntropy 伪随机数生成器以 CPU 执行时间的抖动作为熵源,这是一种高质量的熵源,可作为其他随机数生成器(/dev/urandom)的种子.这样就无需在系统启动时从磁盘上加载已经保存的种子文件了.

User-space interface for hash algorithms

CONFIG_CRYPTO_USER_API_HASH

哈希算法的用户空间接口.systemd 依赖于它.

User-space interface for symmetric key cipher algorithms

CONFIG_CRYPTO_USER_API_SKCIPHER

对称加密算法的用户空间接口.不确定的选"N".

User-space interface for random number generator algorithms

CONFIG_CRYPTO_USER_API_RNG

随机数发生器算法的用户空间接口.不确定的选"N".

User-space interface for AEAD cipher algorithms

CONFIG_CRYPTO_USER_API_AEAD

AEAD(Authenticated-Encryption with Addtional Data)加密算法的用户空间接口.AEAD 是当前最好的加密模式.不确定的选"N".

Hardware crypto devices

CONFIG_CRYPTO_HW

硬件加密设备支持

Support for VIA PadLock ACE

CONFIG_CRYPTO_DEV_PADLOCK

带有 PadLock 技术的 VIA 系列处理器支持

PadLock driver for AES algorithm

CONFIG_CRYPTO_DEV_PADLOCK_AES

利用 PadLock 技术加速 AES 运算.VIA C3 及以上的 CPU 都支持.

PadLock driver for SHA1 and SHA256 algorithms

CONFIG_CRYPTO_DEV_PADLOCK_SHA

利用 PadLock 技术加速 SHA1 和 SHA256 运算.VIA C7 及以上的 CPU 都支持.

Support for AMD Cryptographic Coprocessor

CONFIG_CRYPTO_DEV_CCP

AMD 密码协处理器支持.

Support for Intel(R) DH895xCC

CONFIG_CRYPTO_DEV_QAT_DH895xCC

支持 QuickAssist 技术的 Intel DH895xcc 芯片.Skylake 的服务器平台的部分 CPU 型号支持 QuickAssist 技术.

Support for Intel(R) DH895xCC Virtual Function

CONFIG_CRYPTO_DEV_QAT_DH895xCCVF

支持 QuickAssist 技术的 Intel DH895xcc 芯片.Skylake 的服务器平台的部分 CPU 型号支持 QuickAssist 技术.

Asymmetric (public-key cryptographic) key type

CONFIG_ASYMMETRIC_KEY_TYPE

非对称加密算法(公钥加密算法)

Asymmetric public-key crypto algorithm subtype

CONFIG_ASYMMETRIC_PUBLIC_KEY_SUBTYPE

非对称公钥加密算法子类型.如果需要生成或者校验签名,那就还必须配合哈希算法一起使用.

RSA public-key algorithm

CONFIG_PUBLIC_KEY_ALGO_RSA

RSA 算法(PKCS#1, RFC3447)支持

X.509 certificate parser

CONFIG_X509_CERTIFICATE_PARSER

X.509 证书解析支持

Certificates for signature checking

用于检查签名有效性的证书:(1)用于检查内核模块的签名,(2)用于检查全局密钥环(keyring)中的密钥的可靠性.

Virtualization

虚拟化支持

仅在将此内核用作宿主机(host)的情况下才需要开启这里的子项

Kernel-based Virtual Machine (KVM) support

CONFIG_KVM

KVM(内核虚拟机)是一种基于 Linux 内核的全虚拟化技术,需要 CPU 支持 x86 硬件虚拟化技术(Intel VT 或 AMD-V).开启此项后,将可以通过字符文件"/dev/kvm"使用虚拟机.

KVM for Intel processors support

CONFIG_KVM_INTEL

Intel VT 技术支持.也就是 cpu-flags 中有"vmx"标记.

KVM for AMD processors support

CONFIG_KVM_AMD

AMD-V 技术支持.也就是 cpu-flags 中有"svm"标记.

Audit KVM MMU

CONFIG_KVM_MMU_AUDIT

添加一个"kvm.mmu_audit"内核参数,用于控制是否允许在运行时对 KVM MMU 进行审计."0"表示禁止审计,"1"表示允许审计.主要用于调试目的.不确定的选"N".

KVM legacy PCI device assignment support

CONFIG_KVM_DEVICE_ASSIGNMENT

通过 KVM 支持传统的 PCI 设备分配.内核目前还通过 VFIO(CONFIG_VFIO)支持一个全功能的用户空间设备驱动框架,可以取代这里的功能.不确定的选"N".

Host kernel accelerator for virtio net

CONFIG_VHOST_NET

在宿主机内核中开启此项后,可以加速客户机的网络操作速度(客户机内核需要加载 virtio_net 模块(CONFIG_VIRTIO_NET)).建议选"Y".

VHOST_SCSI TCM fabric driver

CONFIG_VHOST_SCSI

Say M here to enable the vhost_scsi TCM fabric module for use with virtio-scsi guests.看不懂的选"N".

Cross-endian support for vhost

CONFIG_VHOST_CROSS_ENDIAN_LEGACY

允许宿主机支持不同大小端顺序的客户机中老旧的 virtio 设备.仅在宿主机与客户机的大小端顺序不一致的场合(ppc64/arm64)才有意义.不确定的选"N".

Library routines

库子程序

子选项可以全不选,内核中若有其他部分依赖它,会自动选上.使用内核树外的模块时可能需要手动选择.

CRC-CCITT functions

CONFIG_CRC_CCITT

为内核树外的模块提供 CRC-CCITT 循环验证算法支持.

CRC16 functions

CONFIG_CRC16

为内核树外的模块提供 CRC16 循环验证算法支持.

CRC calculation for the T10 Data Integrity Field

CONFIG_CRC_T10DIF

为内核树外的模块提供 CRC 循环验证算法支持.从而允许内核树外的 SCSI 模块利用 T10/SCSI Data Integrity Field 保障端到端的数据完整性.

CRC ITU-T V.41 functions

CONFIG_CRC_ITU_T

为内核树外的模块提供 CRC ITU-T V.41 循环验证算法支持.

CRC32/CRC32c functions

CONFIG_CRC32

为内核树外的模块提供 CRC32/CRC32c 循环验证算法支持.

CRC32 perform self test on init

CONFIG_CRC32_SELFTEST

在 CRC32 算法初始化的时候进行一个简单的自我测试.不确定的选"N".

CRC32 implementation

选择 CRC32 算法的实现方式.不确定的请保持默认值"Slice by 8 bytes",除非你知道自己在做什么.

CRC7 functions

CONFIG_CRC7

为内核树外的模块提供 CRC7 循环验证算法支持.

CRC32c (Castagnoli, et al) Cyclic Redundancy-Check

CONFIG_LIBCRC32C

为内核树外的模块提供 CRC32c 循环验证算法支持.

CRC8 function

CONFIG_CRC8

为内核树外的模块提供 CRC8 循环验证算法支持.

XZ decompression support

CONFIG_XZ_DEC

为内核树外的模块提供 XZ 解压支持.详见"Documentation/xz.txt"文档.子项是针对不同平台的"BCJ filter decoder".按需选择即可.

XZ decompressor tester

CONFIG_XZ_DEC_TEST

XZ 解压测试程序.不确定的选"N".

Averaging functions

CONFIG_AVERAGE

为内核树外的模块提供 averaging 函数支持

CORDIC algorithm

CONFIG_CORDIC

为内核树外的模块提供 CORDIC 算法支持

JEDEC DDR data

CONFIG_DDR

为内核树外的 DDR SDRAM 内存控制器驱动提供获取 JEDEC 数据支持.

Select compiled-in fonts

CONFIG_FONTS

选择内嵌到内核中的字体(点阵字库,仅包含 ASCII 字符和扩展 ASCII 字符,共 256 个).选"N"表示内嵌自动选择的默认字体,选"Y"表示可以手动选择内嵌的字体.[提示]可到 drivers/video/console 目录下找到相应的"font_*.c"文件,将其中的"0"全部替换为空格,即可看到点阵字符.

VGA 8x8 font

CONFIG_FONT_8x8

这是传统上高分辨率(高于 80x50)下使用的字体.因为点阵太小,所以显示的字体质量非常低劣.

VGA 8x16 font

CONFIG_FONT_8x16

这是传统上的标准字体(用于 80x25),也是默认内嵌的字体,最为常见.

{其它字体省略}

console 16x16 CJK font (cover BMP)

CONFIG_FONT_16x16_CJK

CJKTTY 开源项目提供的中文字体支持(仅支持 UTF-8 字符)补丁.如果你希望能够直接在控制台上显示中文,那么请将此项选"Y",并同时将其他字体选项全部选"N".如何改造 Linux 虚拟终端显示中文一文讲解了此补丁的原理.