**Планировщик EDF**

**Текст программы**

Санкт - Петербург

2017

Листинг 1: Файл sched\_casio.patch

diff -Nur linux-2.6.24/arch/x86/Kconfig linux-2.6.24-casio/arch/x86/Kconfig

--- linux-2.6.24/arch/x86/Kconfig 2008-01-24 22:58:37.000000000 +0000

+++ linux-2.6.24-casio/arch/x86/Kconfig 2009-06-02 04:11:04.000000000 +0100

@@ -1603,6 +1603,13 @@

endmenu

+menu "CASIO scheduler"

+

+config SCHED\_CASIO\_POLICY

+ bool "CASIO scheduling policy"

+ default y

+endmenu

+

source "net/Kconfig"

source "drivers/Kconfig"

diff -Nur linux-2.6.24/.config linux-2.6.24-casio/.config

--- linux-2.6.24/.config 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/.config 2009-06-02 04:11:04.000000000 +0100

@@ -0,0 +1,3896 @@

+#

+# Automatically generated make config: don't edit

+# Linux kernel version: 2.6.24-casio

+# Thu May 7 14:45:09 2009

+#

+# CONFIG\_64BIT is not set

+CONFIG\_X86\_32=y

+# CONFIG\_X86\_64 is not set

+CONFIG\_X86=y

+CONFIG\_GENERIC\_TIME=y

+CONFIG\_GENERIC\_CMOS\_UPDATE=y

+CONFIG\_CLOCKSOURCE\_WATCHDOG=y

+CONFIG\_GENERIC\_CLOCKEVENTS=y

+CONFIG\_GENERIC\_CLOCKEVENTS\_BROADCAST=y

+CONFIG\_LOCKDEP\_SUPPORT=y

+CONFIG\_STACKTRACE\_SUPPORT=y

+CONFIG\_SEMAPHORE\_SLEEPERS=y

+CONFIG\_MMU=y

+CONFIG\_ZONE\_DMA=y

+CONFIG\_QUICKLIST=y

+CONFIG\_GENERIC\_ISA\_DMA=y

+CONFIG\_GENERIC\_IOMAP=y

+CONFIG\_GENERIC\_BUG=y

+CONFIG\_GENERIC\_HWEIGHT=y

+CONFIG\_ARCH\_MAY\_HAVE\_PC\_FDC=y

+CONFIG\_DMI=y

+# CONFIG\_RWSEM\_GENERIC\_SPINLOCK is not set

+CONFIG\_RWSEM\_XCHGADD\_ALGORITHM=y

+# CONFIG\_ARCH\_HAS\_ILOG2\_U32 is not set

+# CONFIG\_ARCH\_HAS\_ILOG2\_U64 is not set

+CONFIG\_GENERIC\_CALIBRATE\_DELAY=y

+# CONFIG\_GENERIC\_TIME\_VSYSCALL is not set

+CONFIG\_ARCH\_SUPPORTS\_OPROFILE=y

+# CONFIG\_ZONE\_DMA32 is not set

+CONFIG\_ARCH\_POPULATES\_NODE\_MAP=y

+# CONFIG\_AUDIT\_ARCH is not set

+CONFIG\_GENERIC\_HARDIRQS=y

+CONFIG\_GENERIC\_IRQ\_PROBE=y

+CONFIG\_GENERIC\_PENDING\_IRQ=y

+CONFIG\_X86\_SMP=y

+CONFIG\_X86\_HT=y

+CONFIG\_X86\_BIOS\_REBOOT=y

+CONFIG\_X86\_TRAMPOLINE=y

+CONFIG\_KTIME\_SCALAR=y

+CONFIG\_DEFCONFIG\_LIST="/lib/modules/$UNAME\_RELEASE/.config"

+

+#

+# General setup

+#

+CONFIG\_EXPERIMENTAL=y

+CONFIG\_LOCK\_KERNEL=y

+CONFIG\_INIT\_ENV\_ARG\_LIMIT=32

+CONFIG\_LOCALVERSION=""

+# CONFIG\_LOCALVERSION\_AUTO is not set

+CONFIG\_SWAP=y

+CONFIG\_SYSVIPC=y

+CONFIG\_SYSVIPC\_SYSCTL=y

+CONFIG\_POSIX\_MQUEUE=y

+CONFIG\_BSD\_PROCESS\_ACCT=y

+CONFIG\_BSD\_PROCESS\_ACCT\_V3=y

+CONFIG\_TASKSTATS=y

+# CONFIG\_TASK\_DELAY\_ACCT is not set

+CONFIG\_TASK\_XACCT=y

+CONFIG\_TASK\_IO\_ACCOUNTING=y

+# CONFIG\_USER\_NS is not set

+# CONFIG\_PID\_NS is not set

+CONFIG\_AUDIT=y

+CONFIG\_AUDITSYSCALL=y

+CONFIG\_AUDIT\_TREE=y

+# CONFIG\_IKCONFIG is not set

+CONFIG\_LOG\_BUF\_SHIFT=17

+CONFIG\_CGROUPS=y

+# CONFIG\_CGROUP\_DEBUG is not set

+CONFIG\_CGROUP\_NS=y

+CONFIG\_CPUSETS=y

+CONFIG\_FAIR\_GROUP\_SCHED=y

+# CONFIG\_FAIR\_USER\_SCHED is not set

+CONFIG\_FAIR\_CGROUP\_SCHED=y

+CONFIG\_CGROUP\_CPUACCT=y

+# CONFIG\_SYSFS\_DEPRECATED is not set

+CONFIG\_PROC\_PID\_CPUSET=y

+CONFIG\_RELAY=y

+CONFIG\_BLK\_DEV\_INITRD=y

+CONFIG\_INITRAMFS\_SOURCE=""

+# CONFIG\_CC\_OPTIMIZE\_FOR\_SIZE is not set

+CONFIG\_SYSCTL=y

+CONFIG\_EMBEDDED=y

+CONFIG\_UID16=y

+CONFIG\_SYSCTL\_SYSCALL=y

+CONFIG\_KALLSYMS=y

+CONFIG\_KALLSYMS\_ALL=y

+# CONFIG\_KALLSYMS\_EXTRA\_PASS is not set

+CONFIG\_HOTPLUG=y

+CONFIG\_PRINTK=y

+CONFIG\_BUG=y

+CONFIG\_ELF\_CORE=y

+CONFIG\_BASE\_FULL=y

+CONFIG\_FUTEX=y

+CONFIG\_ANON\_INODES=y

+CONFIG\_EPOLL=y

+CONFIG\_SIGNALFD=y

+CONFIG\_EVENTFD=y

+CONFIG\_SHMEM=y

+CONFIG\_VM\_EVENT\_COUNTERS=y

+CONFIG\_SLUB\_DEBUG=y

+# CONFIG\_SLAB is not set

+CONFIG\_SLUB=y

+# CONFIG\_SLOB is not set

+CONFIG\_SLABINFO=y

+CONFIG\_RT\_MUTEXES=y

+# CONFIG\_TINY\_SHMEM is not set

+CONFIG\_BASE\_SMALL=0

+CONFIG\_MODULES=y

+CONFIG\_MODULE\_UNLOAD=y

+# CONFIG\_MODULE\_FORCE\_UNLOAD is not set

+CONFIG\_MODVERSIONS=y

+CONFIG\_MODULE\_SRCVERSION\_ALL=y

+CONFIG\_KMOD=y

+CONFIG\_STOP\_MACHINE=y

+CONFIG\_BLOCK=y

+CONFIG\_LBD=y

+CONFIG\_BLK\_DEV\_IO\_TRACE=y

+# CONFIG\_LSF is not set

+# CONFIG\_BLK\_DEV\_BSG is not set

+

+#

+# IO Schedulers

+#

+CONFIG\_IOSCHED\_NOOP=y

+CONFIG\_IOSCHED\_AS=y

+CONFIG\_IOSCHED\_DEADLINE=y

+CONFIG\_IOSCHED\_CFQ=y

+# CONFIG\_DEFAULT\_AS is not set

+# CONFIG\_DEFAULT\_DEADLINE is not set

+CONFIG\_DEFAULT\_CFQ=y

+# CONFIG\_DEFAULT\_NOOP is not set

+CONFIG\_DEFAULT\_IOSCHED="cfq"

+CONFIG\_PREEMPT\_NOTIFIERS=y

+

+#

+# Processor type and features

+#

+CONFIG\_TICK\_ONESHOT=y

+CONFIG\_NO\_HZ=y

+CONFIG\_HIGH\_RES\_TIMERS=y

+CONFIG\_GENERIC\_CLOCKEVENTS\_BUILD=y

+CONFIG\_SMP=y

+CONFIG\_X86\_PC=y

+# CONFIG\_X86\_ELAN is not set

+# CONFIG\_X86\_VOYAGER is not set

+# CONFIG\_X86\_NUMAQ is not set

+# CONFIG\_X86\_SUMMIT is not set

+# CONFIG\_X86\_BIGSMP is not set

+# CONFIG\_X86\_VISWS is not set

+# CONFIG\_X86\_GENERICARCH is not set

+# CONFIG\_X86\_ES7000 is not set

+# CONFIG\_X86\_VSMP is not set

+CONFIG\_SCHED\_NO\_NO\_OMIT\_FRAME\_POINTER=y

+CONFIG\_PARAVIRT=y

+CONFIG\_PARAVIRT\_GUEST=y

+CONFIG\_VMI=y

+# CONFIG\_LGUEST\_GUEST is not set

+# CONFIG\_M386 is not set

+# CONFIG\_M486 is not set

+CONFIG\_M586=y

+# CONFIG\_M586TSC is not set

+# CONFIG\_M586MMX is not set

+# CONFIG\_M686 is not set

+# CONFIG\_MPENTIUMII is not set

+# CONFIG\_MPENTIUMIII is not set

+# CONFIG\_MPENTIUMM is not set

+# CONFIG\_MPENTIUM4 is not set

+# CONFIG\_MK6 is not set

+# CONFIG\_MK7 is not set

+# CONFIG\_MK8 is not set

+# CONFIG\_MCRUSOE is not set

+# CONFIG\_MEFFICEON is not set

+# CONFIG\_MWINCHIPC6 is not set

+# CONFIG\_MWINCHIP2 is not set

+# CONFIG\_MWINCHIP3D is not set

+# CONFIG\_MGEODEGX1 is not set

+# CONFIG\_MGEODE\_LX is not set

+# CONFIG\_MCYRIXIII is not set

+# CONFIG\_MVIAC3\_2 is not set

+# CONFIG\_MVIAC7 is not set

+# CONFIG\_MPSC is not set

+# CONFIG\_MCORE2 is not set

+# CONFIG\_GENERIC\_CPU is not set

+CONFIG\_X86\_GENERIC=y

+CONFIG\_X86\_CMPXCHG=y

+CONFIG\_X86\_L1\_CACHE\_SHIFT=7

+CONFIG\_X86\_XADD=y

+CONFIG\_X86\_PPRO\_FENCE=y

+CONFIG\_X86\_F00F\_BUG=y

+CONFIG\_X86\_WP\_WORKS\_OK=y

+CONFIG\_X86\_INVLPG=y

+CONFIG\_X86\_BSWAP=y

+CONFIG\_X86\_POPAD\_OK=y

+CONFIG\_X86\_ALIGNMENT\_16=y

+CONFIG\_X86\_INTEL\_USERCOPY=y

+CONFIG\_X86\_MINIMUM\_CPU\_FAMILY=4

+CONFIG\_HPET\_TIMER=y

+CONFIG\_HPET\_EMULATE\_RTC=y

+CONFIG\_NR\_CPUS=8

+CONFIG\_SCHED\_SMT=y

+CONFIG\_SCHED\_MC=y

+# CONFIG\_PREEMPT\_NONE is not set

+CONFIG\_PREEMPT\_VOLUNTARY=y

+# CONFIG\_PREEMPT is not set

+CONFIG\_PREEMPT\_BKL=y

+CONFIG\_X86\_LOCAL\_APIC=y

+CONFIG\_X86\_IO\_APIC=y

+# CONFIG\_X86\_MCE is not set

+CONFIG\_VM86=y

+CONFIG\_TOSHIBA=m

+CONFIG\_I8K=m

+CONFIG\_X86\_REBOOTFIXUPS=y

+CONFIG\_MICROCODE=m

+CONFIG\_MICROCODE\_OLD\_INTERFACE=y

+CONFIG\_X86\_MSR=m

+CONFIG\_X86\_CPUID=m

+# CONFIG\_NOHIGHMEM is not set

+CONFIG\_HIGHMEM4G=y

+# CONFIG\_HIGHMEM64G is not set

+CONFIG\_VMSPLIT\_3G=y

+# CONFIG\_VMSPLIT\_3G\_OPT is not set

+# CONFIG\_VMSPLIT\_2G is not set

+# CONFIG\_VMSPLIT\_2G\_OPT is not set

+# CONFIG\_VMSPLIT\_1G is not set

+CONFIG\_PAGE\_OFFSET=0xC0000000

+CONFIG\_HIGHMEM=y

+CONFIG\_ARCH\_FLATMEM\_ENABLE=y

+CONFIG\_ARCH\_SPARSEMEM\_ENABLE=y

+CONFIG\_ARCH\_SELECT\_MEMORY\_MODEL=y

+CONFIG\_SELECT\_MEMORY\_MODEL=y

+CONFIG\_FLATMEM\_MANUAL=y

+# CONFIG\_DISCONTIGMEM\_MANUAL is not set

+# CONFIG\_SPARSEMEM\_MANUAL is not set

+CONFIG\_FLATMEM=y

+CONFIG\_FLAT\_NODE\_MEM\_MAP=y

+CONFIG\_SPARSEMEM\_STATIC=y

+# CONFIG\_SPARSEMEM\_VMEMMAP\_ENABLE is not set

+CONFIG\_SPLIT\_PTLOCK\_CPUS=4

+# CONFIG\_RESOURCES\_64BIT is not set

+CONFIG\_ZONE\_DMA\_FLAG=1

+CONFIG\_BOUNCE=y

+CONFIG\_NR\_QUICK=1

+CONFIG\_VIRT\_TO\_BUS=y

+CONFIG\_HIGHPTE=y

+# CONFIG\_MATH\_EMULATION is not set

+CONFIG\_MTRR=y

+CONFIG\_EFI=y

+# CONFIG\_IRQBALANCE is not set

+CONFIG\_BOOT\_IOREMAP=y

+CONFIG\_SECCOMP=y

+# CONFIG\_HZ\_100 is not set

+CONFIG\_HZ\_250=y

+# CONFIG\_HZ\_300 is not set

+# CONFIG\_HZ\_1000 is not set

+CONFIG\_HZ=250

+CONFIG\_KEXEC=y

+CONFIG\_CRASH\_DUMP=y

+CONFIG\_PHYSICAL\_START=0x100000

+CONFIG\_RELOCATABLE=y

+CONFIG\_PHYSICAL\_ALIGN=0x100000

+CONFIG\_HOTPLUG\_CPU=y

+# CONFIG\_COMPAT\_VDSO is not set

+CONFIG\_ARCH\_ENABLE\_MEMORY\_HOTPLUG=y

+

+#

+# Power management options

+#

+CONFIG\_PM=y

+CONFIG\_PM\_LEGACY=y

+CONFIG\_PM\_DEBUG=y

+# CONFIG\_PM\_VERBOSE is not set

+CONFIG\_PM\_TRACE=y

+CONFIG\_PM\_SLEEP\_SMP=y

+CONFIG\_PM\_SLEEP=y

+CONFIG\_SUSPEND\_SMP\_POSSIBLE=y

+CONFIG\_SUSPEND=y

+CONFIG\_HIBERNATION\_SMP\_POSSIBLE=y

+CONFIG\_HIBERNATION=y

+CONFIG\_PM\_STD\_PARTITION=""

+CONFIG\_ACPI=y

+CONFIG\_ACPI\_SLEEP=y

+CONFIG\_ACPI\_PROCFS=y

+CONFIG\_ACPI\_PROCFS\_POWER=y

+CONFIG\_ACPI\_SYSFS\_POWER=y

+CONFIG\_ACPI\_PROC\_EVENT=y

+CONFIG\_ACPI\_AC=m

+CONFIG\_ACPI\_BATTERY=m

+CONFIG\_ACPI\_BUTTON=m

+CONFIG\_ACPI\_VIDEO=m

+CONFIG\_ACPI\_FAN=m

+CONFIG\_ACPI\_DOCK=m

+CONFIG\_ACPI\_BAY=m

+CONFIG\_ACPI\_PROCESSOR=m

+CONFIG\_ACPI\_HOTPLUG\_CPU=y

+CONFIG\_ACPI\_THERMAL=m

+CONFIG\_ACPI\_ASUS=m

+CONFIG\_ACPI\_TOSHIBA=m

+CONFIG\_ACPI\_BLACKLIST\_YEAR=2000

+# CONFIG\_ACPI\_DEBUG is not set

+CONFIG\_ACPI\_EC=y

+CONFIG\_ACPI\_POWER=y

+CONFIG\_ACPI\_SYSTEM=y

+CONFIG\_X86\_PM\_TIMER=y

+CONFIG\_ACPI\_CONTAINER=m

+CONFIG\_ACPI\_SBS=m

+CONFIG\_APM=m

+# CONFIG\_APM\_IGNORE\_USER\_SUSPEND is not set

+# CONFIG\_APM\_DO\_ENABLE is not set

+# CONFIG\_APM\_CPU\_IDLE is not set

+# CONFIG\_APM\_DISPLAY\_BLANK is not set

+# CONFIG\_APM\_ALLOW\_INTS is not set

+# CONFIG\_APM\_REAL\_MODE\_POWER\_OFF is not set

+

+#

+# CPU Frequency scaling

+#

+CONFIG\_CPU\_FREQ=y

+CONFIG\_CPU\_FREQ\_TABLE=m

+# CONFIG\_CPU\_FREQ\_DEBUG is not set

+CONFIG\_CPU\_FREQ\_STAT=m

+CONFIG\_CPU\_FREQ\_STAT\_DETAILS=y

+CONFIG\_CPU\_FREQ\_DEFAULT\_GOV\_PERFORMANCE=y

+# CONFIG\_CPU\_FREQ\_DEFAULT\_GOV\_USERSPACE is not set

+# CONFIG\_CPU\_FREQ\_DEFAULT\_GOV\_ONDEMAND is not set

+# CONFIG\_CPU\_FREQ\_DEFAULT\_GOV\_CONSERVATIVE is not set

+CONFIG\_CPU\_FREQ\_GOV\_PERFORMANCE=y

+CONFIG\_CPU\_FREQ\_GOV\_POWERSAVE=m

+CONFIG\_CPU\_FREQ\_GOV\_USERSPACE=m

+CONFIG\_CPU\_FREQ\_GOV\_ONDEMAND=m

+CONFIG\_CPU\_FREQ\_GOV\_CONSERVATIVE=m

+

+#

+# CPUFreq processor drivers

+#

+CONFIG\_X86\_ACPI\_CPUFREQ=m

+CONFIG\_X86\_POWERNOW\_K6=m

+CONFIG\_X86\_POWERNOW\_K7=m

+CONFIG\_X86\_POWERNOW\_K7\_ACPI=y

+CONFIG\_X86\_POWERNOW\_K8=m

+CONFIG\_X86\_POWERNOW\_K8\_ACPI=y

+CONFIG\_X86\_GX\_SUSPMOD=m

+CONFIG\_X86\_SPEEDSTEP\_CENTRINO=m

+CONFIG\_X86\_SPEEDSTEP\_CENTRINO\_TABLE=y

+CONFIG\_X86\_SPEEDSTEP\_ICH=m

+CONFIG\_X86\_SPEEDSTEP\_SMI=m

+CONFIG\_X86\_P4\_CLOCKMOD=m

+CONFIG\_X86\_CPUFREQ\_NFORCE2=m

+CONFIG\_X86\_LONGRUN=m

+CONFIG\_X86\_LONGHAUL=m

+# CONFIG\_X86\_E\_POWERSAVER is not set

+

+#

+# shared options

+#

+# CONFIG\_X86\_ACPI\_CPUFREQ\_PROC\_INTF is not set

+CONFIG\_X86\_SPEEDSTEP\_LIB=m

+CONFIG\_X86\_SPEEDSTEP\_RELAXED\_CAP\_CHECK=y

+CONFIG\_CPU\_IDLE=y

+CONFIG\_CPU\_IDLE\_GOV\_LADDER=y

+CONFIG\_CPU\_IDLE\_GOV\_MENU=y

+

+#

+# Bus options (PCI etc.)

+#

+CONFIG\_PCI=y

+# CONFIG\_PCI\_GOBIOS is not set

+# CONFIG\_PCI\_GOMMCONFIG is not set

+# CONFIG\_PCI\_GODIRECT is not set

+CONFIG\_PCI\_GOANY=y

+CONFIG\_PCI\_BIOS=y

+CONFIG\_PCI\_DIRECT=y

+CONFIG\_PCI\_MMCONFIG=y

+CONFIG\_PCI\_DOMAINS=y

+CONFIG\_PCIEPORTBUS=y

+CONFIG\_HOTPLUG\_PCI\_PCIE=m

+CONFIG\_PCIEAER=y

+CONFIG\_ARCH\_SUPPORTS\_MSI=y

+CONFIG\_PCI\_MSI=y

+CONFIG\_PCI\_LEGACY=y

+# CONFIG\_PCI\_DEBUG is not set

+CONFIG\_HT\_IRQ=y

+CONFIG\_ISA\_DMA\_API=y

+CONFIG\_ISA=y

+CONFIG\_EISA=y

+CONFIG\_EISA\_VLB\_PRIMING=y

+CONFIG\_EISA\_PCI\_EISA=y

+CONFIG\_EISA\_VIRTUAL\_ROOT=y

+CONFIG\_EISA\_NAMES=y

+CONFIG\_MCA=y

+CONFIG\_MCA\_LEGACY=y

+# CONFIG\_MCA\_PROC\_FS is not set

+CONFIG\_SCx200=m

+CONFIG\_SCx200HR\_TIMER=m

+CONFIG\_K8\_NB=y

+CONFIG\_PCCARD=m

+# CONFIG\_PCMCIA\_DEBUG is not set

+CONFIG\_PCMCIA=m

+CONFIG\_PCMCIA\_LOAD\_CIS=y

+CONFIG\_PCMCIA\_IOCTL=y

+CONFIG\_CARDBUS=y

+

+#

+# PC-card bridges

+#

+CONFIG\_YENTA=m

+CONFIG\_YENTA\_O2=y

+CONFIG\_YENTA\_RICOH=y

+CONFIG\_YENTA\_TI=y

+CONFIG\_YENTA\_ENE\_TUNE=y

+CONFIG\_YENTA\_TOSHIBA=y

+CONFIG\_PD6729=m

+CONFIG\_I82092=m

+CONFIG\_I82365=m

+CONFIG\_TCIC=m

+CONFIG\_PCMCIA\_PROBE=y

+CONFIG\_PCCARD\_NONSTATIC=m

+CONFIG\_HOTPLUG\_PCI=m

+CONFIG\_HOTPLUG\_PCI\_FAKE=m

+CONFIG\_HOTPLUG\_PCI\_COMPAQ=m

+CONFIG\_HOTPLUG\_PCI\_COMPAQ\_NVRAM=y

+CONFIG\_HOTPLUG\_PCI\_IBM=m

+CONFIG\_HOTPLUG\_PCI\_ACPI=m

+CONFIG\_HOTPLUG\_PCI\_ACPI\_IBM=m

+CONFIG\_HOTPLUG\_PCI\_CPCI=y

+CONFIG\_HOTPLUG\_PCI\_CPCI\_ZT5550=m

+CONFIG\_HOTPLUG\_PCI\_CPCI\_GENERIC=m

+CONFIG\_HOTPLUG\_PCI\_SHPC=m

+

+#

+# Executable file formats / Emulations

+#

+CONFIG\_BINFMT\_ELF=y

+CONFIG\_BINFMT\_AOUT=m

+CONFIG\_BINFMT\_MISC=m

+

+#

+# CASIO scheduler

+#

+CONFIG\_SCHED\_CASIO\_POLICY=y

+

+#

+# Networking

+#

+CONFIG\_NET=y

+

+#

+# Networking options

+#

+CONFIG\_PACKET=m

+CONFIG\_PACKET\_MMAP=y

+CONFIG\_UNIX=y

+CONFIG\_XFRM=y

+CONFIG\_XFRM\_USER=m

+# CONFIG\_XFRM\_SUB\_POLICY is not set

+# CONFIG\_XFRM\_MIGRATE is not set

+CONFIG\_NET\_KEY=m

+# CONFIG\_NET\_KEY\_MIGRATE is not set

+CONFIG\_INET=y

+CONFIG\_IP\_MULTICAST=y

+CONFIG\_IP\_ADVANCED\_ROUTER=y

+CONFIG\_ASK\_IP\_FIB\_HASH=y

+# CONFIG\_IP\_FIB\_TRIE is not set

+CONFIG\_IP\_FIB\_HASH=y

+CONFIG\_IP\_MULTIPLE\_TABLES=y

+CONFIG\_IP\_ROUTE\_MULTIPATH=y

+CONFIG\_IP\_ROUTE\_VERBOSE=y

+# CONFIG\_IP\_PNP is not set

+CONFIG\_NET\_IPIP=m

+CONFIG\_NET\_IPGRE=m

+CONFIG\_NET\_IPGRE\_BROADCAST=y

+CONFIG\_IP\_MROUTE=y

+CONFIG\_IP\_PIMSM\_V1=y

+CONFIG\_IP\_PIMSM\_V2=y

+# CONFIG\_ARPD is not set

+CONFIG\_SYN\_COOKIES=y

+CONFIG\_INET\_AH=m

+CONFIG\_INET\_ESP=m

+CONFIG\_INET\_IPCOMP=m

+CONFIG\_INET\_XFRM\_TUNNEL=m

+CONFIG\_INET\_TUNNEL=m

+CONFIG\_INET\_XFRM\_MODE\_TRANSPORT=m

+CONFIG\_INET\_XFRM\_MODE\_TUNNEL=m

+CONFIG\_INET\_XFRM\_MODE\_BEET=m

+CONFIG\_INET\_LRO=m

+CONFIG\_INET\_DIAG=y

+CONFIG\_INET\_TCP\_DIAG=y

+CONFIG\_TCP\_CONG\_ADVANCED=y

+CONFIG\_TCP\_CONG\_BIC=m

+CONFIG\_TCP\_CONG\_CUBIC=m

+CONFIG\_TCP\_CONG\_WESTWOOD=m

+CONFIG\_TCP\_CONG\_HTCP=m

+CONFIG\_TCP\_CONG\_HSTCP=m

+CONFIG\_TCP\_CONG\_HYBLA=m

+CONFIG\_TCP\_CONG\_VEGAS=m

+CONFIG\_TCP\_CONG\_SCALABLE=m

+CONFIG\_TCP\_CONG\_LP=m

+CONFIG\_TCP\_CONG\_VENO=m

+CONFIG\_TCP\_CONG\_YEAH=m

+CONFIG\_TCP\_CONG\_ILLINOIS=m

+# CONFIG\_DEFAULT\_BIC is not set

+# CONFIG\_DEFAULT\_CUBIC is not set

+# CONFIG\_DEFAULT\_HTCP is not set

+# CONFIG\_DEFAULT\_VEGAS is not set

+# CONFIG\_DEFAULT\_WESTWOOD is not set

+CONFIG\_DEFAULT\_RENO=y

+CONFIG\_DEFAULT\_TCP\_CONG="reno"

+CONFIG\_TCP\_MD5SIG=y

+CONFIG\_IP\_VS=m

+# CONFIG\_IP\_VS\_DEBUG is not set

+CONFIG\_IP\_VS\_TAB\_BITS=12

+

+#

+# IPVS transport protocol load balancing support

+#

+CONFIG\_IP\_VS\_PROTO\_TCP=y

+CONFIG\_IP\_VS\_PROTO\_UDP=y

+CONFIG\_IP\_VS\_PROTO\_ESP=y

+CONFIG\_IP\_VS\_PROTO\_AH=y

+

+#

+# IPVS scheduler

+#

+CONFIG\_IP\_VS\_RR=m

+CONFIG\_IP\_VS\_WRR=m

+CONFIG\_IP\_VS\_LC=m

+CONFIG\_IP\_VS\_WLC=m

+CONFIG\_IP\_VS\_LBLC=m

+CONFIG\_IP\_VS\_LBLCR=m

+CONFIG\_IP\_VS\_DH=m

+CONFIG\_IP\_VS\_SH=m

+CONFIG\_IP\_VS\_SED=m

+CONFIG\_IP\_VS\_NQ=m

+

+#

+# IPVS application helper

+#

+CONFIG\_IP\_VS\_FTP=m

+CONFIG\_IPV6=m

+CONFIG\_IPV6\_PRIVACY=y

+# CONFIG\_IPV6\_ROUTER\_PREF is not set

+# CONFIG\_IPV6\_OPTIMISTIC\_DAD is not set

+CONFIG\_INET6\_AH=m

+CONFIG\_INET6\_ESP=m

+CONFIG\_INET6\_IPCOMP=m

+# CONFIG\_IPV6\_MIP6 is not set

+CONFIG\_INET6\_XFRM\_TUNNEL=m

+CONFIG\_INET6\_TUNNEL=m

+CONFIG\_INET6\_XFRM\_MODE\_TRANSPORT=m

+CONFIG\_INET6\_XFRM\_MODE\_TUNNEL=m

+CONFIG\_INET6\_XFRM\_MODE\_BEET=m

+CONFIG\_INET6\_XFRM\_MODE\_ROUTEOPTIMIZATION=m

+CONFIG\_IPV6\_SIT=m

+CONFIG\_IPV6\_TUNNEL=m

+# CONFIG\_IPV6\_MULTIPLE\_TABLES is not set

+# CONFIG\_NETLABEL is not set

+CONFIG\_NETWORK\_SECMARK=y

+CONFIG\_NETFILTER=y

+# CONFIG\_NETFILTER\_DEBUG is not set

+CONFIG\_BRIDGE\_NETFILTER=y

+

+#

+# Core Netfilter Configuration

+#

+CONFIG\_NETFILTER\_NETLINK=m

+CONFIG\_NETFILTER\_NETLINK\_QUEUE=m

+CONFIG\_NETFILTER\_NETLINK\_LOG=m

+CONFIG\_NF\_CONNTRACK\_ENABLED=m

+CONFIG\_NF\_CONNTRACK=m

+CONFIG\_NF\_CT\_ACCT=y

+CONFIG\_NF\_CONNTRACK\_MARK=y

+CONFIG\_NF\_CONNTRACK\_SECMARK=y

+CONFIG\_NF\_CONNTRACK\_EVENTS=y

+CONFIG\_NF\_CT\_PROTO\_GRE=m

+CONFIG\_NF\_CT\_PROTO\_SCTP=m

+CONFIG\_NF\_CT\_PROTO\_UDPLITE=m

+CONFIG\_NF\_CONNTRACK\_AMANDA=m

+CONFIG\_NF\_CONNTRACK\_FTP=m

+CONFIG\_NF\_CONNTRACK\_H323=m

+CONFIG\_NF\_CONNTRACK\_IRC=m

+CONFIG\_NF\_CONNTRACK\_NETBIOS\_NS=m

+CONFIG\_NF\_CONNTRACK\_PPTP=m

+# CONFIG\_NF\_CONNTRACK\_SANE is not set

+CONFIG\_NF\_CONNTRACK\_SIP=m

+CONFIG\_NF\_CONNTRACK\_TFTP=m

+CONFIG\_NF\_CT\_NETLINK=m

+CONFIG\_NETFILTER\_XTABLES=m

+CONFIG\_NETFILTER\_XT\_TARGET\_CLASSIFY=m

+CONFIG\_NETFILTER\_XT\_TARGET\_CONNMARK=m

+CONFIG\_NETFILTER\_XT\_TARGET\_DSCP=m

+CONFIG\_NETFILTER\_XT\_TARGET\_MARK=m

+CONFIG\_NETFILTER\_XT\_TARGET\_NFQUEUE=m

+CONFIG\_NETFILTER\_XT\_TARGET\_NFLOG=m

+CONFIG\_NETFILTER\_XT\_TARGET\_NOTRACK=m

+CONFIG\_NETFILTER\_XT\_TARGET\_TRACE=m

+CONFIG\_NETFILTER\_XT\_TARGET\_SECMARK=m

+CONFIG\_NETFILTER\_XT\_TARGET\_CONNSECMARK=m

+CONFIG\_NETFILTER\_XT\_TARGET\_TCPMSS=m

+CONFIG\_NETFILTER\_XT\_MATCH\_COMMENT=m

+CONFIG\_NETFILTER\_XT\_MATCH\_CONNBYTES=m

+CONFIG\_NETFILTER\_XT\_MATCH\_CONNLIMIT=m

+CONFIG\_NETFILTER\_XT\_MATCH\_CONNMARK=m

+CONFIG\_NETFILTER\_XT\_MATCH\_CONNTRACK=m

+CONFIG\_NETFILTER\_XT\_MATCH\_DCCP=m

+CONFIG\_NETFILTER\_XT\_MATCH\_DSCP=m

+CONFIG\_NETFILTER\_XT\_MATCH\_ESP=m

+CONFIG\_NETFILTER\_XT\_MATCH\_HELPER=m

+CONFIG\_NETFILTER\_XT\_MATCH\_LENGTH=m

+CONFIG\_NETFILTER\_XT\_MATCH\_LIMIT=m

+CONFIG\_NETFILTER\_XT\_MATCH\_MAC=m

+CONFIG\_NETFILTER\_XT\_MATCH\_MARK=m

+CONFIG\_NETFILTER\_XT\_MATCH\_POLICY=m

+CONFIG\_NETFILTER\_XT\_MATCH\_MULTIPORT=m

+CONFIG\_NETFILTER\_XT\_MATCH\_PHYSDEV=m

+CONFIG\_NETFILTER\_XT\_MATCH\_PKTTYPE=m

+CONFIG\_NETFILTER\_XT\_MATCH\_QUOTA=m

+CONFIG\_NETFILTER\_XT\_MATCH\_REALM=m

+CONFIG\_NETFILTER\_XT\_MATCH\_SCTP=m

+CONFIG\_NETFILTER\_XT\_MATCH\_STATE=m

+CONFIG\_NETFILTER\_XT\_MATCH\_STATISTIC=m

+CONFIG\_NETFILTER\_XT\_MATCH\_STRING=m

+CONFIG\_NETFILTER\_XT\_MATCH\_TCPMSS=m

+CONFIG\_NETFILTER\_XT\_MATCH\_TIME=m

+CONFIG\_NETFILTER\_XT\_MATCH\_U32=m

+CONFIG\_NETFILTER\_XT\_MATCH\_HASHLIMIT=m

+

+#

+# IP: Netfilter Configuration

+#

+CONFIG\_NF\_CONNTRACK\_IPV4=m

+CONFIG\_NF\_CONNTRACK\_PROC\_COMPAT=y

+CONFIG\_IP\_NF\_QUEUE=m

+CONFIG\_IP\_NF\_IPTABLES=m

+CONFIG\_IP\_NF\_MATCH\_IPRANGE=m

+CONFIG\_IP\_NF\_MATCH\_TOS=m

+CONFIG\_IP\_NF\_MATCH\_RECENT=m

+CONFIG\_IP\_NF\_MATCH\_ECN=m

+CONFIG\_IP\_NF\_MATCH\_AH=m

+CONFIG\_IP\_NF\_MATCH\_TTL=m

+CONFIG\_IP\_NF\_MATCH\_OWNER=m

+CONFIG\_IP\_NF\_MATCH\_ADDRTYPE=m

+CONFIG\_IP\_NF\_FILTER=m

+CONFIG\_IP\_NF\_TARGET\_REJECT=m

+CONFIG\_IP\_NF\_TARGET\_LOG=m

+CONFIG\_IP\_NF\_TARGET\_ULOG=m

+CONFIG\_NF\_NAT=m

+CONFIG\_NF\_NAT\_NEEDED=y

+CONFIG\_IP\_NF\_TARGET\_MASQUERADE=m

+CONFIG\_IP\_NF\_TARGET\_REDIRECT=m

+CONFIG\_IP\_NF\_TARGET\_NETMAP=m

+CONFIG\_IP\_NF\_TARGET\_SAME=m

+CONFIG\_NF\_NAT\_SNMP\_BASIC=m

+CONFIG\_NF\_NAT\_PROTO\_GRE=m

+CONFIG\_NF\_NAT\_FTP=m

+CONFIG\_NF\_NAT\_IRC=m

+CONFIG\_NF\_NAT\_TFTP=m

+CONFIG\_NF\_NAT\_AMANDA=m

+CONFIG\_NF\_NAT\_PPTP=m

+CONFIG\_NF\_NAT\_H323=m

+CONFIG\_NF\_NAT\_SIP=m

+CONFIG\_IP\_NF\_MANGLE=m

+CONFIG\_IP\_NF\_TARGET\_TOS=m

+CONFIG\_IP\_NF\_TARGET\_ECN=m

+CONFIG\_IP\_NF\_TARGET\_TTL=m

+CONFIG\_IP\_NF\_TARGET\_CLUSTERIP=m

+CONFIG\_IP\_NF\_RAW=m

+CONFIG\_IP\_NF\_ARPTABLES=m

+CONFIG\_IP\_NF\_ARPFILTER=m

+CONFIG\_IP\_NF\_ARP\_MANGLE=m

+

+#

+# IPv6: Netfilter Configuration (EXPERIMENTAL)

+#

+CONFIG\_NF\_CONNTRACK\_IPV6=m

+CONFIG\_IP6\_NF\_QUEUE=m

+CONFIG\_IP6\_NF\_IPTABLES=m

+CONFIG\_IP6\_NF\_MATCH\_RT=m

+CONFIG\_IP6\_NF\_MATCH\_OPTS=m

+CONFIG\_IP6\_NF\_MATCH\_FRAG=m

+CONFIG\_IP6\_NF\_MATCH\_HL=m

+CONFIG\_IP6\_NF\_MATCH\_OWNER=m

+CONFIG\_IP6\_NF\_MATCH\_IPV6HEADER=m

+CONFIG\_IP6\_NF\_MATCH\_AH=m

+CONFIG\_IP6\_NF\_MATCH\_MH=m

+CONFIG\_IP6\_NF\_MATCH\_EUI64=m

+CONFIG\_IP6\_NF\_FILTER=m

+CONFIG\_IP6\_NF\_TARGET\_LOG=m

+CONFIG\_IP6\_NF\_TARGET\_REJECT=m

+CONFIG\_IP6\_NF\_MANGLE=m

+CONFIG\_IP6\_NF\_TARGET\_HL=m

+CONFIG\_IP6\_NF\_RAW=m

+

+#

+# DECnet: Netfilter Configuration

+#

+CONFIG\_DECNET\_NF\_GRABULATOR=m

+

+#

+# Bridge: Netfilter Configuration

+#

+CONFIG\_BRIDGE\_NF\_EBTABLES=m

+CONFIG\_BRIDGE\_EBT\_BROUTE=m

+CONFIG\_BRIDGE\_EBT\_T\_FILTER=m

+CONFIG\_BRIDGE\_EBT\_T\_NAT=m

+CONFIG\_BRIDGE\_EBT\_802\_3=m

+CONFIG\_BRIDGE\_EBT\_AMONG=m

+CONFIG\_BRIDGE\_EBT\_ARP=m

+CONFIG\_BRIDGE\_EBT\_IP=m

+CONFIG\_BRIDGE\_EBT\_LIMIT=m

+CONFIG\_BRIDGE\_EBT\_MARK=m

+CONFIG\_BRIDGE\_EBT\_PKTTYPE=m

+CONFIG\_BRIDGE\_EBT\_STP=m

+CONFIG\_BRIDGE\_EBT\_VLAN=m

+CONFIG\_BRIDGE\_EBT\_ARPREPLY=m

+CONFIG\_BRIDGE\_EBT\_DNAT=m

+CONFIG\_BRIDGE\_EBT\_MARK\_T=m

+CONFIG\_BRIDGE\_EBT\_REDIRECT=m

+CONFIG\_BRIDGE\_EBT\_SNAT=m

+CONFIG\_BRIDGE\_EBT\_LOG=m

+CONFIG\_BRIDGE\_EBT\_ULOG=m

+CONFIG\_IP\_DCCP=m

+CONFIG\_INET\_DCCP\_DIAG=m

+CONFIG\_IP\_DCCP\_ACKVEC=y

+

+#

+# DCCP CCIDs Configuration (EXPERIMENTAL)

+#

+CONFIG\_IP\_DCCP\_CCID2=m

+# CONFIG\_IP\_DCCP\_CCID2\_DEBUG is not set

+CONFIG\_IP\_DCCP\_CCID3=m

+CONFIG\_IP\_DCCP\_TFRC\_LIB=m

+# CONFIG\_IP\_DCCP\_CCID3\_DEBUG is not set

+CONFIG\_IP\_DCCP\_CCID3\_RTO=100

+

+#

+# DCCP Kernel Hacking

+#

+# CONFIG\_IP\_DCCP\_DEBUG is not set

+CONFIG\_NET\_DCCPPROBE=m

+CONFIG\_IP\_SCTP=m

+# CONFIG\_SCTP\_DBG\_MSG is not set

+# CONFIG\_SCTP\_DBG\_OBJCNT is not set

+# CONFIG\_SCTP\_HMAC\_NONE is not set

+# CONFIG\_SCTP\_HMAC\_SHA1 is not set

+CONFIG\_SCTP\_HMAC\_MD5=y

+CONFIG\_TIPC=m

+# CONFIG\_TIPC\_ADVANCED is not set

+# CONFIG\_TIPC\_DEBUG is not set

+CONFIG\_ATM=y

+CONFIG\_ATM\_CLIP=y

+# CONFIG\_ATM\_CLIP\_NO\_ICMP is not set

+CONFIG\_ATM\_LANE=m

+CONFIG\_ATM\_MPOA=m

+CONFIG\_ATM\_BR2684=m

+# CONFIG\_ATM\_BR2684\_IPFILTER is not set

+CONFIG\_BRIDGE=m

+CONFIG\_VLAN\_8021Q=m

+CONFIG\_DECNET=m

+# CONFIG\_DECNET\_ROUTER is not set

+CONFIG\_LLC=y

+CONFIG\_LLC2=m

+CONFIG\_IPX=m

+# CONFIG\_IPX\_INTERN is not set

+CONFIG\_ATALK=m

+CONFIG\_DEV\_APPLETALK=m

+CONFIG\_LTPC=m

+CONFIG\_COPS=m

+CONFIG\_COPS\_DAYNA=y

+CONFIG\_COPS\_TANGENT=y

+CONFIG\_IPDDP=m

+CONFIG\_IPDDP\_ENCAP=y

+CONFIG\_IPDDP\_DECAP=y

+CONFIG\_X25=m

+CONFIG\_LAPB=m

+CONFIG\_ECONET=m

+CONFIG\_ECONET\_AUNUDP=y

+CONFIG\_ECONET\_NATIVE=y

+CONFIG\_WAN\_ROUTER=m

+CONFIG\_NET\_SCHED=y

+

+#

+# Queueing/Scheduling

+#

+CONFIG\_NET\_SCH\_CBQ=m

+CONFIG\_NET\_SCH\_HTB=m

+CONFIG\_NET\_SCH\_HFSC=m

+CONFIG\_NET\_SCH\_ATM=m

+CONFIG\_NET\_SCH\_PRIO=m

+CONFIG\_NET\_SCH\_RR=m

+CONFIG\_NET\_SCH\_RED=m

+CONFIG\_NET\_SCH\_SFQ=m

+CONFIG\_NET\_SCH\_TEQL=m

+CONFIG\_NET\_SCH\_TBF=m

+CONFIG\_NET\_SCH\_GRED=m

+CONFIG\_NET\_SCH\_DSMARK=m

+CONFIG\_NET\_SCH\_NETEM=m

+CONFIG\_NET\_SCH\_INGRESS=m

+

+#

+# Classification

+#

+CONFIG\_NET\_CLS=y

+CONFIG\_NET\_CLS\_BASIC=m

+CONFIG\_NET\_CLS\_TCINDEX=m

+CONFIG\_NET\_CLS\_ROUTE4=m

+CONFIG\_NET\_CLS\_ROUTE=y

+CONFIG\_NET\_CLS\_FW=m

+CONFIG\_NET\_CLS\_U32=m

+# CONFIG\_CLS\_U32\_PERF is not set

+CONFIG\_CLS\_U32\_MARK=y

+CONFIG\_NET\_CLS\_RSVP=m

+CONFIG\_NET\_CLS\_RSVP6=m

+CONFIG\_NET\_EMATCH=y

+CONFIG\_NET\_EMATCH\_STACK=32

+CONFIG\_NET\_EMATCH\_CMP=m

+CONFIG\_NET\_EMATCH\_NBYTE=m

+CONFIG\_NET\_EMATCH\_U32=m

+CONFIG\_NET\_EMATCH\_META=m

+CONFIG\_NET\_EMATCH\_TEXT=m

+CONFIG\_NET\_CLS\_ACT=y

+CONFIG\_NET\_ACT\_POLICE=m

+CONFIG\_NET\_ACT\_GACT=m

+CONFIG\_GACT\_PROB=y

+CONFIG\_NET\_ACT\_MIRRED=m

+CONFIG\_NET\_ACT\_IPT=m

+CONFIG\_NET\_ACT\_NAT=m

+CONFIG\_NET\_ACT\_PEDIT=m

+CONFIG\_NET\_ACT\_SIMP=m

+# CONFIG\_NET\_CLS\_POLICE is not set

+# CONFIG\_NET\_CLS\_IND is not set

+CONFIG\_NET\_SCH\_FIFO=y

+

+#

+# Network testing

+#

+CONFIG\_NET\_PKTGEN=m

+CONFIG\_NET\_TCPPROBE=m

+CONFIG\_HAMRADIO=y

+

+#

+# Packet Radio protocols

+#

+CONFIG\_AX25=m

+CONFIG\_AX25\_DAMA\_SLAVE=y

+CONFIG\_NETROM=m

+CONFIG\_ROSE=m

+

+#

+# AX.25 network device drivers

+#

+CONFIG\_MKISS=m

+CONFIG\_6PACK=m

+CONFIG\_BPQETHER=m

+CONFIG\_SCC=m

+# CONFIG\_SCC\_DELAY is not set

+# CONFIG\_SCC\_TRXECHO is not set

+CONFIG\_BAYCOM\_SER\_FDX=m

+CONFIG\_BAYCOM\_SER\_HDX=m

+CONFIG\_BAYCOM\_PAR=m

+CONFIG\_BAYCOM\_EPP=m

+CONFIG\_YAM=m

+CONFIG\_IRDA=m

+

+#

+# IrDA protocols

+#

+CONFIG\_IRLAN=m

+CONFIG\_IRNET=m

+CONFIG\_IRCOMM=m

+CONFIG\_IRDA\_ULTRA=y

+

+#

+# IrDA options

+#

+CONFIG\_IRDA\_CACHE\_LAST\_LSAP=y

+CONFIG\_IRDA\_FAST\_RR=y

+CONFIG\_IRDA\_DEBUG=y

+

+#

+# Infrared-port device drivers

+#

+

+#

+# SIR device drivers

+#

+CONFIG\_IRTTY\_SIR=m

+

+#

+# Dongle support

+#

+CONFIG\_DONGLE=y

+CONFIG\_ESI\_DONGLE=m

+CONFIG\_ACTISYS\_DONGLE=m

+CONFIG\_TEKRAM\_DONGLE=m

+# CONFIG\_TOIM3232\_DONGLE is not set

+CONFIG\_LITELINK\_DONGLE=m

+CONFIG\_MA600\_DONGLE=m

+CONFIG\_GIRBIL\_DONGLE=m

+CONFIG\_MCP2120\_DONGLE=m

+CONFIG\_OLD\_BELKIN\_DONGLE=m

+CONFIG\_ACT200L\_DONGLE=m

+CONFIG\_KINGSUN\_DONGLE=m

+CONFIG\_KSDAZZLE\_DONGLE=m

+CONFIG\_KS959\_DONGLE=m

+

+#

+# Old SIR device drivers

+#

+

+#

+# Old Serial dongle support

+#

+

+#

+# FIR device drivers

+#

+CONFIG\_USB\_IRDA=m

+CONFIG\_SIGMATEL\_FIR=m

+CONFIG\_NSC\_FIR=m

+CONFIG\_WINBOND\_FIR=m

+CONFIG\_TOSHIBA\_FIR=m

+CONFIG\_SMC\_IRCC\_FIR=m

+CONFIG\_ALI\_FIR=m

+CONFIG\_VLSI\_FIR=m

+CONFIG\_VIA\_FIR=m

+CONFIG\_MCS\_FIR=m

+CONFIG\_BT=m

+CONFIG\_BT\_L2CAP=m

+CONFIG\_BT\_SCO=m

+CONFIG\_BT\_RFCOMM=m

+CONFIG\_BT\_RFCOMM\_TTY=y

+CONFIG\_BT\_BNEP=m

+CONFIG\_BT\_BNEP\_MC\_FILTER=y

+CONFIG\_BT\_BNEP\_PROTO\_FILTER=y

+CONFIG\_BT\_CMTP=m

+CONFIG\_BT\_HIDP=m

+

+#

+# Bluetooth device drivers

+#

+CONFIG\_BT\_HCIUSB=m

+CONFIG\_BT\_HCIUSB\_SCO=y

+CONFIG\_BT\_HCIBTSDIO=m

+CONFIG\_BT\_HCIUART=m

+CONFIG\_BT\_HCIUART\_H4=y

+CONFIG\_BT\_HCIUART\_BCSP=y

+CONFIG\_BT\_HCIUART\_LL=y

+CONFIG\_BT\_HCIBCM203X=m

+CONFIG\_BT\_HCIBPA10X=m

+CONFIG\_BT\_HCIBFUSB=m

+CONFIG\_BT\_HCIDTL1=m

+CONFIG\_BT\_HCIBT3C=m

+CONFIG\_BT\_HCIBLUECARD=m

+CONFIG\_BT\_HCIBTUART=m

+CONFIG\_BT\_HCIVHCI=m

+CONFIG\_AF\_RXRPC=m

+# CONFIG\_AF\_RXRPC\_DEBUG is not set

+CONFIG\_RXKAD=m

+CONFIG\_FIB\_RULES=y

+

+#

+# Wireless

+#

+CONFIG\_CFG80211=m

+CONFIG\_NL80211=y

+CONFIG\_WIRELESS\_EXT=y

+CONFIG\_MAC80211=m

+CONFIG\_MAC80211\_RCSIMPLE=y

+CONFIG\_MAC80211\_LEDS=y

+CONFIG\_MAC80211\_DEBUGFS=y

+# CONFIG\_MAC80211\_DEBUG is not set

+CONFIG\_IEEE80211=m

+# CONFIG\_IEEE80211\_DEBUG is not set

+CONFIG\_IEEE80211\_CRYPT\_WEP=m

+CONFIG\_IEEE80211\_CRYPT\_CCMP=m

+CONFIG\_IEEE80211\_CRYPT\_TKIP=m

+CONFIG\_IEEE80211\_SOFTMAC=m

+# CONFIG\_IEEE80211\_SOFTMAC\_DEBUG is not set

+CONFIG\_RFKILL=m

+CONFIG\_RFKILL\_INPUT=m

+CONFIG\_RFKILL\_LEDS=y

+CONFIG\_NET\_9P=m

+CONFIG\_NET\_9P\_FD=m

+# CONFIG\_NET\_9P\_DEBUG is not set

+

+#

+# Device Drivers

+#

+

+#

+# Generic Driver Options

+#

+CONFIG\_UEVENT\_HELPER\_PATH="/sbin/hotplug"

+CONFIG\_STANDALONE=y

+CONFIG\_PREVENT\_FIRMWARE\_BUILD=y

+CONFIG\_FW\_LOADER=y

+# CONFIG\_DEBUG\_DRIVER is not set

+# CONFIG\_DEBUG\_DEVRES is not set

+# CONFIG\_SYS\_HYPERVISOR is not set

+CONFIG\_CONNECTOR=m

+CONFIG\_MTD=m

+# CONFIG\_MTD\_DEBUG is not set

+CONFIG\_MTD\_CONCAT=m

+CONFIG\_MTD\_PARTITIONS=y

+CONFIG\_MTD\_REDBOOT\_PARTS=m

+CONFIG\_MTD\_REDBOOT\_DIRECTORY\_BLOCK=-1

+# CONFIG\_MTD\_REDBOOT\_PARTS\_UNALLOCATED is not set

+# CONFIG\_MTD\_REDBOOT\_PARTS\_READONLY is not set

+

+#

+# User Modules And Translation Layers

+#

+CONFIG\_MTD\_CHAR=m

+CONFIG\_MTD\_BLKDEVS=m

+CONFIG\_MTD\_BLOCK=m

+CONFIG\_MTD\_BLOCK\_RO=m

+CONFIG\_FTL=m

+CONFIG\_NFTL=m

+CONFIG\_NFTL\_RW=y

+CONFIG\_INFTL=m

+CONFIG\_RFD\_FTL=m

+CONFIG\_SSFDC=m

+CONFIG\_MTD\_OOPS=m

+

+#

+# RAM/ROM/Flash chip drivers

+#

+CONFIG\_MTD\_CFI=m

+CONFIG\_MTD\_JEDECPROBE=m

+CONFIG\_MTD\_GEN\_PROBE=m

+# CONFIG\_MTD\_CFI\_ADV\_OPTIONS is not set

+CONFIG\_MTD\_MAP\_BANK\_WIDTH\_1=y

+CONFIG\_MTD\_MAP\_BANK\_WIDTH\_2=y

+CONFIG\_MTD\_MAP\_BANK\_WIDTH\_4=y

+# CONFIG\_MTD\_MAP\_BANK\_WIDTH\_8 is not set

+# CONFIG\_MTD\_MAP\_BANK\_WIDTH\_16 is not set

+# CONFIG\_MTD\_MAP\_BANK\_WIDTH\_32 is not set

+CONFIG\_MTD\_CFI\_I1=y

+CONFIG\_MTD\_CFI\_I2=y

+# CONFIG\_MTD\_CFI\_I4 is not set

+# CONFIG\_MTD\_CFI\_I8 is not set

+CONFIG\_MTD\_CFI\_INTELEXT=m

+CONFIG\_MTD\_CFI\_AMDSTD=m

+CONFIG\_MTD\_CFI\_STAA=m

+CONFIG\_MTD\_CFI\_UTIL=m

+CONFIG\_MTD\_RAM=m

+CONFIG\_MTD\_ROM=m

+CONFIG\_MTD\_ABSENT=m

+

+#

+# Mapping drivers for chip access

+#

+CONFIG\_MTD\_COMPLEX\_MAPPINGS=y

+CONFIG\_MTD\_PHYSMAP=m

+CONFIG\_MTD\_PHYSMAP\_START=0x8000000

+CONFIG\_MTD\_PHYSMAP\_LEN=0x4000000

+CONFIG\_MTD\_PHYSMAP\_BANKWIDTH=2

+CONFIG\_MTD\_PNC2000=m

+CONFIG\_MTD\_SC520CDP=m

+CONFIG\_MTD\_NETSC520=m

+CONFIG\_MTD\_TS5500=m

+CONFIG\_MTD\_SBC\_GXX=m

+CONFIG\_MTD\_SCx200\_DOCFLASH=m

+CONFIG\_MTD\_AMD76XROM=m

+CONFIG\_MTD\_ICHXROM=m

+CONFIG\_MTD\_ESB2ROM=m

+CONFIG\_MTD\_CK804XROM=m

+CONFIG\_MTD\_SCB2\_FLASH=m

+CONFIG\_MTD\_NETtel=m

+CONFIG\_MTD\_DILNETPC=m

+CONFIG\_MTD\_DILNETPC\_BOOTSIZE=0x80000

+CONFIG\_MTD\_L440GX=m

+CONFIG\_MTD\_PCI=m

+CONFIG\_MTD\_INTEL\_VR\_NOR=m

+CONFIG\_MTD\_PLATRAM=m

+

+#

+# Self-contained MTD device drivers

+#

+CONFIG\_MTD\_PMC551=m

+# CONFIG\_MTD\_PMC551\_BUGFIX is not set

+# CONFIG\_MTD\_PMC551\_DEBUG is not set

+CONFIG\_MTD\_DATAFLASH=m

+CONFIG\_MTD\_M25P80=m

+CONFIG\_MTD\_SLRAM=m

+CONFIG\_MTD\_PHRAM=m

+CONFIG\_MTD\_MTDRAM=m

+CONFIG\_MTDRAM\_TOTAL\_SIZE=4096

+CONFIG\_MTDRAM\_ERASE\_SIZE=128

+CONFIG\_MTD\_BLOCK2MTD=m

+

+#

+# Disk-On-Chip Device Drivers

+#

+CONFIG\_MTD\_DOC2000=m

+CONFIG\_MTD\_DOC2001=m

+CONFIG\_MTD\_DOC2001PLUS=m

+CONFIG\_MTD\_DOCPROBE=m

+CONFIG\_MTD\_DOCECC=m

+# CONFIG\_MTD\_DOCPROBE\_ADVANCED is not set

+CONFIG\_MTD\_DOCPROBE\_ADDRESS=0

+CONFIG\_MTD\_NAND=m

+# CONFIG\_MTD\_NAND\_VERIFY\_WRITE is not set

+# CONFIG\_MTD\_NAND\_ECC\_SMC is not set

+# CONFIG\_MTD\_NAND\_MUSEUM\_IDS is not set

+CONFIG\_MTD\_NAND\_IDS=m

+CONFIG\_MTD\_NAND\_DISKONCHIP=m

+# CONFIG\_MTD\_NAND\_DISKONCHIP\_PROBE\_ADVANCED is not set

+CONFIG\_MTD\_NAND\_DISKONCHIP\_PROBE\_ADDRESS=0

+# CONFIG\_MTD\_NAND\_DISKONCHIP\_BBTWRITE is not set

+CONFIG\_MTD\_NAND\_CAFE=m

+CONFIG\_MTD\_NAND\_CS553X=m

+CONFIG\_MTD\_NAND\_NANDSIM=m

+CONFIG\_MTD\_NAND\_PLATFORM=m

+CONFIG\_MTD\_ALAUDA=m

+CONFIG\_MTD\_ONENAND=m

+CONFIG\_MTD\_ONENAND\_VERIFY\_WRITE=y

+# CONFIG\_MTD\_ONENAND\_OTP is not set

+CONFIG\_MTD\_ONENAND\_2X\_PROGRAM=y

+CONFIG\_MTD\_ONENAND\_SIM=m

+

+#

+# UBI - Unsorted block images

+#

+CONFIG\_MTD\_UBI=m

+CONFIG\_MTD\_UBI\_WL\_THRESHOLD=4096

+CONFIG\_MTD\_UBI\_BEB\_RESERVE=1

+CONFIG\_MTD\_UBI\_GLUEBI=y

+

+#

+# UBI debugging options

+#

+# CONFIG\_MTD\_UBI\_DEBUG is not set

+CONFIG\_PARPORT=m

+CONFIG\_PARPORT\_PC=m

+CONFIG\_PARPORT\_SERIAL=m

+CONFIG\_PARPORT\_PC\_FIFO=y

+# CONFIG\_PARPORT\_PC\_SUPERIO is not set

+CONFIG\_PARPORT\_PC\_PCMCIA=m

+# CONFIG\_PARPORT\_GSC is not set

+CONFIG\_PARPORT\_AX88796=m

+CONFIG\_PARPORT\_1284=y

+CONFIG\_PARPORT\_NOT\_PC=y

+CONFIG\_PNP=y

+# CONFIG\_PNP\_DEBUG is not set

+

+#

+# Protocols

+#

+CONFIG\_ISAPNP=y

+CONFIG\_PNPBIOS=y

+CONFIG\_PNPBIOS\_PROC\_FS=y

+CONFIG\_PNPACPI=y

+CONFIG\_BLK\_DEV=y

+CONFIG\_BLK\_DEV\_FD=m

+CONFIG\_BLK\_DEV\_XD=m

+CONFIG\_PARIDE=m

+

+#

+# Parallel IDE high-level drivers

+#

+CONFIG\_PARIDE\_PD=m

+CONFIG\_PARIDE\_PCD=m

+CONFIG\_PARIDE\_PF=m

+CONFIG\_PARIDE\_PT=m

+CONFIG\_PARIDE\_PG=m

+

+#

+# Parallel IDE protocol modules

+#

+CONFIG\_PARIDE\_ATEN=m

+CONFIG\_PARIDE\_BPCK=m

+CONFIG\_PARIDE\_BPCK6=m

+CONFIG\_PARIDE\_COMM=m

+CONFIG\_PARIDE\_DSTR=m

+CONFIG\_PARIDE\_FIT2=m

+CONFIG\_PARIDE\_FIT3=m

+CONFIG\_PARIDE\_EPAT=m

+# CONFIG\_PARIDE\_EPATC8 is not set

+CONFIG\_PARIDE\_EPIA=m

+CONFIG\_PARIDE\_FRIQ=m

+CONFIG\_PARIDE\_FRPW=m

+CONFIG\_PARIDE\_KBIC=m

+CONFIG\_PARIDE\_KTTI=m

+CONFIG\_PARIDE\_ON20=m

+CONFIG\_PARIDE\_ON26=m

+CONFIG\_BLK\_CPQ\_DA=m

+CONFIG\_BLK\_CPQ\_CISS\_DA=m

+CONFIG\_CISS\_SCSI\_TAPE=y

+CONFIG\_BLK\_DEV\_DAC960=m

+CONFIG\_BLK\_DEV\_UMEM=m

+# CONFIG\_BLK\_DEV\_COW\_COMMON is not set

+CONFIG\_BLK\_DEV\_LOOP=m

+CONFIG\_BLK\_DEV\_CRYPTOLOOP=m

+CONFIG\_BLK\_DEV\_NBD=m

+CONFIG\_BLK\_DEV\_SX8=m

+# CONFIG\_BLK\_DEV\_UB is not set

+CONFIG\_BLK\_DEV\_RAM=y

+CONFIG\_BLK\_DEV\_RAM\_COUNT=16

+CONFIG\_BLK\_DEV\_RAM\_SIZE=65536

+CONFIG\_BLK\_DEV\_RAM\_BLOCKSIZE=1024

+CONFIG\_CDROM\_PKTCDVD=m

+CONFIG\_CDROM\_PKTCDVD\_BUFFERS=8

+# CONFIG\_CDROM\_PKTCDVD\_WCACHE is not set

+CONFIG\_ATA\_OVER\_ETH=m

+CONFIG\_MISC\_DEVICES=y

+CONFIG\_IBM\_ASM=m

+CONFIG\_PHANTOM=m

+CONFIG\_EEPROM\_93CX6=m

+CONFIG\_SGI\_IOC4=m

+CONFIG\_TIFM\_CORE=m

+CONFIG\_TIFM\_7XX1=m

+CONFIG\_ASUS\_LAPTOP=m

+CONFIG\_FUJITSU\_LAPTOP=m

+CONFIG\_MSI\_LAPTOP=m

+CONFIG\_SONY\_LAPTOP=m

+CONFIG\_SONYPI\_COMPAT=y

+CONFIG\_THINKPAD\_ACPI=m

+# CONFIG\_THINKPAD\_ACPI\_DEBUG is not set

+CONFIG\_THINKPAD\_ACPI\_BAY=y

+CONFIG\_IDE=y

+CONFIG\_IDE\_MAX\_HWIFS=4

+CONFIG\_BLK\_DEV\_IDE=m

+

+#

+# Please see Documentation/ide.txt for help/info on IDE drives

+#

+# CONFIG\_BLK\_DEV\_IDE\_SATA is not set

+# CONFIG\_BLK\_DEV\_HD\_IDE is not set

+CONFIG\_BLK\_DEV\_IDEDISK=m

+# CONFIG\_IDEDISK\_MULTI\_MODE is not set

+# CONFIG\_BLK\_DEV\_IDECS is not set

+CONFIG\_BLK\_DEV\_DELKIN=m

+CONFIG\_BLK\_DEV\_IDECD=m

+CONFIG\_BLK\_DEV\_IDETAPE=m

+CONFIG\_BLK\_DEV\_IDEFLOPPY=m

+CONFIG\_BLK\_DEV\_IDESCSI=m

+CONFIG\_BLK\_DEV\_IDEACPI=y

+# CONFIG\_IDE\_TASK\_IOCTL is not set

+CONFIG\_IDE\_PROC\_FS=y

+

+#

+# IDE chipset support/bugfixes

+#

+CONFIG\_IDE\_GENERIC=m

+CONFIG\_BLK\_DEV\_PLATFORM=m

+CONFIG\_BLK\_DEV\_CMD640=y

+# CONFIG\_BLK\_DEV\_CMD640\_ENHANCED is not set

+CONFIG\_BLK\_DEV\_IDEPNP=y

+

+#

+# PCI IDE chipsets support

+#

+CONFIG\_BLK\_DEV\_IDEPCI=y

+CONFIG\_IDEPCI\_SHARE\_IRQ=y

+# CONFIG\_IDEPCI\_PCIBUS\_ORDER is not set

+# CONFIG\_BLK\_DEV\_OFFBOARD is not set

+# CONFIG\_BLK\_DEV\_GENERIC is not set

+CONFIG\_BLK\_DEV\_OPTI621=m

+# CONFIG\_BLK\_DEV\_RZ1000 is not set

+CONFIG\_BLK\_DEV\_IDEDMA\_PCI=y

+CONFIG\_BLK\_DEV\_AEC62XX=m

+CONFIG\_BLK\_DEV\_ALI15X3=m

+# CONFIG\_WDC\_ALI15X3 is not set

+# CONFIG\_BLK\_DEV\_AMD74XX is not set

+CONFIG\_BLK\_DEV\_ATIIXP=m

+CONFIG\_BLK\_DEV\_CMD64X=m

+# CONFIG\_BLK\_DEV\_TRIFLEX is not set

+CONFIG\_BLK\_DEV\_CY82C693=m

+# CONFIG\_BLK\_DEV\_CS5520 is not set

+CONFIG\_BLK\_DEV\_CS5530=m

+CONFIG\_BLK\_DEV\_CS5535=m

+CONFIG\_BLK\_DEV\_HPT34X=m

+# CONFIG\_HPT34X\_AUTODMA is not set

+CONFIG\_BLK\_DEV\_HPT366=m

+# CONFIG\_BLK\_DEV\_JMICRON is not set

+CONFIG\_BLK\_DEV\_SC1200=m

+# CONFIG\_BLK\_DEV\_PIIX is not set

+# CONFIG\_BLK\_DEV\_IT8213 is not set

+# CONFIG\_BLK\_DEV\_IT821X is not set

+CONFIG\_BLK\_DEV\_NS87415=m

+CONFIG\_BLK\_DEV\_PDC202XX\_OLD=m

+CONFIG\_PDC202XX\_BURST=y

+# CONFIG\_BLK\_DEV\_PDC202XX\_NEW is not set

+# CONFIG\_BLK\_DEV\_SVWKS is not set

+# CONFIG\_BLK\_DEV\_SIIMAGE is not set

+# CONFIG\_BLK\_DEV\_SIS5513 is not set

+# CONFIG\_BLK\_DEV\_SLC90E66 is not set

+CONFIG\_BLK\_DEV\_TRM290=m

+# CONFIG\_BLK\_DEV\_VIA82CXXX is not set

+CONFIG\_BLK\_DEV\_TC86C001=m

+# CONFIG\_IDE\_ARM is not set

+

+#

+# Other IDE chipsets support

+#

+

+#

+# Note: most of these also require special kernel boot parameters

+#

+CONFIG\_BLK\_DEV\_4DRIVES=y

+CONFIG\_BLK\_DEV\_ALI14XX=m

+CONFIG\_BLK\_DEV\_DTC2278=m

+CONFIG\_BLK\_DEV\_HT6560B=m

+CONFIG\_BLK\_DEV\_QD65XX=m

+CONFIG\_BLK\_DEV\_UMC8672=m

+CONFIG\_BLK\_DEV\_IDEDMA=y

+CONFIG\_IDE\_ARCH\_OBSOLETE\_INIT=y

+# CONFIG\_BLK\_DEV\_HD is not set

+

+#

+# SCSI device support

+#

+CONFIG\_RAID\_ATTRS=m

+CONFIG\_SCSI=m

+CONFIG\_SCSI\_DMA=y

+CONFIG\_SCSI\_TGT=m

+CONFIG\_SCSI\_NETLINK=y

+CONFIG\_SCSI\_PROC\_FS=y

+

+#

+# SCSI support type (disk, tape, CD-ROM)

+#

+CONFIG\_BLK\_DEV\_SD=m

+CONFIG\_CHR\_DEV\_ST=m

+CONFIG\_CHR\_DEV\_OSST=m

+CONFIG\_BLK\_DEV\_SR=m

+# CONFIG\_BLK\_DEV\_SR\_VENDOR is not set

+CONFIG\_CHR\_DEV\_SG=m

+CONFIG\_CHR\_DEV\_SCH=m

+

+#

+# Some SCSI devices (e.g. CD jukebox) support multiple LUNs

+#

+CONFIG\_SCSI\_MULTI\_LUN=y

+CONFIG\_SCSI\_CONSTANTS=y

+CONFIG\_SCSI\_LOGGING=y

+CONFIG\_SCSI\_SCAN\_ASYNC=y

+CONFIG\_SCSI\_WAIT\_SCAN=m

+

+#

+# SCSI Transports

+#

+CONFIG\_SCSI\_SPI\_ATTRS=m

+CONFIG\_SCSI\_FC\_ATTRS=m

+CONFIG\_SCSI\_FC\_TGT\_ATTRS=y

+CONFIG\_SCSI\_ISCSI\_ATTRS=m

+CONFIG\_SCSI\_SAS\_ATTRS=m

+CONFIG\_SCSI\_SAS\_LIBSAS=m

+CONFIG\_SCSI\_SAS\_ATA=y

+# CONFIG\_SCSI\_SAS\_LIBSAS\_DEBUG is not set

+CONFIG\_SCSI\_SRP\_ATTRS=m

+CONFIG\_SCSI\_SRP\_TGT\_ATTRS=y

+CONFIG\_SCSI\_LOWLEVEL=y

+CONFIG\_ISCSI\_TCP=m

+CONFIG\_BLK\_DEV\_3W\_XXXX\_RAID=m

+CONFIG\_SCSI\_3W\_9XXX=m

+CONFIG\_SCSI\_7000FASST=m

+CONFIG\_SCSI\_ACARD=m

+CONFIG\_SCSI\_AHA152X=m

+CONFIG\_SCSI\_AHA1542=m

+CONFIG\_SCSI\_AHA1740=m

+CONFIG\_SCSI\_AACRAID=m

+CONFIG\_SCSI\_AIC7XXX=m

+CONFIG\_AIC7XXX\_CMDS\_PER\_DEVICE=8

+CONFIG\_AIC7XXX\_RESET\_DELAY\_MS=15000

+CONFIG\_AIC7XXX\_DEBUG\_ENABLE=y

+CONFIG\_AIC7XXX\_DEBUG\_MASK=0

+CONFIG\_AIC7XXX\_REG\_PRETTY\_PRINT=y

+# CONFIG\_SCSI\_AIC7XXX\_OLD is not set

+CONFIG\_SCSI\_AIC79XX=m

+CONFIG\_AIC79XX\_CMDS\_PER\_DEVICE=32

+CONFIG\_AIC79XX\_RESET\_DELAY\_MS=15000

+CONFIG\_AIC79XX\_DEBUG\_ENABLE=y

+CONFIG\_AIC79XX\_DEBUG\_MASK=0

+CONFIG\_AIC79XX\_REG\_PRETTY\_PRINT=y

+CONFIG\_SCSI\_AIC94XX=m

+# CONFIG\_AIC94XX\_DEBUG is not set

+CONFIG\_SCSI\_DPT\_I2O=m

+CONFIG\_SCSI\_ADVANSYS=m

+CONFIG\_SCSI\_IN2000=m

+CONFIG\_SCSI\_ARCMSR=m

+CONFIG\_SCSI\_ARCMSR\_AER=y

+CONFIG\_MEGARAID\_NEWGEN=y

+CONFIG\_MEGARAID\_MM=m

+CONFIG\_MEGARAID\_MAILBOX=m

+CONFIG\_MEGARAID\_LEGACY=m

+CONFIG\_MEGARAID\_SAS=m

+CONFIG\_SCSI\_HPTIOP=m

+CONFIG\_SCSI\_BUSLOGIC=m

+# CONFIG\_SCSI\_OMIT\_FLASHPOINT is not set

+CONFIG\_SCSI\_DMX3191D=m

+CONFIG\_SCSI\_DTC3280=m

+CONFIG\_SCSI\_EATA=m

+CONFIG\_SCSI\_EATA\_TAGGED\_QUEUE=y

+CONFIG\_SCSI\_EATA\_LINKED\_COMMANDS=y

+CONFIG\_SCSI\_EATA\_MAX\_TAGS=16

+CONFIG\_SCSI\_FUTURE\_DOMAIN=m

+CONFIG\_SCSI\_FD\_MCS=m

+CONFIG\_SCSI\_GDTH=m

+CONFIG\_SCSI\_GENERIC\_NCR5380=m

+CONFIG\_SCSI\_GENERIC\_NCR5380\_MMIO=m

+CONFIG\_SCSI\_GENERIC\_NCR53C400=y

+CONFIG\_SCSI\_IBMMCA=m

+CONFIG\_IBMMCA\_SCSI\_ORDER\_STANDARD=y

+# CONFIG\_IBMMCA\_SCSI\_DEV\_RESET is not set

+CONFIG\_SCSI\_IPS=m

+CONFIG\_SCSI\_INITIO=m

+CONFIG\_SCSI\_INIA100=m

+CONFIG\_SCSI\_PPA=m

+CONFIG\_SCSI\_IMM=m

+# CONFIG\_SCSI\_IZIP\_EPP16 is not set

+# CONFIG\_SCSI\_IZIP\_SLOW\_CTR is not set

+CONFIG\_SCSI\_NCR53C406A=m

+CONFIG\_SCSI\_NCR\_D700=m

+CONFIG\_SCSI\_STEX=m

+CONFIG\_SCSI\_SYM53C8XX\_2=m

+CONFIG\_SCSI\_SYM53C8XX\_DMA\_ADDRESSING\_MODE=1

+CONFIG\_SCSI\_SYM53C8XX\_DEFAULT\_TAGS=16

+CONFIG\_SCSI\_SYM53C8XX\_MAX\_TAGS=64

+CONFIG\_SCSI\_SYM53C8XX\_MMIO=y

+CONFIG\_SCSI\_IPR=m

+# CONFIG\_SCSI\_IPR\_TRACE is not set

+# CONFIG\_SCSI\_IPR\_DUMP is not set

+CONFIG\_SCSI\_NCR\_Q720=m

+CONFIG\_SCSI\_NCR53C8XX\_DEFAULT\_TAGS=8

+CONFIG\_SCSI\_NCR53C8XX\_MAX\_TAGS=4

+CONFIG\_SCSI\_NCR53C8XX\_SYNC=5

+CONFIG\_SCSI\_PAS16=m

+CONFIG\_SCSI\_PSI240I=m

+CONFIG\_SCSI\_QLOGIC\_FAS=m

+CONFIG\_SCSI\_QLOGIC\_1280=m

+CONFIG\_SCSI\_QLA\_FC=m

+CONFIG\_SCSI\_QLA\_ISCSI=m

+CONFIG\_SCSI\_LPFC=m

+CONFIG\_SCSI\_SEAGATE=m

+CONFIG\_SCSI\_SIM710=m

+CONFIG\_SCSI\_SYM53C416=m

+CONFIG\_SCSI\_DC395x=m

+CONFIG\_SCSI\_DC390T=m

+CONFIG\_SCSI\_T128=m

+CONFIG\_SCSI\_U14\_34F=m

+CONFIG\_SCSI\_U14\_34F\_TAGGED\_QUEUE=y

+CONFIG\_SCSI\_U14\_34F\_LINKED\_COMMANDS=y

+CONFIG\_SCSI\_U14\_34F\_MAX\_TAGS=8

+CONFIG\_SCSI\_ULTRASTOR=m

+CONFIG\_SCSI\_NSP32=m

+CONFIG\_SCSI\_DEBUG=m

+CONFIG\_SCSI\_SRP=m

+CONFIG\_SCSI\_LOWLEVEL\_PCMCIA=y

+CONFIG\_PCMCIA\_AHA152X=m

+CONFIG\_PCMCIA\_FDOMAIN=m

+CONFIG\_PCMCIA\_NINJA\_SCSI=m

+CONFIG\_PCMCIA\_QLOGIC=m

+CONFIG\_PCMCIA\_SYM53C500=m

+CONFIG\_ATA=m

+# CONFIG\_ATA\_NONSTANDARD is not set

+CONFIG\_ATA\_ACPI=y

+CONFIG\_SATA\_AHCI=m

+CONFIG\_SATA\_SVW=m

+CONFIG\_ATA\_PIIX=m

+CONFIG\_SATA\_MV=m

+CONFIG\_SATA\_NV=m

+CONFIG\_PDC\_ADMA=m

+CONFIG\_SATA\_QSTOR=m

+CONFIG\_SATA\_PROMISE=m

+CONFIG\_SATA\_SX4=m

+CONFIG\_SATA\_SIL=m

+CONFIG\_SATA\_SIL24=m

+CONFIG\_SATA\_SIS=m

+CONFIG\_SATA\_ULI=m

+CONFIG\_SATA\_VIA=m

+CONFIG\_SATA\_VITESSE=m

+CONFIG\_SATA\_INIC162X=m

+CONFIG\_PATA\_ACPI=m

+# CONFIG\_PATA\_ALI is not set

+CONFIG\_PATA\_AMD=m

+CONFIG\_PATA\_ARTOP=m

+CONFIG\_PATA\_ATIIXP=m

+# CONFIG\_PATA\_CMD640\_PCI is not set

+CONFIG\_PATA\_CMD64X=m

+CONFIG\_PATA\_CS5520=m

+# CONFIG\_PATA\_CS5530 is not set

+# CONFIG\_PATA\_CS5535 is not set

+CONFIG\_PATA\_CS5536=m

+# CONFIG\_PATA\_CYPRESS is not set

+CONFIG\_PATA\_EFAR=m

+CONFIG\_ATA\_GENERIC=m

+CONFIG\_PATA\_HPT366=m

+# CONFIG\_PATA\_HPT37X is not set

+# CONFIG\_PATA\_HPT3X2N is not set

+CONFIG\_PATA\_HPT3X3=m

+# CONFIG\_PATA\_HPT3X3\_DMA is not set

+# CONFIG\_PATA\_ISAPNP is not set

+CONFIG\_PATA\_IT821X=m

+CONFIG\_PATA\_IT8213=m

+CONFIG\_PATA\_JMICRON=m

+# CONFIG\_PATA\_LEGACY is not set

+CONFIG\_PATA\_TRIFLEX=m

+CONFIG\_PATA\_MARVELL=m

+CONFIG\_PATA\_MPIIX=m

+CONFIG\_PATA\_OLDPIIX=m

+CONFIG\_PATA\_NETCELL=m

+# CONFIG\_PATA\_NS87410 is not set

+# CONFIG\_PATA\_NS87415 is not set

+# CONFIG\_PATA\_OPTI is not set

+# CONFIG\_PATA\_OPTIDMA is not set

+CONFIG\_PATA\_PCMCIA=m

+# CONFIG\_PATA\_PDC\_OLD is not set

+CONFIG\_PATA\_QDI=m

+# CONFIG\_PATA\_RADISYS is not set

+CONFIG\_PATA\_RZ1000=m

+# CONFIG\_PATA\_SC1200 is not set

+CONFIG\_PATA\_SERVERWORKS=m

+CONFIG\_PATA\_PDC2027X=m

+CONFIG\_PATA\_SIL680=m

+CONFIG\_PATA\_SIS=m

+CONFIG\_PATA\_VIA=m

+CONFIG\_PATA\_WINBOND=m

+# CONFIG\_PATA\_WINBOND\_VLB is not set

+CONFIG\_PATA\_PLATFORM=m

+CONFIG\_MD=y

+CONFIG\_BLK\_DEV\_MD=m

+CONFIG\_MD\_LINEAR=m

+CONFIG\_MD\_RAID0=m

+CONFIG\_MD\_RAID1=m

+CONFIG\_MD\_RAID10=m

+CONFIG\_MD\_RAID456=m

+CONFIG\_MD\_RAID5\_RESHAPE=y

+CONFIG\_MD\_MULTIPATH=m

+CONFIG\_MD\_FAULTY=m

+CONFIG\_BLK\_DEV\_DM=m

+# CONFIG\_DM\_DEBUG is not set

+CONFIG\_DM\_CRYPT=m

+CONFIG\_DM\_SNAPSHOT=m

+CONFIG\_DM\_MIRROR=m

+CONFIG\_DM\_ZERO=m

+CONFIG\_DM\_MULTIPATH=m

+CONFIG\_DM\_MULTIPATH\_EMC=m

+CONFIG\_DM\_MULTIPATH\_RDAC=m

+CONFIG\_DM\_MULTIPATH\_HP=m

+# CONFIG\_DM\_DELAY is not set

+CONFIG\_DM\_UEVENT=y

+CONFIG\_FUSION=y

+CONFIG\_FUSION\_SPI=m

+CONFIG\_FUSION\_FC=m

+CONFIG\_FUSION\_SAS=m

+CONFIG\_FUSION\_MAX\_SGE=128

+CONFIG\_FUSION\_CTL=m

+CONFIG\_FUSION\_LAN=m

+CONFIG\_FUSION\_LOGGING=y

+

+#

+# IEEE 1394 (FireWire) support

+#

+# CONFIG\_FIREWIRE is not set

+CONFIG\_IEEE1394=m

+

+#

+# Subsystem Options

+#

+# CONFIG\_IEEE1394\_VERBOSEDEBUG is not set

+

+#

+# Controllers

+#

+CONFIG\_IEEE1394\_PCILYNX=m

+CONFIG\_IEEE1394\_OHCI1394=m

+

+#

+# Protocols

+#

+CONFIG\_IEEE1394\_VIDEO1394=m

+CONFIG\_IEEE1394\_SBP2=m

+# CONFIG\_IEEE1394\_SBP2\_PHYS\_DMA is not set

+CONFIG\_IEEE1394\_ETH1394\_ROM\_ENTRY=y

+CONFIG\_IEEE1394\_ETH1394=m

+CONFIG\_IEEE1394\_DV1394=m

+CONFIG\_IEEE1394\_RAWIO=m

+CONFIG\_I2O=m

+CONFIG\_I2O\_LCT\_NOTIFY\_ON\_CHANGES=y

+CONFIG\_I2O\_EXT\_ADAPTEC=y

+CONFIG\_I2O\_CONFIG=m

+CONFIG\_I2O\_CONFIG\_OLD\_IOCTL=y

+CONFIG\_I2O\_BUS=m

+CONFIG\_I2O\_BLOCK=m

+CONFIG\_I2O\_SCSI=m

+CONFIG\_I2O\_PROC=m

+CONFIG\_MACINTOSH\_DRIVERS=y

+CONFIG\_MAC\_EMUMOUSEBTN=y

+CONFIG\_NETDEVICES=y

+CONFIG\_NETDEVICES\_MULTIQUEUE=y

+CONFIG\_IFB=m

+CONFIG\_DUMMY=m

+CONFIG\_BONDING=m

+CONFIG\_MACVLAN=m

+CONFIG\_EQUALIZER=m

+CONFIG\_TUN=m

+CONFIG\_VETH=m

+CONFIG\_NET\_SB1000=m

+CONFIG\_ARCNET=m

+CONFIG\_ARCNET\_1201=m

+CONFIG\_ARCNET\_1051=m

+CONFIG\_ARCNET\_RAW=m

+CONFIG\_ARCNET\_CAP=m

+CONFIG\_ARCNET\_COM90xx=m

+CONFIG\_ARCNET\_COM90xxIO=m

+CONFIG\_ARCNET\_RIM\_I=m

+CONFIG\_ARCNET\_COM20020=m

+CONFIG\_ARCNET\_COM20020\_ISA=m

+CONFIG\_ARCNET\_COM20020\_PCI=m

+CONFIG\_PHYLIB=m

+

+#

+# MII PHY device drivers

+#

+CONFIG\_MARVELL\_PHY=m

+CONFIG\_DAVICOM\_PHY=m

+CONFIG\_QSEMI\_PHY=m

+CONFIG\_LXT\_PHY=m

+CONFIG\_CICADA\_PHY=m

+CONFIG\_VITESSE\_PHY=m

+CONFIG\_SMSC\_PHY=m

+CONFIG\_BROADCOM\_PHY=m

+CONFIG\_ICPLUS\_PHY=m

+CONFIG\_FIXED\_PHY=m

+# CONFIG\_FIXED\_MII\_10\_FDX is not set

+# CONFIG\_FIXED\_MII\_100\_FDX is not set

+CONFIG\_FIXED\_MII\_1000\_FDX=y

+CONFIG\_FIXED\_MII\_AMNT=1

+CONFIG\_MDIO\_BITBANG=m

+CONFIG\_NET\_ETHERNET=y

+CONFIG\_MII=m

+CONFIG\_HAPPYMEAL=m

+CONFIG\_SUNGEM=m

+CONFIG\_CASSINI=m

+CONFIG\_NET\_VENDOR\_3COM=y

+CONFIG\_EL1=m

+CONFIG\_EL2=m

+CONFIG\_ELPLUS=m

+CONFIG\_EL16=m

+CONFIG\_EL3=m

+CONFIG\_3C515=m

+CONFIG\_ELMC=m

+CONFIG\_ELMC\_II=m

+CONFIG\_VORTEX=m

+CONFIG\_TYPHOON=m

+CONFIG\_LANCE=m

+CONFIG\_NET\_VENDOR\_SMC=y

+CONFIG\_WD80x3=m

+CONFIG\_ULTRAMCA=m

+CONFIG\_ULTRA=m

+CONFIG\_ULTRA32=m

+CONFIG\_SMC9194=m

+CONFIG\_NET\_VENDOR\_RACAL=y

+CONFIG\_NI52=m

+CONFIG\_NI65=m

+CONFIG\_NET\_TULIP=y

+CONFIG\_DE2104X=m

+CONFIG\_TULIP=m

+# CONFIG\_TULIP\_MWI is not set

+# CONFIG\_TULIP\_MMIO is not set

+# CONFIG\_TULIP\_NAPI is not set

+CONFIG\_DE4X5=m

+CONFIG\_WINBOND\_840=m

+CONFIG\_DM9102=m

+CONFIG\_ULI526X=m

+CONFIG\_PCMCIA\_XIRCOM=m

+CONFIG\_AT1700=m

+CONFIG\_DEPCA=m

+CONFIG\_HP100=m

+CONFIG\_NET\_ISA=y

+CONFIG\_E2100=m

+CONFIG\_EWRK3=m

+CONFIG\_EEXPRESS=m

+CONFIG\_EEXPRESS\_PRO=m

+CONFIG\_HPLAN\_PLUS=m

+CONFIG\_HPLAN=m

+CONFIG\_LP486E=m

+CONFIG\_ETH16I=m

+CONFIG\_NE2000=m

+CONFIG\_ZNET=m

+CONFIG\_SEEQ8005=m

+CONFIG\_NE2\_MCA=m

+CONFIG\_IBMLANA=m

+# CONFIG\_IBM\_NEW\_EMAC\_ZMII is not set

+# CONFIG\_IBM\_NEW\_EMAC\_RGMII is not set

+# CONFIG\_IBM\_NEW\_EMAC\_TAH is not set

+# CONFIG\_IBM\_NEW\_EMAC\_EMAC4 is not set

+CONFIG\_NET\_PCI=y

+CONFIG\_PCNET32=m

+# CONFIG\_PCNET32\_NAPI is not set

+CONFIG\_AMD8111\_ETH=m

+# CONFIG\_AMD8111E\_NAPI is not set

+CONFIG\_ADAPTEC\_STARFIRE=m

+# CONFIG\_ADAPTEC\_STARFIRE\_NAPI is not set

+CONFIG\_AC3200=m

+CONFIG\_APRICOT=m

+CONFIG\_B44=m

+CONFIG\_B44\_PCI\_AUTOSELECT=y

+CONFIG\_B44\_PCICORE\_AUTOSELECT=y

+CONFIG\_B44\_PCI=y

+CONFIG\_FORCEDETH=m

+# CONFIG\_FORCEDETH\_NAPI is not set

+CONFIG\_CS89x0=m

+CONFIG\_EEPRO100=m

+CONFIG\_E100=m

+CONFIG\_LNE390=m

+CONFIG\_FEALNX=m

+CONFIG\_NATSEMI=m

+CONFIG\_NE2K\_PCI=m

+CONFIG\_NE3210=m

+CONFIG\_ES3210=m

+CONFIG\_8139CP=m

+CONFIG\_8139TOO=m

+CONFIG\_8139TOO\_PIO=y

+# CONFIG\_8139TOO\_TUNE\_TWISTER is not set

+CONFIG\_8139TOO\_8129=y

+# CONFIG\_8139\_OLD\_RX\_RESET is not set

+CONFIG\_SIS900=m

+CONFIG\_EPIC100=m

+CONFIG\_SUNDANCE=m

+# CONFIG\_SUNDANCE\_MMIO is not set

+CONFIG\_TLAN=m

+CONFIG\_VIA\_RHINE=m

+CONFIG\_VIA\_RHINE\_MMIO=y

+CONFIG\_VIA\_RHINE\_NAPI=y

+CONFIG\_SC92031=m

+CONFIG\_NET\_POCKET=y

+CONFIG\_ATP=m

+CONFIG\_DE600=m

+CONFIG\_DE620=m

+CONFIG\_NETDEV\_1000=y

+CONFIG\_ACENIC=m

+# CONFIG\_ACENIC\_OMIT\_TIGON\_I is not set

+CONFIG\_DL2K=m

+CONFIG\_E1000=m

+CONFIG\_E1000\_NAPI=y

+# CONFIG\_E1000\_DISABLE\_PACKET\_SPLIT is not set

+CONFIG\_E1000E=m

+CONFIG\_IP1000=m

+CONFIG\_NS83820=m

+CONFIG\_HAMACHI=m

+CONFIG\_YELLOWFIN=m

+CONFIG\_R8169=m

+# CONFIG\_R8169\_NAPI is not set

+CONFIG\_R8169\_VLAN=y

+CONFIG\_SIS190=m

+CONFIG\_SKGE=m

+# CONFIG\_SKGE\_DEBUG is not set

+CONFIG\_SKY2=m

+# CONFIG\_SKY2\_DEBUG is not set

+# CONFIG\_SK98LIN is not set

+CONFIG\_VIA\_VELOCITY=m

+CONFIG\_TIGON3=m

+CONFIG\_BNX2=m

+CONFIG\_QLA3XXX=m

+CONFIG\_ATL1=m

+CONFIG\_NETDEV\_10000=y

+CONFIG\_CHELSIO\_T1=m

+CONFIG\_CHELSIO\_T1\_1G=y

+CONFIG\_CHELSIO\_T1\_NAPI=y

+CONFIG\_CHELSIO\_T3=m

+CONFIG\_IXGBE=m

+CONFIG\_IXGB=m

+# CONFIG\_IXGB\_NAPI is not set

+CONFIG\_S2IO=m

+# CONFIG\_S2IO\_NAPI is not set

+CONFIG\_MYRI10GE=m

+CONFIG\_NETXEN\_NIC=m

+CONFIG\_NIU=m

+CONFIG\_MLX4\_CORE=m

+CONFIG\_MLX4\_DEBUG=y

+CONFIG\_TEHUTI=m

+CONFIG\_TR=y

+CONFIG\_IBMTR=m

+CONFIG\_IBMOL=m

+CONFIG\_IBMLS=m

+CONFIG\_3C359=m

+CONFIG\_TMS380TR=m

+CONFIG\_TMSPCI=m

+CONFIG\_SKISA=m

+CONFIG\_PROTEON=m

+CONFIG\_ABYSS=m

+CONFIG\_MADGEMC=m

+CONFIG\_SMCTR=m

+

+#

+# Wireless LAN

+#

+CONFIG\_WLAN\_PRE80211=y

+CONFIG\_STRIP=m

+CONFIG\_ARLAN=m

+CONFIG\_WAVELAN=m

+CONFIG\_PCMCIA\_WAVELAN=m

+CONFIG\_PCMCIA\_NETWAVE=m

+CONFIG\_WLAN\_80211=y

+CONFIG\_PCMCIA\_RAYCS=m

+CONFIG\_IPW2100=m

+CONFIG\_IPW2100\_MONITOR=y

+# CONFIG\_IPW2100\_DEBUG is not set

+CONFIG\_IPW2200=m

+CONFIG\_IPW2200\_MONITOR=y

+CONFIG\_IPW2200\_RADIOTAP=y

+CONFIG\_IPW2200\_PROMISCUOUS=y

+CONFIG\_IPW2200\_QOS=y

+# CONFIG\_IPW2200\_DEBUG is not set

+CONFIG\_LIBERTAS=m

+CONFIG\_LIBERTAS\_USB=m

+CONFIG\_LIBERTAS\_CS=m

+CONFIG\_LIBERTAS\_SDIO=m

+# CONFIG\_LIBERTAS\_DEBUG is not set

+CONFIG\_AIRO=m

+CONFIG\_HERMES=m

+# CONFIG\_PLX\_HERMES is not set

+# CONFIG\_TMD\_HERMES is not set

+# CONFIG\_NORTEL\_HERMES is not set

+# CONFIG\_PCI\_HERMES is not set

+CONFIG\_PCMCIA\_HERMES=m

+CONFIG\_PCMCIA\_SPECTRUM=m

+CONFIG\_ATMEL=m

+CONFIG\_PCI\_ATMEL=m

+CONFIG\_PCMCIA\_ATMEL=m

+CONFIG\_AIRO\_CS=m

+CONFIG\_PCMCIA\_WL3501=m

+CONFIG\_PRISM54=m

+CONFIG\_USB\_ZD1201=m

+CONFIG\_RTL8187=m

+CONFIG\_ADM8211=m

+CONFIG\_P54\_COMMON=m

+CONFIG\_P54\_USB=m

+CONFIG\_P54\_PCI=m

+# CONFIG\_IWLWIFI is not set

+CONFIG\_HOSTAP=m

+CONFIG\_HOSTAP\_FIRMWARE=y

+CONFIG\_HOSTAP\_FIRMWARE\_NVRAM=y

+CONFIG\_HOSTAP\_PLX=m

+CONFIG\_HOSTAP\_PCI=m

+CONFIG\_HOSTAP\_CS=m

+CONFIG\_BCM43XX=m

+# CONFIG\_BCM43XX\_DEBUG is not set

+CONFIG\_BCM43XX\_DMA=y

+CONFIG\_BCM43XX\_PIO=y

+CONFIG\_BCM43XX\_DMA\_AND\_PIO\_MODE=y

+# CONFIG\_BCM43XX\_DMA\_MODE is not set

+# CONFIG\_BCM43XX\_PIO\_MODE is not set

+CONFIG\_B43=m

+CONFIG\_B43\_PCI\_AUTOSELECT=y

+CONFIG\_B43\_PCICORE\_AUTOSELECT=y

+# CONFIG\_B43\_PCMCIA is not set

+CONFIG\_B43\_LEDS=y

+CONFIG\_B43\_RFKILL=y

+CONFIG\_B43\_DEBUG=y

+CONFIG\_B43\_DMA=y

+CONFIG\_B43\_PIO=y

+CONFIG\_B43\_DMA\_AND\_PIO\_MODE=y

+# CONFIG\_B43\_DMA\_MODE is not set

+# CONFIG\_B43\_PIO\_MODE is not set

+CONFIG\_B43LEGACY=m

+CONFIG\_B43LEGACY\_PCI\_AUTOSELECT=y

+CONFIG\_B43LEGACY\_PCICORE\_AUTOSELECT=y

+CONFIG\_B43LEGACY\_DEBUG=y

+CONFIG\_B43LEGACY\_DMA=y

+CONFIG\_B43LEGACY\_PIO=y

+CONFIG\_B43LEGACY\_DMA\_AND\_PIO\_MODE=y

+# CONFIG\_B43LEGACY\_DMA\_MODE is not set

+# CONFIG\_B43LEGACY\_PIO\_MODE is not set

+CONFIG\_ZD1211RW=m

+# CONFIG\_ZD1211RW\_DEBUG is not set

+CONFIG\_RT2X00=m

+CONFIG\_RT2X00\_LIB=m

+CONFIG\_RT2X00\_LIB\_PCI=m

+CONFIG\_RT2X00\_LIB\_USB=m

+CONFIG\_RT2X00\_LIB\_FIRMWARE=y

+CONFIG\_RT2X00\_LIB\_RFKILL=y

+CONFIG\_RT2400PCI=m

+CONFIG\_RT2400PCI\_RFKILL=y

+CONFIG\_RT2500PCI=m

+CONFIG\_RT2500PCI\_RFKILL=y

+CONFIG\_RT61PCI=m

+CONFIG\_RT61PCI\_RFKILL=y

+CONFIG\_RT2500USB=m

+CONFIG\_RT73USB=m

+# CONFIG\_RT2X00\_LIB\_DEBUGFS is not set

+# CONFIG\_RT2X00\_DEBUG is not set

+

+#

+# USB Network Adapters

+#

+CONFIG\_USB\_CATC=m

+CONFIG\_USB\_KAWETH=m

+CONFIG\_USB\_PEGASUS=m

+CONFIG\_USB\_RTL8150=m

+CONFIG\_USB\_USBNET=m

+CONFIG\_USB\_NET\_AX8817X=m

+CONFIG\_USB\_NET\_CDCETHER=m

+CONFIG\_USB\_NET\_DM9601=m

+CONFIG\_USB\_NET\_GL620A=m

+CONFIG\_USB\_NET\_NET1080=m

+CONFIG\_USB\_NET\_PLUSB=m

+CONFIG\_USB\_NET\_MCS7830=m

+CONFIG\_USB\_NET\_RNDIS\_HOST=m

+CONFIG\_USB\_NET\_CDC\_SUBSET=m

+CONFIG\_USB\_ALI\_M5632=y

+CONFIG\_USB\_AN2720=y

+CONFIG\_USB\_BELKIN=y

+CONFIG\_USB\_ARMLINUX=y

+CONFIG\_USB\_EPSON2888=y

+CONFIG\_USB\_KC2190=y

+# CONFIG\_USB\_NET\_ZAURUS is not set

+CONFIG\_NET\_PCMCIA=y

+CONFIG\_PCMCIA\_3C589=m

+CONFIG\_PCMCIA\_3C574=m

+CONFIG\_PCMCIA\_FMVJ18X=m

+CONFIG\_PCMCIA\_PCNET=m

+CONFIG\_PCMCIA\_NMCLAN=m

+CONFIG\_PCMCIA\_SMC91C92=m

+CONFIG\_PCMCIA\_XIRC2PS=m

+CONFIG\_PCMCIA\_AXNET=m

+CONFIG\_ARCNET\_COM20020\_CS=m

+CONFIG\_PCMCIA\_IBMTR=m

+CONFIG\_WAN=y

+CONFIG\_HOSTESS\_SV11=m

+CONFIG\_COSA=m

+CONFIG\_LANMEDIA=m

+CONFIG\_SEALEVEL\_4021=m

+CONFIG\_HDLC=m

+CONFIG\_HDLC\_RAW=m

+CONFIG\_HDLC\_RAW\_ETH=m

+CONFIG\_HDLC\_CISCO=m

+CONFIG\_HDLC\_FR=m

+CONFIG\_HDLC\_PPP=m

+CONFIG\_HDLC\_X25=m

+CONFIG\_PCI200SYN=m

+CONFIG\_WANXL=m

+CONFIG\_PC300=m

+CONFIG\_PC300\_MLPPP=y

+

+#

+# Cyclades-PC300 MLPPP support is disabled.

+#

+

+#

+# Refer to the file README.mlppp, provided by PC300 package.

+#

+# CONFIG\_PC300TOO is not set

+CONFIG\_N2=m

+CONFIG\_C101=m

+CONFIG\_FARSYNC=m

+CONFIG\_DSCC4=m

+CONFIG\_DSCC4\_PCISYNC=y

+CONFIG\_DSCC4\_PCI\_RST=y

+CONFIG\_DLCI=m

+CONFIG\_DLCI\_MAX=8

+CONFIG\_SDLA=m

+CONFIG\_WAN\_ROUTER\_DRIVERS=m

+CONFIG\_CYCLADES\_SYNC=m

+CONFIG\_CYCLOMX\_X25=y

+CONFIG\_LAPBETHER=m

+CONFIG\_X25\_ASY=m

+CONFIG\_SBNI=m

+# CONFIG\_SBNI\_MULTILINE is not set

+CONFIG\_ATM\_DRIVERS=y

+# CONFIG\_ATM\_DUMMY is not set

+CONFIG\_ATM\_TCP=m

+CONFIG\_ATM\_LANAI=m

+CONFIG\_ATM\_ENI=m

+# CONFIG\_ATM\_ENI\_DEBUG is not set

+# CONFIG\_ATM\_ENI\_TUNE\_BURST is not set

+CONFIG\_ATM\_FIRESTREAM=m

+CONFIG\_ATM\_ZATM=m

+# CONFIG\_ATM\_ZATM\_DEBUG is not set

+CONFIG\_ATM\_NICSTAR=m

+# CONFIG\_ATM\_NICSTAR\_USE\_SUNI is not set

+# CONFIG\_ATM\_NICSTAR\_USE\_IDT77105 is not set

+CONFIG\_ATM\_IDT77252=m

+# CONFIG\_ATM\_IDT77252\_DEBUG is not set

+# CONFIG\_ATM\_IDT77252\_RCV\_ALL is not set

+CONFIG\_ATM\_IDT77252\_USE\_SUNI=y

+CONFIG\_ATM\_AMBASSADOR=m

+# CONFIG\_ATM\_AMBASSADOR\_DEBUG is not set

+CONFIG\_ATM\_HORIZON=m

+# CONFIG\_ATM\_HORIZON\_DEBUG is not set

+CONFIG\_ATM\_IA=m

+# CONFIG\_ATM\_IA\_DEBUG is not set

+CONFIG\_ATM\_FORE200E\_MAYBE=m

+CONFIG\_ATM\_FORE200E\_PCA=y

+CONFIG\_ATM\_FORE200E\_PCA\_DEFAULT\_FW=y

+# CONFIG\_ATM\_FORE200E\_USE\_TASKLET is not set

+CONFIG\_ATM\_FORE200E\_TX\_RETRY=16

+CONFIG\_ATM\_FORE200E\_DEBUG=0

+CONFIG\_ATM\_FORE200E=m

+CONFIG\_ATM\_HE=m

+CONFIG\_ATM\_HE\_USE\_SUNI=y

+CONFIG\_FDDI=y

+CONFIG\_DEFXX=m

+# CONFIG\_DEFXX\_MMIO is not set

+CONFIG\_SKFP=m

+CONFIG\_HIPPI=y

+CONFIG\_ROADRUNNER=m

+# CONFIG\_ROADRUNNER\_LARGE\_RINGS is not set

+CONFIG\_PLIP=m

+CONFIG\_PPP=m

+CONFIG\_PPP\_MULTILINK=y

+CONFIG\_PPP\_FILTER=y

+CONFIG\_PPP\_ASYNC=m

+CONFIG\_PPP\_SYNC\_TTY=m

+CONFIG\_PPP\_DEFLATE=m

+CONFIG\_PPP\_BSDCOMP=m

+CONFIG\_PPP\_MPPE=m

+CONFIG\_PPPOE=m

+CONFIG\_PPPOATM=m

+CONFIG\_PPPOL2TP=m

+CONFIG\_SLIP=m

+CONFIG\_SLIP\_COMPRESSED=y

+CONFIG\_SLHC=m

+CONFIG\_SLIP\_SMART=y

+CONFIG\_SLIP\_MODE\_SLIP6=y

+CONFIG\_NET\_FC=y

+CONFIG\_SHAPER=m

+CONFIG\_NETCONSOLE=m

+CONFIG\_NETCONSOLE\_DYNAMIC=y

+CONFIG\_NETPOLL=y

+# CONFIG\_NETPOLL\_TRAP is not set

+CONFIG\_NET\_POLL\_CONTROLLER=y

+CONFIG\_ISDN=m

+CONFIG\_ISDN\_I4L=m

+CONFIG\_ISDN\_PPP=y

+CONFIG\_ISDN\_PPP\_VJ=y

+CONFIG\_ISDN\_MPP=y

+CONFIG\_IPPP\_FILTER=y

+CONFIG\_ISDN\_PPP\_BSDCOMP=m

+CONFIG\_ISDN\_AUDIO=y

+CONFIG\_ISDN\_TTY\_FAX=y

+CONFIG\_ISDN\_X25=y

+

+#

+# ISDN feature submodules

+#

+CONFIG\_ISDN\_DIVERSION=m

+

+#

+# ISDN4Linux hardware drivers

+#

+

+#

+# Passive cards

+#

+CONFIG\_ISDN\_DRV\_HISAX=m

+

+#

+# D-channel protocol features

+#

+CONFIG\_HISAX\_EURO=y

+CONFIG\_DE\_AOC=y

+# CONFIG\_HISAX\_NO\_SENDCOMPLETE is not set

+# CONFIG\_HISAX\_NO\_LLC is not set

+# CONFIG\_HISAX\_NO\_KEYPAD is not set

+CONFIG\_HISAX\_1TR6=y

+CONFIG\_HISAX\_NI1=y

+CONFIG\_HISAX\_MAX\_CARDS=8

+

+#

+# HiSax supported cards

+#

+CONFIG\_HISAX\_16\_0=y

+CONFIG\_HISAX\_16\_3=y

+CONFIG\_HISAX\_TELESPCI=y

+CONFIG\_HISAX\_S0BOX=y

+CONFIG\_HISAX\_AVM\_A1=y

+CONFIG\_HISAX\_FRITZPCI=y

+CONFIG\_HISAX\_AVM\_A1\_PCMCIA=y

+CONFIG\_HISAX\_ELSA=y

+CONFIG\_HISAX\_IX1MICROR2=y

+CONFIG\_HISAX\_DIEHLDIVA=y

+CONFIG\_HISAX\_ASUSCOM=y

+CONFIG\_HISAX\_TELEINT=y

+CONFIG\_HISAX\_HFCS=y

+CONFIG\_HISAX\_SEDLBAUER=y

+CONFIG\_HISAX\_SPORTSTER=y

+CONFIG\_HISAX\_MIC=y

+CONFIG\_HISAX\_NETJET=y

+CONFIG\_HISAX\_NETJET\_U=y

+CONFIG\_HISAX\_NICCY=y

+CONFIG\_HISAX\_ISURF=y

+CONFIG\_HISAX\_HSTSAPHIR=y

+CONFIG\_HISAX\_BKM\_A4T=y

+CONFIG\_HISAX\_SCT\_QUADRO=y

+CONFIG\_HISAX\_GAZEL=y

+CONFIG\_HISAX\_HFC\_PCI=y

+CONFIG\_HISAX\_W6692=y

+CONFIG\_HISAX\_HFC\_SX=y

+CONFIG\_HISAX\_ENTERNOW\_PCI=y

+# CONFIG\_HISAX\_DEBUG is not set

+

+#

+# HiSax PCMCIA card service modules

+#

+CONFIG\_HISAX\_SEDLBAUER\_CS=m

+CONFIG\_HISAX\_ELSA\_CS=m

+CONFIG\_HISAX\_AVM\_A1\_CS=m

+CONFIG\_HISAX\_TELES\_CS=m

+

+#

+# HiSax sub driver modules

+#

+CONFIG\_HISAX\_ST5481=m

+CONFIG\_HISAX\_HFCUSB=m

+CONFIG\_HISAX\_HFC4S8S=m

+CONFIG\_HISAX\_FRITZ\_PCIPNP=m

+CONFIG\_HISAX\_HDLC=y

+

+#

+# Active cards

+#

+CONFIG\_ISDN\_DRV\_ICN=m

+CONFIG\_ISDN\_DRV\_PCBIT=m

+CONFIG\_ISDN\_DRV\_SC=m

+CONFIG\_ISDN\_DRV\_ACT2000=m

+CONFIG\_ISDN\_DRV\_GIGASET=m

+CONFIG\_GIGASET\_BASE=m

+CONFIG\_GIGASET\_M105=m

+CONFIG\_GIGASET\_M101=m

+# CONFIG\_GIGASET\_DEBUG is not set

+# CONFIG\_GIGASET\_UNDOCREQ is not set

+CONFIG\_ISDN\_CAPI=m

+CONFIG\_ISDN\_DRV\_AVMB1\_VERBOSE\_REASON=y

+CONFIG\_CAPI\_TRACE=y

+CONFIG\_ISDN\_CAPI\_MIDDLEWARE=y

+CONFIG\_ISDN\_CAPI\_CAPI20=m

+CONFIG\_ISDN\_CAPI\_CAPIFS\_BOOL=y

+CONFIG\_ISDN\_CAPI\_CAPIFS=m

+CONFIG\_ISDN\_CAPI\_CAPIDRV=m

+

+#

+# CAPI hardware drivers

+#

+CONFIG\_CAPI\_AVM=y

+CONFIG\_ISDN\_DRV\_AVMB1\_B1ISA=m

+CONFIG\_ISDN\_DRV\_AVMB1\_B1PCI=m

+CONFIG\_ISDN\_DRV\_AVMB1\_B1PCIV4=y

+CONFIG\_ISDN\_DRV\_AVMB1\_T1ISA=m

+CONFIG\_ISDN\_DRV\_AVMB1\_B1PCMCIA=m

+CONFIG\_ISDN\_DRV\_AVMB1\_AVM\_CS=m

+CONFIG\_ISDN\_DRV\_AVMB1\_T1PCI=m

+CONFIG\_ISDN\_DRV\_AVMB1\_C4=m

+CONFIG\_CAPI\_EICON=y

+CONFIG\_ISDN\_DIVAS=m

+CONFIG\_ISDN\_DIVAS\_BRIPCI=y

+CONFIG\_ISDN\_DIVAS\_PRIPCI=y

+CONFIG\_ISDN\_DIVAS\_DIVACAPI=m

+CONFIG\_ISDN\_DIVAS\_USERIDI=m

+CONFIG\_ISDN\_DIVAS\_MAINT=m

+CONFIG\_PHONE=m

+CONFIG\_PHONE\_IXJ=m

+CONFIG\_PHONE\_IXJ\_PCMCIA=m

+

+#

+# Input device support

+#

+CONFIG\_INPUT=y

+CONFIG\_INPUT\_FF\_MEMLESS=m

+CONFIG\_INPUT\_POLLDEV=m

+

+#

+# Userland interfaces

+#

+CONFIG\_INPUT\_MOUSEDEV=y

+CONFIG\_INPUT\_MOUSEDEV\_PSAUX=y

+CONFIG\_INPUT\_MOUSEDEV\_SCREEN\_X=1024

+CONFIG\_INPUT\_MOUSEDEV\_SCREEN\_Y=768

+CONFIG\_INPUT\_JOYDEV=m

+CONFIG\_INPUT\_EVDEV=m

+CONFIG\_INPUT\_EVBUG=m

+

+#

+# Input Device Drivers

+#

+CONFIG\_INPUT\_KEYBOARD=y

+CONFIG\_KEYBOARD\_ATKBD=y

+CONFIG\_KEYBOARD\_SUNKBD=m

+CONFIG\_KEYBOARD\_LKKBD=m

+CONFIG\_KEYBOARD\_XTKBD=m

+CONFIG\_KEYBOARD\_NEWTON=m

+CONFIG\_KEYBOARD\_STOWAWAY=m

+CONFIG\_INPUT\_MOUSE=y

+CONFIG\_MOUSE\_PS2=m

+CONFIG\_MOUSE\_PS2\_ALPS=y

+CONFIG\_MOUSE\_PS2\_LOGIPS2PP=y

+CONFIG\_MOUSE\_PS2\_SYNAPTICS=y

+CONFIG\_MOUSE\_PS2\_LIFEBOOK=y

+CONFIG\_MOUSE\_PS2\_TRACKPOINT=y

+# CONFIG\_MOUSE\_PS2\_TOUCHKIT is not set

+CONFIG\_MOUSE\_SERIAL=m

+CONFIG\_MOUSE\_APPLETOUCH=m

+CONFIG\_MOUSE\_INPORT=m

+# CONFIG\_MOUSE\_ATIXL is not set

+CONFIG\_MOUSE\_LOGIBM=m

+CONFIG\_MOUSE\_PC110PAD=m

+CONFIG\_MOUSE\_VSXXXAA=m

+CONFIG\_INPUT\_JOYSTICK=y

+CONFIG\_JOYSTICK\_ANALOG=m

+CONFIG\_JOYSTICK\_A3D=m

+CONFIG\_JOYSTICK\_ADI=m

+CONFIG\_JOYSTICK\_COBRA=m

+CONFIG\_JOYSTICK\_GF2K=m

+CONFIG\_JOYSTICK\_GRIP=m

+CONFIG\_JOYSTICK\_GRIP\_MP=m

+CONFIG\_JOYSTICK\_GUILLEMOT=m

+CONFIG\_JOYSTICK\_INTERACT=m

+CONFIG\_JOYSTICK\_SIDEWINDER=m

+CONFIG\_JOYSTICK\_TMDC=m

+CONFIG\_JOYSTICK\_IFORCE=m

+CONFIG\_JOYSTICK\_IFORCE\_USB=y

+CONFIG\_JOYSTICK\_IFORCE\_232=y

+CONFIG\_JOYSTICK\_WARRIOR=m

+CONFIG\_JOYSTICK\_MAGELLAN=m

+CONFIG\_JOYSTICK\_SPACEORB=m

+CONFIG\_JOYSTICK\_SPACEBALL=m

+CONFIG\_JOYSTICK\_STINGER=m

+CONFIG\_JOYSTICK\_TWIDJOY=m

+CONFIG\_JOYSTICK\_DB9=m

+CONFIG\_JOYSTICK\_GAMECON=m

+CONFIG\_JOYSTICK\_TURBOGRAFX=m

+CONFIG\_JOYSTICK\_JOYDUMP=m

+CONFIG\_JOYSTICK\_XPAD=m

+CONFIG\_JOYSTICK\_XPAD\_FF=y

+CONFIG\_JOYSTICK\_XPAD\_LEDS=y

+CONFIG\_INPUT\_TABLET=y

+CONFIG\_TABLET\_USB\_ACECAD=m

+CONFIG\_TABLET\_USB\_AIPTEK=m

+CONFIG\_TABLET\_USB\_GTCO=m

+CONFIG\_TABLET\_USB\_KBTAB=m

+CONFIG\_TABLET\_USB\_WACOM=m

+CONFIG\_INPUT\_TOUCHSCREEN=y

+CONFIG\_TOUCHSCREEN\_ADS7846=m

+CONFIG\_TOUCHSCREEN\_FUJITSU=m

+CONFIG\_TOUCHSCREEN\_GUNZE=m

+CONFIG\_TOUCHSCREEN\_ELO=m

+CONFIG\_TOUCHSCREEN\_MTOUCH=m

+CONFIG\_TOUCHSCREEN\_MK712=m

+CONFIG\_TOUCHSCREEN\_PENMOUNT=m

+CONFIG\_TOUCHSCREEN\_TOUCHRIGHT=m

+CONFIG\_TOUCHSCREEN\_TOUCHWIN=m

+CONFIG\_TOUCHSCREEN\_UCB1400=m

+CONFIG\_TOUCHSCREEN\_USB\_COMPOSITE=m

+CONFIG\_TOUCHSCREEN\_USB\_EGALAX=y

+CONFIG\_TOUCHSCREEN\_USB\_PANJIT=y

+CONFIG\_TOUCHSCREEN\_USB\_3M=y

+CONFIG\_TOUCHSCREEN\_USB\_ITM=y

+CONFIG\_TOUCHSCREEN\_USB\_ETURBO=y

+CONFIG\_TOUCHSCREEN\_USB\_GUNZE=y

+CONFIG\_TOUCHSCREEN\_USB\_DMC\_TSC10=y

+CONFIG\_TOUCHSCREEN\_USB\_IRTOUCH=y

+CONFIG\_TOUCHSCREEN\_USB\_IDEALTEK=y

+CONFIG\_TOUCHSCREEN\_USB\_GENERAL\_TOUCH=y

+CONFIG\_TOUCHSCREEN\_USB\_GOTOP=y

+CONFIG\_INPUT\_MISC=y

+CONFIG\_INPUT\_PCSPKR=m

+CONFIG\_INPUT\_WISTRON\_BTNS=m

+CONFIG\_INPUT\_ATLAS\_BTNS=m

+CONFIG\_INPUT\_ATI\_REMOTE=m

+CONFIG\_INPUT\_ATI\_REMOTE2=m

+CONFIG\_INPUT\_KEYSPAN\_REMOTE=m

+CONFIG\_INPUT\_POWERMATE=m

+CONFIG\_INPUT\_YEALINK=m

+CONFIG\_INPUT\_UINPUT=m

+

+#

+# Hardware I/O ports

+#

+CONFIG\_SERIO=y

+CONFIG\_SERIO\_I8042=y

+CONFIG\_SERIO\_SERPORT=m

+CONFIG\_SERIO\_CT82C710=m

+CONFIG\_SERIO\_PARKBD=m

+CONFIG\_SERIO\_PCIPS2=m

+CONFIG\_SERIO\_LIBPS2=y

+CONFIG\_SERIO\_RAW=m

+CONFIG\_GAMEPORT=m

+CONFIG\_GAMEPORT\_NS558=m

+CONFIG\_GAMEPORT\_L4=m

+CONFIG\_GAMEPORT\_EMU10K1=m

+CONFIG\_GAMEPORT\_FM801=m

+

+#

+# Character devices

+#

+CONFIG\_VT=y

+CONFIG\_VT\_CONSOLE=y

+CONFIG\_HW\_CONSOLE=y

+CONFIG\_VT\_HW\_CONSOLE\_BINDING=y

+CONFIG\_SERIAL\_NONSTANDARD=y

+# CONFIG\_COMPUTONE is not set

+CONFIG\_ROCKETPORT=m

+CONFIG\_CYCLADES=m

+# CONFIG\_CYZ\_INTR is not set

+CONFIG\_DIGIEPCA=m

+# CONFIG\_ESPSERIAL is not set

+# CONFIG\_MOXA\_INTELLIO is not set

+# CONFIG\_MOXA\_SMARTIO is not set

+CONFIG\_MOXA\_SMARTIO\_NEW=m

+# CONFIG\_ISI is not set

+CONFIG\_SYNCLINK=m

+CONFIG\_SYNCLINKMP=m

+CONFIG\_SYNCLINK\_GT=m

+CONFIG\_N\_HDLC=m

+CONFIG\_SPECIALIX=m

+# CONFIG\_SPECIALIX\_RTSCTS is not set

+CONFIG\_SX=m

+# CONFIG\_RIO is not set

+CONFIG\_STALDRV=y

+

+#

+# Serial drivers

+#

+CONFIG\_SERIAL\_8250=y

+CONFIG\_SERIAL\_8250\_CONSOLE=y

+CONFIG\_FIX\_EARLYCON\_MEM=y

+CONFIG\_SERIAL\_8250\_PCI=y

+CONFIG\_SERIAL\_8250\_PNP=y

+CONFIG\_SERIAL\_8250\_CS=m

+CONFIG\_SERIAL\_8250\_NR\_UARTS=48

+CONFIG\_SERIAL\_8250\_RUNTIME\_UARTS=4

+CONFIG\_SERIAL\_8250\_EXTENDED=y

+CONFIG\_SERIAL\_8250\_MANY\_PORTS=y

+CONFIG\_SERIAL\_8250\_FOURPORT=m

+CONFIG\_SERIAL\_8250\_ACCENT=m

+CONFIG\_SERIAL\_8250\_BOCA=m

+CONFIG\_SERIAL\_8250\_EXAR\_ST16C554=m

+CONFIG\_SERIAL\_8250\_HUB6=m

+CONFIG\_SERIAL\_8250\_SHARE\_IRQ=y

+# CONFIG\_SERIAL\_8250\_DETECT\_IRQ is not set

+CONFIG\_SERIAL\_8250\_RSA=y

+CONFIG\_SERIAL\_8250\_MCA=m

+

+#

+# Non-8250 serial port support

+#

+CONFIG\_SERIAL\_CORE=y

+CONFIG\_SERIAL\_CORE\_CONSOLE=y

+CONFIG\_SERIAL\_JSM=m

+CONFIG\_UNIX98\_PTYS=y

+CONFIG\_LEGACY\_PTYS=y

+CONFIG\_LEGACY\_PTY\_COUNT=256

+CONFIG\_PRINTER=m

+# CONFIG\_LP\_CONSOLE is not set

+CONFIG\_PPDEV=m

+CONFIG\_HVC\_DRIVER=y

+CONFIG\_IPMI\_HANDLER=m

+# CONFIG\_IPMI\_PANIC\_EVENT is not set

+CONFIG\_IPMI\_DEVICE\_INTERFACE=m

+CONFIG\_IPMI\_SI=m

+CONFIG\_IPMI\_WATCHDOG=m

+CONFIG\_IPMI\_POWEROFF=m

+CONFIG\_HW\_RANDOM=y

+CONFIG\_HW\_RANDOM\_INTEL=m

+CONFIG\_HW\_RANDOM\_AMD=m

+CONFIG\_HW\_RANDOM\_GEODE=m

+CONFIG\_HW\_RANDOM\_VIA=m

+CONFIG\_NVRAM=m

+CONFIG\_RTC=y

+CONFIG\_DTLK=m

+CONFIG\_R3964=m

+CONFIG\_APPLICOM=m

+CONFIG\_SONYPI=m

+

+#

+# PCMCIA character devices

+#

+CONFIG\_SYNCLINK\_CS=m

+CONFIG\_CARDMAN\_4000=m

+CONFIG\_CARDMAN\_4040=m

+CONFIG\_MWAVE=m

+CONFIG\_SCx200\_GPIO=m

+CONFIG\_PC8736x\_GPIO=m

+CONFIG\_NSC\_GPIO=m

+CONFIG\_CS5535\_GPIO=m

+CONFIG\_RAW\_DRIVER=m

+CONFIG\_MAX\_RAW\_DEVS=256

+CONFIG\_HPET=y

+# CONFIG\_HPET\_RTC\_IRQ is not set

+CONFIG\_HPET\_MMAP=y

+CONFIG\_HANGCHECK\_TIMER=m

+CONFIG\_TCG\_TPM=m

+CONFIG\_TCG\_TIS=m

+CONFIG\_TCG\_NSC=m

+CONFIG\_TCG\_ATMEL=m

+CONFIG\_TCG\_INFINEON=m

+CONFIG\_TELCLOCK=m

+CONFIG\_DEVPORT=y

+CONFIG\_I2C=m

+CONFIG\_I2C\_BOARDINFO=y

+CONFIG\_I2C\_CHARDEV=m

+

+#

+# I2C Algorithms

+#

+CONFIG\_I2C\_ALGOBIT=m

+CONFIG\_I2C\_ALGOPCF=m

+CONFIG\_I2C\_ALGOPCA=m

+

+#

+# I2C Hardware Bus support

+#

+CONFIG\_I2C\_ALI1535=m

+CONFIG\_I2C\_ALI1563=m

+CONFIG\_I2C\_ALI15X3=m

+CONFIG\_I2C\_AMD756=m

+CONFIG\_I2C\_AMD756\_S4882=m

+CONFIG\_I2C\_AMD8111=m

+CONFIG\_I2C\_I801=m

+CONFIG\_I2C\_I810=m

+CONFIG\_I2C\_PIIX4=m

+CONFIG\_I2C\_NFORCE2=m

+CONFIG\_I2C\_OCORES=m

+CONFIG\_I2C\_PARPORT=m

+CONFIG\_I2C\_PARPORT\_LIGHT=m

+CONFIG\_I2C\_PROSAVAGE=m

+CONFIG\_I2C\_SAVAGE4=m

+CONFIG\_I2C\_SIMTEC=m

+CONFIG\_SCx200\_I2C=m

+CONFIG\_SCx200\_I2C\_SCL=12

+CONFIG\_SCx200\_I2C\_SDA=13

+CONFIG\_SCx200\_ACB=m

+CONFIG\_I2C\_SIS5595=m

+CONFIG\_I2C\_SIS630=m

+CONFIG\_I2C\_SIS96X=m

+CONFIG\_I2C\_TAOS\_EVM=m

+CONFIG\_I2C\_STUB=m

+CONFIG\_I2C\_TINY\_USB=m

+CONFIG\_I2C\_VIA=m

+CONFIG\_I2C\_VIAPRO=m

+CONFIG\_I2C\_VOODOO3=m

+CONFIG\_I2C\_PCA\_ISA=m

+

+#

+# Miscellaneous I2C Chip support

+#

+CONFIG\_SENSORS\_DS1337=m

+CONFIG\_SENSORS\_DS1374=m

+CONFIG\_DS1682=m

+CONFIG\_SENSORS\_EEPROM=m

+CONFIG\_SENSORS\_PCF8574=m

+CONFIG\_SENSORS\_PCA9539=m

+CONFIG\_SENSORS\_PCF8591=m

+CONFIG\_SENSORS\_MAX6875=m

+CONFIG\_SENSORS\_TSL2550=m

+# CONFIG\_I2C\_DEBUG\_CORE is not set

+# CONFIG\_I2C\_DEBUG\_ALGO is not set

+# CONFIG\_I2C\_DEBUG\_BUS is not set

+# CONFIG\_I2C\_DEBUG\_CHIP is not set

+

+#

+# SPI support

+#

+CONFIG\_SPI=y

+# CONFIG\_SPI\_DEBUG is not set

+CONFIG\_SPI\_MASTER=y

+

+#

+# SPI Master Controller Drivers

+#

+CONFIG\_SPI\_BITBANG=m

+CONFIG\_SPI\_BUTTERFLY=m

+CONFIG\_SPI\_LM70\_LLP=m

+

+#

+# SPI Protocol Masters

+#

+CONFIG\_SPI\_AT25=m

+CONFIG\_SPI\_SPIDEV=m

+CONFIG\_SPI\_TLE62X0=m

+CONFIG\_W1=m

+CONFIG\_W1\_CON=y

+

+#

+# 1-wire Bus Masters

+#

+CONFIG\_W1\_MASTER\_MATROX=m

+CONFIG\_W1\_MASTER\_DS2490=m

+CONFIG\_W1\_MASTER\_DS2482=m

+

+#

+# 1-wire Slaves

+#

+CONFIG\_W1\_SLAVE\_THERM=m

+CONFIG\_W1\_SLAVE\_SMEM=m

+CONFIG\_W1\_SLAVE\_DS2433=m

+# CONFIG\_W1\_SLAVE\_DS2433\_CRC is not set

+CONFIG\_W1\_SLAVE\_DS2760=m

+CONFIG\_POWER\_SUPPLY=y

+# CONFIG\_POWER\_SUPPLY\_DEBUG is not set

+CONFIG\_PDA\_POWER=m

+CONFIG\_BATTERY\_DS2760=m

+CONFIG\_HWMON=y

+CONFIG\_HWMON\_VID=m

+CONFIG\_SENSORS\_ABITUGURU=m

+CONFIG\_SENSORS\_ABITUGURU3=m

+CONFIG\_SENSORS\_AD7418=m

+CONFIG\_SENSORS\_ADM1021=m

+CONFIG\_SENSORS\_ADM1025=m

+CONFIG\_SENSORS\_ADM1026=m

+CONFIG\_SENSORS\_ADM1029=m

+CONFIG\_SENSORS\_ADM1031=m

+CONFIG\_SENSORS\_ADM9240=m

+CONFIG\_SENSORS\_ADT7470=m

+CONFIG\_SENSORS\_K8TEMP=m

+CONFIG\_SENSORS\_ASB100=m

+CONFIG\_SENSORS\_ATXP1=m

+CONFIG\_SENSORS\_DS1621=m

+CONFIG\_SENSORS\_I5K\_AMB=m

+CONFIG\_SENSORS\_F71805F=m

+CONFIG\_SENSORS\_F71882FG=m

+CONFIG\_SENSORS\_F75375S=m

+CONFIG\_SENSORS\_FSCHER=m

+CONFIG\_SENSORS\_FSCPOS=m

+CONFIG\_SENSORS\_FSCHMD=m

+CONFIG\_SENSORS\_GL518SM=m

+CONFIG\_SENSORS\_GL520SM=m

+CONFIG\_SENSORS\_CORETEMP=m

+CONFIG\_SENSORS\_IBMPEX=m

+CONFIG\_SENSORS\_IT87=m

+CONFIG\_SENSORS\_LM63=m

+CONFIG\_SENSORS\_LM70=m

+CONFIG\_SENSORS\_LM75=m

+CONFIG\_SENSORS\_LM77=m

+CONFIG\_SENSORS\_LM78=m

+CONFIG\_SENSORS\_LM80=m

+CONFIG\_SENSORS\_LM83=m

+CONFIG\_SENSORS\_LM85=m

+CONFIG\_SENSORS\_LM87=m

+CONFIG\_SENSORS\_LM90=m

+CONFIG\_SENSORS\_LM92=m

+CONFIG\_SENSORS\_LM93=m

+CONFIG\_SENSORS\_MAX1619=m

+CONFIG\_SENSORS\_MAX6650=m

+CONFIG\_SENSORS\_PC87360=m

+CONFIG\_SENSORS\_PC87427=m

+CONFIG\_SENSORS\_SIS5595=m

+CONFIG\_SENSORS\_DME1737=m

+CONFIG\_SENSORS\_SMSC47M1=m

+CONFIG\_SENSORS\_SMSC47M192=m

+CONFIG\_SENSORS\_SMSC47B397=m

+CONFIG\_SENSORS\_THMC50=m

+CONFIG\_SENSORS\_VIA686A=m

+CONFIG\_SENSORS\_VT1211=m

+CONFIG\_SENSORS\_VT8231=m

+CONFIG\_SENSORS\_W83781D=m

+CONFIG\_SENSORS\_W83791D=m

+CONFIG\_SENSORS\_W83792D=m

+CONFIG\_SENSORS\_W83793=m

+CONFIG\_SENSORS\_W83L785TS=m

+CONFIG\_SENSORS\_W83627HF=m

+CONFIG\_SENSORS\_W83627EHF=m

+CONFIG\_SENSORS\_HDAPS=m

+CONFIG\_SENSORS\_APPLESMC=m

+# CONFIG\_HWMON\_DEBUG\_CHIP is not set

+CONFIG\_WATCHDOG=y

+# CONFIG\_WATCHDOG\_NOWAYOUT is not set

+

+#

+# Watchdog Device Drivers

+#

+CONFIG\_SOFT\_WATCHDOG=m

+CONFIG\_ACQUIRE\_WDT=m

+CONFIG\_ADVANTECH\_WDT=m

+CONFIG\_ALIM1535\_WDT=m

+CONFIG\_ALIM7101\_WDT=m

+CONFIG\_SC520\_WDT=m

+CONFIG\_EUROTECH\_WDT=m

+CONFIG\_IB700\_WDT=m

+CONFIG\_IBMASR=m

+CONFIG\_WAFER\_WDT=m

+CONFIG\_I6300ESB\_WDT=m

+CONFIG\_ITCO\_WDT=m

+CONFIG\_ITCO\_VENDOR\_SUPPORT=y

+CONFIG\_IT8712F\_WDT=m

+CONFIG\_SC1200\_WDT=m

+CONFIG\_SCx200\_WDT=m

+CONFIG\_PC87413\_WDT=m

+CONFIG\_60XX\_WDT=m

+CONFIG\_SBC8360\_WDT=m

+CONFIG\_SBC7240\_WDT=m

+CONFIG\_CPU5\_WDT=m

+CONFIG\_SMSC37B787\_WDT=m

+CONFIG\_W83627HF\_WDT=m

+CONFIG\_W83697HF\_WDT=m

+CONFIG\_W83877F\_WDT=m

+CONFIG\_W83977F\_WDT=m

+CONFIG\_MACHZ\_WDT=m

+CONFIG\_SBC\_EPX\_C3\_WATCHDOG=m

+

+#

+# ISA-based Watchdog Cards

+#

+CONFIG\_PCWATCHDOG=m

+CONFIG\_MIXCOMWD=m

+CONFIG\_WDT=m

+CONFIG\_WDT\_501=y

+

+#

+# PCI-based Watchdog Cards

+#

+CONFIG\_PCIPCWATCHDOG=m

+CONFIG\_WDTPCI=m

+CONFIG\_WDT\_501\_PCI=y

+

+#

+# USB-based Watchdog Cards

+#

+CONFIG\_USBPCWATCHDOG=m

+

+#

+# Sonics Silicon Backplane

+#

+CONFIG\_SSB\_POSSIBLE=y

+CONFIG\_SSB=m

+CONFIG\_SSB\_PCIHOST\_POSSIBLE=y

+CONFIG\_SSB\_PCIHOST=y

+CONFIG\_SSB\_PCMCIAHOST\_POSSIBLE=y

+# CONFIG\_SSB\_PCMCIAHOST is not set

+# CONFIG\_SSB\_SILENT is not set

+CONFIG\_SSB\_DEBUG=y

+CONFIG\_SSB\_DRIVER\_PCICORE\_POSSIBLE=y

+CONFIG\_SSB\_DRIVER\_PCICORE=y

+

+#

+# Multifunction device drivers

+#

+CONFIG\_MFD\_SM501=m

+

+#

+# Multimedia devices

+#

+CONFIG\_VIDEO\_DEV=m

+CONFIG\_VIDEO\_V4L1=y

+CONFIG\_VIDEO\_V4L1\_COMPAT=y

+CONFIG\_VIDEO\_V4L2=y

+CONFIG\_VIDEO\_CAPTURE\_DRIVERS=y

+# CONFIG\_VIDEO\_ADV\_DEBUG is not set

+# CONFIG\_VIDEO\_HELPER\_CHIPS\_AUTO is not set

+

+#

+# Encoders/decoders and other helper chips

+#

+

+#

+# Audio decoders

+#

+CONFIG\_VIDEO\_TVAUDIO=m

+CONFIG\_VIDEO\_TDA7432=m

+CONFIG\_VIDEO\_TDA9840=m

+CONFIG\_VIDEO\_TDA9875=m

+CONFIG\_VIDEO\_TEA6415C=m

+CONFIG\_VIDEO\_TEA6420=m

+CONFIG\_VIDEO\_MSP3400=m

+CONFIG\_VIDEO\_CS53L32A=m

+CONFIG\_VIDEO\_TLV320AIC23B=m

+CONFIG\_VIDEO\_WM8775=m

+CONFIG\_VIDEO\_WM8739=m

+CONFIG\_VIDEO\_VP27SMPX=m

+

+#

+# Video decoders

+#

+CONFIG\_VIDEO\_BT819=m

+CONFIG\_VIDEO\_BT856=m

+CONFIG\_VIDEO\_BT866=m

+CONFIG\_VIDEO\_KS0127=m

+CONFIG\_VIDEO\_OV7670=m

+CONFIG\_VIDEO\_TCM825X=m

+CONFIG\_VIDEO\_SAA7110=m

+CONFIG\_VIDEO\_SAA7111=m

+CONFIG\_VIDEO\_SAA7114=m

+CONFIG\_VIDEO\_SAA711X=m

+CONFIG\_VIDEO\_SAA7191=m

+CONFIG\_VIDEO\_TVP5150=m

+CONFIG\_VIDEO\_VPX3220=m

+

+#

+# Video and audio decoders

+#

+CONFIG\_VIDEO\_CX25840=m

+

+#

+# MPEG video encoders

+#

+CONFIG\_VIDEO\_CX2341X=m

+

+#

+# Video encoders

+#

+CONFIG\_VIDEO\_SAA7127=m

+CONFIG\_VIDEO\_SAA7185=m

+CONFIG\_VIDEO\_ADV7170=m

+CONFIG\_VIDEO\_ADV7175=m

+

+#

+# Video improvement chips

+#

+CONFIG\_VIDEO\_UPD64031A=m

+CONFIG\_VIDEO\_UPD64083=m

+CONFIG\_VIDEO\_VIVI=m

+CONFIG\_VIDEO\_BT848=m

+CONFIG\_VIDEO\_BT848\_DVB=y

+CONFIG\_VIDEO\_SAA6588=m

+CONFIG\_VIDEO\_PMS=m

+CONFIG\_VIDEO\_BWQCAM=m

+CONFIG\_VIDEO\_CQCAM=m

+CONFIG\_VIDEO\_W9966=m

+CONFIG\_VIDEO\_CPIA=m

+CONFIG\_VIDEO\_CPIA\_PP=m

+CONFIG\_VIDEO\_CPIA\_USB=m

+CONFIG\_VIDEO\_CPIA2=m

+CONFIG\_VIDEO\_SAA5246A=m

+CONFIG\_VIDEO\_SAA5249=m

+CONFIG\_TUNER\_3036=m

+CONFIG\_VIDEO\_STRADIS=m

+CONFIG\_VIDEO\_ZORAN\_ZR36060=m

+CONFIG\_VIDEO\_ZORAN=m

+CONFIG\_VIDEO\_ZORAN\_BUZ=m

+CONFIG\_VIDEO\_ZORAN\_DC10=m

+CONFIG\_VIDEO\_ZORAN\_DC30=m

+CONFIG\_VIDEO\_ZORAN\_LML33=m

+CONFIG\_VIDEO\_ZORAN\_LML33R10=m

+CONFIG\_VIDEO\_ZORAN\_AVS6EYES=m

+CONFIG\_VIDEO\_MEYE=m

+CONFIG\_VIDEO\_SAA7134=m

+CONFIG\_VIDEO\_SAA7134\_DVB=m

+# CONFIG\_VIDEO\_MXB is not set

+# CONFIG\_VIDEO\_DPC is not set

+CONFIG\_VIDEO\_HEXIUM\_ORION=m

+CONFIG\_VIDEO\_HEXIUM\_GEMINI=m

+CONFIG\_VIDEO\_CX88=m

+CONFIG\_VIDEO\_CX88\_BLACKBIRD=m

+CONFIG\_VIDEO\_CX88\_DVB=m

+CONFIG\_VIDEO\_CX88\_VP3054=m

+CONFIG\_VIDEO\_CX23885=m

+CONFIG\_VIDEO\_IVTV=m

+CONFIG\_VIDEO\_FB\_IVTV=m

+CONFIG\_VIDEO\_CAFE\_CCIC=m

+CONFIG\_V4L\_USB\_DRIVERS=y

+CONFIG\_VIDEO\_PVRUSB2=m

+CONFIG\_VIDEO\_PVRUSB2\_29XXX=y

+CONFIG\_VIDEO\_PVRUSB2\_24XXX=y

+CONFIG\_VIDEO\_PVRUSB2\_SYSFS=y

+# CONFIG\_VIDEO\_PVRUSB2\_DEBUGIFC is not set

+CONFIG\_VIDEO\_EM28XX=m

+CONFIG\_VIDEO\_USBVISION=m

+CONFIG\_VIDEO\_USBVIDEO=m

+CONFIG\_USB\_VICAM=m

+CONFIG\_USB\_IBMCAM=m

+CONFIG\_USB\_KONICAWC=m

+CONFIG\_USB\_QUICKCAM\_MESSENGER=m

+CONFIG\_USB\_ET61X251=m

+CONFIG\_VIDEO\_OVCAMCHIP=m

+CONFIG\_USB\_W9968CF=m

+# CONFIG\_USB\_OV511 is not set

+CONFIG\_USB\_SE401=m

+CONFIG\_USB\_SN9C102=m

+CONFIG\_USB\_STV680=m

+CONFIG\_USB\_ZC0301=m

+CONFIG\_USB\_PWC=m

+# CONFIG\_USB\_PWC\_DEBUG is not set

+CONFIG\_USB\_ZR364XX=m

+CONFIG\_RADIO\_ADAPTERS=y

+CONFIG\_RADIO\_CADET=m

+CONFIG\_RADIO\_RTRACK=m

+CONFIG\_RADIO\_RTRACK2=m

+CONFIG\_RADIO\_AZTECH=m

+CONFIG\_RADIO\_GEMTEK=m

+CONFIG\_RADIO\_GEMTEK\_PCI=m

+CONFIG\_RADIO\_MAXIRADIO=m

+CONFIG\_RADIO\_MAESTRO=m

+CONFIG\_RADIO\_SF16FMI=m

+CONFIG\_RADIO\_SF16FMR2=m

+CONFIG\_RADIO\_TERRATEC=m

+CONFIG\_RADIO\_TRUST=m

+CONFIG\_RADIO\_TYPHOON=m

+# CONFIG\_RADIO\_TYPHOON\_PROC\_FS is not set

+CONFIG\_RADIO\_ZOLTRIX=m

+CONFIG\_USB\_DSBR=m

+CONFIG\_DVB\_CORE=m

+CONFIG\_DVB\_CORE\_ATTACH=y

+CONFIG\_DVB\_CAPTURE\_DRIVERS=y

+

+#

+# Supported SAA7146 based PCI Adapters

+#

+CONFIG\_DVB\_AV7110=m

+CONFIG\_DVB\_AV7110\_OSD=y

+CONFIG\_DVB\_BUDGET=m

+CONFIG\_DVB\_BUDGET\_CI=m

+CONFIG\_DVB\_BUDGET\_AV=m

+CONFIG\_DVB\_BUDGET\_PATCH=m

+

+#

+# Supported USB Adapters

+#

+CONFIG\_DVB\_USB=m

+# CONFIG\_DVB\_USB\_DEBUG is not set

+CONFIG\_DVB\_USB\_A800=m

+CONFIG\_DVB\_USB\_DIBUSB\_MB=m

+CONFIG\_DVB\_USB\_DIBUSB\_MB\_FAULTY=y

+CONFIG\_DVB\_USB\_DIBUSB\_MC=m

+CONFIG\_DVB\_USB\_DIB0700=m

+CONFIG\_DVB\_USB\_UMT\_010=m

+CONFIG\_DVB\_USB\_CXUSB=m

+CONFIG\_DVB\_USB\_M920X=m

+CONFIG\_DVB\_USB\_GL861=m

+CONFIG\_DVB\_USB\_AU6610=m

+CONFIG\_DVB\_USB\_DIGITV=m

+CONFIG\_DVB\_USB\_VP7045=m

+CONFIG\_DVB\_USB\_VP702X=m

+CONFIG\_DVB\_USB\_GP8PSK=m

+CONFIG\_DVB\_USB\_NOVA\_T\_USB2=m

+CONFIG\_DVB\_USB\_TTUSB2=m

+CONFIG\_DVB\_USB\_DTT200U=m

+CONFIG\_DVB\_USB\_OPERA1=m

+CONFIG\_DVB\_USB\_AF9005=m

+CONFIG\_DVB\_USB\_AF9005\_REMOTE=m

+CONFIG\_DVB\_TTUSB\_BUDGET=m

+CONFIG\_DVB\_TTUSB\_DEC=m

+CONFIG\_DVB\_CINERGYT2=m

+CONFIG\_DVB\_CINERGYT2\_TUNING=y

+CONFIG\_DVB\_CINERGYT2\_STREAM\_URB\_COUNT=32

+CONFIG\_DVB\_CINERGYT2\_STREAM\_BUF\_SIZE=512

+CONFIG\_DVB\_CINERGYT2\_QUERY\_INTERVAL=250

+CONFIG\_DVB\_CINERGYT2\_ENABLE\_RC\_INPUT\_DEVICE=y

+CONFIG\_DVB\_CINERGYT2\_RC\_QUERY\_INTERVAL=100

+

+#

+# Supported FlexCopII (B2C2) Adapters

+#

+CONFIG\_DVB\_B2C2\_FLEXCOP=m

+CONFIG\_DVB\_B2C2\_FLEXCOP\_PCI=m

+CONFIG\_DVB\_B2C2\_FLEXCOP\_USB=m

+# CONFIG\_DVB\_B2C2\_FLEXCOP\_DEBUG is not set

+

+#

+# Supported BT878 Adapters

+#

+CONFIG\_DVB\_BT8XX=m

+

+#

+# Supported Pluto2 Adapters

+#

+CONFIG\_DVB\_PLUTO2=m

+

+#

+# Supported DVB Frontends

+#

+

+#

+# Customise DVB Frontends

+#

+# CONFIG\_DVB\_FE\_CUSTOMISE is not set

+

+#

+# DVB-S (satellite) frontends

+#

+CONFIG\_DVB\_STV0299=m

+CONFIG\_DVB\_CX24110=m

+CONFIG\_DVB\_CX24123=m

+CONFIG\_DVB\_TDA8083=m

+CONFIG\_DVB\_MT312=m

+CONFIG\_DVB\_VES1X93=m

+CONFIG\_DVB\_S5H1420=m

+CONFIG\_DVB\_TDA10086=m

+

+#

+# DVB-T (terrestrial) frontends

+#

+CONFIG\_DVB\_SP8870=m

+CONFIG\_DVB\_SP887X=m

+CONFIG\_DVB\_CX22700=m

+CONFIG\_DVB\_CX22702=m

+CONFIG\_DVB\_L64781=m

+CONFIG\_DVB\_TDA1004X=m

+CONFIG\_DVB\_NXT6000=m

+CONFIG\_DVB\_MT352=m

+CONFIG\_DVB\_ZL10353=m

+CONFIG\_DVB\_DIB3000MB=m

+CONFIG\_DVB\_DIB3000MC=m

+CONFIG\_DVB\_DIB7000M=m

+CONFIG\_DVB\_DIB7000P=m

+

+#

+# DVB-C (cable) frontends

+#

+CONFIG\_DVB\_VES1820=m

+CONFIG\_DVB\_TDA10021=m

+CONFIG\_DVB\_TDA10023=m

+CONFIG\_DVB\_STV0297=m

+

+#

+# ATSC (North American/Korean Terrestrial/Cable DTV) frontends

+#

+CONFIG\_DVB\_NXT200X=m

+CONFIG\_DVB\_OR51211=m

+CONFIG\_DVB\_OR51132=m

+CONFIG\_DVB\_BCM3510=m

+CONFIG\_DVB\_LGDT330X=m

+CONFIG\_DVB\_S5H1409=m

+

+#

+# Tuners/PLL support

+#

+CONFIG\_DVB\_PLL=m

+CONFIG\_DVB\_TDA826X=m

+CONFIG\_DVB\_TDA827X=m

+CONFIG\_DVB\_TUNER\_QT1010=m

+CONFIG\_DVB\_TUNER\_MT2060=m

+CONFIG\_DVB\_TUNER\_MT2266=m

+CONFIG\_DVB\_TUNER\_MT2131=m

+CONFIG\_DVB\_TUNER\_DIB0070=m

+

+#

+# Miscellaneous devices

+#

+CONFIG\_DVB\_LNBP21=m

+CONFIG\_DVB\_ISL6421=m

+CONFIG\_DVB\_TUA6100=m

+CONFIG\_VIDEO\_SAA7146=m

+CONFIG\_VIDEO\_SAA7146\_VV=m

+CONFIG\_VIDEO\_TUNER=m

+# CONFIG\_VIDEO\_TUNER\_CUSTOMIZE is not set

+CONFIG\_TUNER\_MT20XX=m

+CONFIG\_TUNER\_TDA8290=m

+CONFIG\_TUNER\_TEA5761=m

+CONFIG\_TUNER\_TEA5767=m

+CONFIG\_TUNER\_SIMPLE=m

+CONFIG\_VIDEOBUF\_GEN=m

+CONFIG\_VIDEOBUF\_DMA\_SG=m

+CONFIG\_VIDEOBUF\_VMALLOC=m

+CONFIG\_VIDEOBUF\_DVB=m

+CONFIG\_VIDEO\_BTCX=m

+CONFIG\_VIDEO\_IR\_I2C=m

+CONFIG\_VIDEO\_IR=m

+CONFIG\_VIDEO\_TVEEPROM=m

+CONFIG\_DAB=y

+CONFIG\_USB\_DABUSB=m

+

+#

+# Graphics support

+#

+CONFIG\_AGP=m

+CONFIG\_AGP\_ALI=m

+CONFIG\_AGP\_ATI=m

+CONFIG\_AGP\_AMD=m

+CONFIG\_AGP\_AMD64=m

+CONFIG\_AGP\_INTEL=m

+CONFIG\_AGP\_NVIDIA=m

+CONFIG\_AGP\_SIS=m

+CONFIG\_AGP\_SWORKS=m

+CONFIG\_AGP\_VIA=m

+CONFIG\_AGP\_EFFICEON=m

+CONFIG\_DRM=m

+CONFIG\_DRM\_TDFX=m

+CONFIG\_DRM\_R128=m

+CONFIG\_DRM\_RADEON=m

+CONFIG\_DRM\_I810=m

+CONFIG\_DRM\_I830=m

+CONFIG\_DRM\_I915=m

+CONFIG\_DRM\_MGA=m

+CONFIG\_DRM\_SIS=m

+CONFIG\_DRM\_VIA=m

+CONFIG\_DRM\_SAVAGE=m

+CONFIG\_VGASTATE=m

+CONFIG\_VIDEO\_OUTPUT\_CONTROL=m

+CONFIG\_FB=y

+CONFIG\_FIRMWARE\_EDID=y

+CONFIG\_FB\_DDC=m

+CONFIG\_FB\_CFB\_FILLRECT=y

+CONFIG\_FB\_CFB\_COPYAREA=y

+CONFIG\_FB\_CFB\_IMAGEBLIT=y

+# CONFIG\_FB\_CFB\_REV\_PIXELS\_IN\_BYTE is not set

+CONFIG\_FB\_SYS\_FILLRECT=m

+CONFIG\_FB\_SYS\_COPYAREA=m

+CONFIG\_FB\_SYS\_IMAGEBLIT=m

+CONFIG\_FB\_SYS\_FOPS=m

+CONFIG\_FB\_DEFERRED\_IO=y

+CONFIG\_FB\_SVGALIB=m

+# CONFIG\_FB\_MACMODES is not set

+CONFIG\_FB\_BACKLIGHT=y

+CONFIG\_FB\_MODE\_HELPERS=y

+CONFIG\_FB\_TILEBLITTING=y

+

+#

+# Frame buffer hardware drivers

+#

+CONFIG\_FB\_CIRRUS=m

+CONFIG\_FB\_PM2=m

+CONFIG\_FB\_PM2\_FIFO\_DISCONNECT=y

+CONFIG\_FB\_CYBER2000=m

+CONFIG\_FB\_ARC=m

+CONFIG\_FB\_ASILIANT=y

+CONFIG\_FB\_IMSTT=y

+CONFIG\_FB\_VGA16=m

+CONFIG\_FB\_UVESA=m

+# CONFIG\_FB\_VESA is not set

+CONFIG\_FB\_EFI=y

+CONFIG\_FB\_IMAC=y

+CONFIG\_FB\_HECUBA=m

+CONFIG\_FB\_HGA=m

+# CONFIG\_FB\_HGA\_ACCEL is not set

+CONFIG\_FB\_S1D13XXX=m

+CONFIG\_FB\_NVIDIA=m

+CONFIG\_FB\_NVIDIA\_I2C=y

+# CONFIG\_FB\_NVIDIA\_DEBUG is not set

+CONFIG\_FB\_NVIDIA\_BACKLIGHT=y

+CONFIG\_FB\_RIVA=m

+CONFIG\_FB\_RIVA\_I2C=y

+# CONFIG\_FB\_RIVA\_DEBUG is not set

+CONFIG\_FB\_RIVA\_BACKLIGHT=y

+CONFIG\_FB\_I810=m

+# CONFIG\_FB\_I810\_GTF is not set

+CONFIG\_FB\_LE80578=m

+CONFIG\_FB\_CARILLO\_RANCH=m

+CONFIG\_FB\_INTEL=m

+# CONFIG\_FB\_INTEL\_DEBUG is not set

+CONFIG\_FB\_INTEL\_I2C=y

+CONFIG\_FB\_MATROX=m

+CONFIG\_FB\_MATROX\_MILLENIUM=y

+CONFIG\_FB\_MATROX\_MYSTIQUE=y

+CONFIG\_FB\_MATROX\_G=y

+CONFIG\_FB\_MATROX\_I2C=m

+CONFIG\_FB\_MATROX\_MAVEN=m

+CONFIG\_FB\_MATROX\_MULTIHEAD=y

+CONFIG\_FB\_RADEON=m

+CONFIG\_FB\_RADEON\_I2C=y

+CONFIG\_FB\_RADEON\_BACKLIGHT=y

+# CONFIG\_FB\_RADEON\_DEBUG is not set

+CONFIG\_FB\_ATY128=m

+CONFIG\_FB\_ATY128\_BACKLIGHT=y

+CONFIG\_FB\_ATY=m

+CONFIG\_FB\_ATY\_CT=y

+CONFIG\_FB\_ATY\_GENERIC\_LCD=y

+CONFIG\_FB\_ATY\_GX=y

+CONFIG\_FB\_ATY\_BACKLIGHT=y

+CONFIG\_FB\_S3=m

+CONFIG\_FB\_SAVAGE=m

+CONFIG\_FB\_SAVAGE\_I2C=y

+CONFIG\_FB\_SAVAGE\_ACCEL=y

+CONFIG\_FB\_SIS=m

+CONFIG\_FB\_SIS\_300=y

+CONFIG\_FB\_SIS\_315=y

+CONFIG\_FB\_NEOMAGIC=m

+CONFIG\_FB\_KYRO=m

+CONFIG\_FB\_3DFX=m

+# CONFIG\_FB\_3DFX\_ACCEL is not set

+CONFIG\_FB\_VOODOO1=m

+CONFIG\_FB\_VT8623=m

+CONFIG\_FB\_CYBLA=m

+CONFIG\_FB\_TRIDENT=m

+# CONFIG\_FB\_TRIDENT\_ACCEL is not set

+CONFIG\_FB\_ARK=m

+CONFIG\_FB\_PM3=m

+CONFIG\_FB\_GEODE=y

+CONFIG\_FB\_GEODE\_LX=m

+CONFIG\_FB\_GEODE\_GX=m

+# CONFIG\_FB\_GEODE\_GX\_SET\_FBSIZE is not set

+CONFIG\_FB\_GEODE\_GX1=m

+CONFIG\_FB\_SM501=m

+# CONFIG\_FB\_VIRTUAL is not set

+CONFIG\_BACKLIGHT\_LCD\_SUPPORT=y

+CONFIG\_LCD\_CLASS\_DEVICE=m

+CONFIG\_LCD\_LTV350QV=m

+CONFIG\_BACKLIGHT\_CLASS\_DEVICE=y

+CONFIG\_BACKLIGHT\_CORGI=m

+CONFIG\_BACKLIGHT\_PROGEAR=m

+CONFIG\_BACKLIGHT\_CARILLO\_RANCH=m

+

+#

+# Display device support

+#

+CONFIG\_DISPLAY\_SUPPORT=m

+

+#

+# Display hardware drivers

+#

+

+#

+# Console display driver support

+#

+CONFIG\_VGA\_CONSOLE=y

+# CONFIG\_VGACON\_SOFT\_SCROLLBACK is not set

+CONFIG\_VIDEO\_SELECT=y

+CONFIG\_MDA\_CONSOLE=m

+CONFIG\_DUMMY\_CONSOLE=y

+CONFIG\_FRAMEBUFFER\_CONSOLE=m

+# CONFIG\_FRAMEBUFFER\_CONSOLE\_DETECT\_PRIMARY is not set

+# CONFIG\_FRAMEBUFFER\_CONSOLE\_ROTATION is not set

+# CONFIG\_FONTS is not set

+CONFIG\_FONT\_8x8=y

+CONFIG\_FONT\_8x16=y

+# CONFIG\_LOGO is not set

+

+#

+# Sound

+#

+CONFIG\_SOUND=m

+

+#

+# Advanced Linux Sound Architecture

+#

+# CONFIG\_SND is not set

+

+#

+# Open Sound System

+#

+# CONFIG\_SOUND\_PRIME is not set

+CONFIG\_AC97\_BUS=m

+CONFIG\_HID\_SUPPORT=y

+CONFIG\_HID=m

+# CONFIG\_HID\_DEBUG is not set

+CONFIG\_HIDRAW=y

+

+#

+# USB Input Devices

+#

+CONFIG\_USB\_HID=m

+CONFIG\_USB\_HIDINPUT\_POWERBOOK=y

+# CONFIG\_HID\_FF is not set

+CONFIG\_USB\_HIDDEV=y

+

+#

+# USB HID Boot Protocol drivers

+#

+CONFIG\_USB\_KBD=m

+CONFIG\_USB\_MOUSE=m

+CONFIG\_USB\_SUPPORT=y

+CONFIG\_USB\_ARCH\_HAS\_HCD=y

+CONFIG\_USB\_ARCH\_HAS\_OHCI=y

+CONFIG\_USB\_ARCH\_HAS\_EHCI=y

+CONFIG\_USB=m

+# CONFIG\_USB\_DEBUG is not set

+

+#

+# Miscellaneous USB options

+#

+CONFIG\_USB\_DEVICEFS=y

+# CONFIG\_USB\_DEVICE\_CLASS is not set

+# CONFIG\_USB\_DYNAMIC\_MINORS is not set

+CONFIG\_USB\_SUSPEND=y

+CONFIG\_USB\_PERSIST=y

+# CONFIG\_USB\_OTG is not set

+

+#

+# USB Host Controller Drivers

+#

+CONFIG\_USB\_EHCI\_HCD=m

+CONFIG\_USB\_EHCI\_SPLIT\_ISO=y

+CONFIG\_USB\_EHCI\_ROOT\_HUB\_TT=y

+CONFIG\_USB\_EHCI\_TT\_NEWSCHED=y

+CONFIG\_USB\_ISP116X\_HCD=m

+CONFIG\_USB\_OHCI\_HCD=m

+# CONFIG\_USB\_OHCI\_HCD\_SSB is not set

+# CONFIG\_USB\_OHCI\_BIG\_ENDIAN\_DESC is not set

+# CONFIG\_USB\_OHCI\_BIG\_ENDIAN\_MMIO is not set

+CONFIG\_USB\_OHCI\_LITTLE\_ENDIAN=y

+CONFIG\_USB\_UHCI\_HCD=m

+CONFIG\_USB\_U132\_HCD=m

+CONFIG\_USB\_SL811\_HCD=m

+CONFIG\_USB\_SL811\_CS=m

+CONFIG\_USB\_R8A66597\_HCD=m

+

+#

+# USB Device Class drivers

+#

+CONFIG\_USB\_ACM=m

+CONFIG\_USB\_PRINTER=m

+

+#

+# NOTE: USB\_STORAGE enables SCSI, and 'SCSI disk support'

+#

+

+#

+# may also be needed; see USB\_STORAGE Help for more information

+#

+CONFIG\_USB\_STORAGE=m

+# CONFIG\_USB\_STORAGE\_DEBUG is not set

+CONFIG\_USB\_STORAGE\_DATAFAB=y

+CONFIG\_USB\_STORAGE\_FREECOM=y

+CONFIG\_USB\_STORAGE\_ISD200=y

+CONFIG\_USB\_STORAGE\_DPCM=y

+CONFIG\_USB\_STORAGE\_USBAT=y

+CONFIG\_USB\_STORAGE\_SDDR09=y

+CONFIG\_USB\_STORAGE\_SDDR55=y

+CONFIG\_USB\_STORAGE\_JUMPSHOT=y

+CONFIG\_USB\_STORAGE\_ALAUDA=y

+CONFIG\_USB\_STORAGE\_KARMA=y

+CONFIG\_USB\_LIBUSUAL=y

+

+#

+# USB Imaging devices

+#

+CONFIG\_USB\_MDC800=m

+CONFIG\_USB\_MICROTEK=m

+CONFIG\_USB\_MON=y

+

+#

+# USB port drivers

+#

+CONFIG\_USB\_USS720=m

+

+#

+# USB Serial Converter support

+#

+CONFIG\_USB\_SERIAL=m

+CONFIG\_USB\_SERIAL\_GENERIC=y

+CONFIG\_USB\_SERIAL\_AIRCABLE=m

+CONFIG\_USB\_SERIAL\_AIRPRIME=m

+CONFIG\_USB\_SERIAL\_ARK3116=m

+CONFIG\_USB\_SERIAL\_BELKIN=m

+CONFIG\_USB\_SERIAL\_CH341=m

+CONFIG\_USB\_SERIAL\_WHITEHEAT=m

+CONFIG\_USB\_SERIAL\_DIGI\_ACCELEPORT=m

+CONFIG\_USB\_SERIAL\_CP2101=m

+CONFIG\_USB\_SERIAL\_CYPRESS\_M8=m

+CONFIG\_USB\_SERIAL\_EMPEG=m

+CONFIG\_USB\_SERIAL\_FTDI\_SIO=m

+CONFIG\_USB\_SERIAL\_FUNSOFT=m

+CONFIG\_USB\_SERIAL\_VISOR=m

+CONFIG\_USB\_SERIAL\_IPAQ=m

+# CONFIG\_USB\_SERIAL\_IR is not set

+CONFIG\_USB\_SERIAL\_EDGEPORT=m

+CONFIG\_USB\_SERIAL\_EDGEPORT\_TI=m

+CONFIG\_USB\_SERIAL\_GARMIN=m

+CONFIG\_USB\_SERIAL\_IPW=m

+CONFIG\_USB\_SERIAL\_KEYSPAN\_PDA=m

+CONFIG\_USB\_SERIAL\_KEYSPAN=m

+CONFIG\_USB\_SERIAL\_KEYSPAN\_MPR=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA28=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA28X=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA28XA=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA28XB=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA19=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA18X=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA19W=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA19QW=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA19QI=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA49W=y

+CONFIG\_USB\_SERIAL\_KEYSPAN\_USA49WLC=y

+CONFIG\_USB\_SERIAL\_KLSI=m

+CONFIG\_USB\_SERIAL\_KOBIL\_SCT=m

+CONFIG\_USB\_SERIAL\_MCT\_U232=m

+CONFIG\_USB\_SERIAL\_MOS7720=m

+CONFIG\_USB\_SERIAL\_MOS7840=m

+CONFIG\_USB\_SERIAL\_NAVMAN=m

+CONFIG\_USB\_SERIAL\_PL2303=m

+CONFIG\_USB\_SERIAL\_OTI6858=m

+CONFIG\_USB\_SERIAL\_HP4X=m

+CONFIG\_USB\_SERIAL\_SAFE=m

+# CONFIG\_USB\_SERIAL\_SAFE\_PADDED is not set

+CONFIG\_USB\_SERIAL\_SIERRAWIRELESS=m

+CONFIG\_USB\_SERIAL\_TI=m

+CONFIG\_USB\_SERIAL\_CYBERJACK=m

+CONFIG\_USB\_SERIAL\_XIRCOM=m

+CONFIG\_USB\_SERIAL\_OPTION=m

+CONFIG\_USB\_SERIAL\_OMNINET=m

+CONFIG\_USB\_SERIAL\_DEBUG=m

+CONFIG\_USB\_EZUSB=y

+

+#

+# USB Miscellaneous drivers

+#

+CONFIG\_USB\_EMI62=m

+CONFIG\_USB\_EMI26=m

+CONFIG\_USB\_ADUTUX=m

+CONFIG\_USB\_AUERSWALD=m

+CONFIG\_USB\_RIO500=m

+CONFIG\_USB\_LEGOTOWER=m

+CONFIG\_USB\_LCD=m

+CONFIG\_USB\_BERRY\_CHARGE=m

+CONFIG\_USB\_LED=m

+CONFIG\_USB\_CYPRESS\_CY7C63=m

+CONFIG\_USB\_CYTHERM=m

+CONFIG\_USB\_PHIDGET=m

+CONFIG\_USB\_PHIDGETKIT=m

+CONFIG\_USB\_PHIDGETMOTORCONTROL=m

+CONFIG\_USB\_PHIDGETSERVO=m

+CONFIG\_USB\_IDMOUSE=m

+CONFIG\_USB\_FTDI\_ELAN=m

+CONFIG\_USB\_APPLEDISPLAY=m

+CONFIG\_USB\_SISUSBVGA=m

+# CONFIG\_USB\_SISUSBVGA\_CON is not set

+CONFIG\_USB\_LD=m

+CONFIG\_USB\_TRANCEVIBRATOR=m

+CONFIG\_USB\_IOWARRIOR=m

+# CONFIG\_USB\_TEST is not set

+

+#

+# USB DSL modem support

+#

+CONFIG\_USB\_ATM=m

+CONFIG\_USB\_SPEEDTOUCH=m

+CONFIG\_USB\_CXACRU=m

+CONFIG\_USB\_UEAGLEATM=m

+CONFIG\_USB\_XUSBATM=m

+

+#

+# USB Gadget Support

+#

+CONFIG\_USB\_GADGET=m

+# CONFIG\_USB\_GADGET\_DEBUG is not set

+# CONFIG\_USB\_GADGET\_DEBUG\_FILES is not set

+# CONFIG\_USB\_GADGET\_DEBUG\_FS is not set

+CONFIG\_USB\_GADGET\_SELECTED=y

+# CONFIG\_USB\_GADGET\_AMD5536UDC is not set

+# CONFIG\_USB\_GADGET\_ATMEL\_USBA is not set

+# CONFIG\_USB\_GADGET\_FSL\_USB2 is not set

+CONFIG\_USB\_GADGET\_NET2280=y

+CONFIG\_USB\_NET2280=m

+# CONFIG\_USB\_GADGET\_PXA2XX is not set

+# CONFIG\_USB\_GADGET\_M66592 is not set

+# CONFIG\_USB\_GADGET\_GOKU is not set

+# CONFIG\_USB\_GADGET\_LH7A40X is not set

+# CONFIG\_USB\_GADGET\_OMAP is not set

+# CONFIG\_USB\_GADGET\_S3C2410 is not set

+# CONFIG\_USB\_GADGET\_AT91 is not set

+# CONFIG\_USB\_GADGET\_DUMMY\_HCD is not set

+CONFIG\_USB\_GADGET\_DUALSPEED=y

+CONFIG\_USB\_ZERO=m

+CONFIG\_USB\_ETH=m

+CONFIG\_USB\_ETH\_RNDIS=y

+CONFIG\_USB\_GADGETFS=m

+CONFIG\_USB\_FILE\_STORAGE=m

+# CONFIG\_USB\_FILE\_STORAGE\_TEST is not set

+CONFIG\_USB\_G\_SERIAL=m

+# CONFIG\_USB\_MIDI\_GADGET is not set

+CONFIG\_MMC=m

+# CONFIG\_MMC\_DEBUG is not set

+# CONFIG\_MMC\_UNSAFE\_RESUME is not set

+

+#

+# MMC/SD Card Drivers

+#

+CONFIG\_MMC\_BLOCK=m

+CONFIG\_MMC\_BLOCK\_BOUNCE=y

+CONFIG\_SDIO\_UART=m

+

+#

+# MMC/SD Host Controller Drivers

+#

+CONFIG\_MMC\_SDHCI=m

+CONFIG\_MMC\_RICOH\_MMC=m

+CONFIG\_MMC\_WBSD=m

+CONFIG\_MMC\_TIFM\_SD=m

+CONFIG\_NEW\_LEDS=y

+CONFIG\_LEDS\_CLASS=m

+

+#

+# LED drivers

+#

+CONFIG\_LEDS\_NET48XX=m

+CONFIG\_LEDS\_WRAP=m

+

+#

+# LED Triggers

+#

+CONFIG\_LEDS\_TRIGGERS=y

+CONFIG\_LEDS\_TRIGGER\_TIMER=m

+CONFIG\_LEDS\_TRIGGER\_IDE\_DISK=y

+CONFIG\_LEDS\_TRIGGER\_HEARTBEAT=m

+CONFIG\_INFINIBAND=m

+CONFIG\_INFINIBAND\_USER\_MAD=m

+CONFIG\_INFINIBAND\_USER\_ACCESS=m

+CONFIG\_INFINIBAND\_USER\_MEM=y

+CONFIG\_INFINIBAND\_ADDR\_TRANS=y

+CONFIG\_INFINIBAND\_MTHCA=m

+CONFIG\_INFINIBAND\_MTHCA\_DEBUG=y

+CONFIG\_INFINIBAND\_AMSO1100=m

+CONFIG\_INFINIBAND\_AMSO1100\_DEBUG=y

+CONFIG\_INFINIBAND\_CXGB3=m

+# CONFIG\_INFINIBAND\_CXGB3\_DEBUG is not set

+CONFIG\_MLX4\_INFINIBAND=m

+CONFIG\_INFINIBAND\_IPOIB=m

+CONFIG\_INFINIBAND\_IPOIB\_CM=y

+CONFIG\_INFINIBAND\_IPOIB\_DEBUG=y

+# CONFIG\_INFINIBAND\_IPOIB\_DEBUG\_DATA is not set

+CONFIG\_INFINIBAND\_SRP=m

+CONFIG\_INFINIBAND\_ISER=m

+CONFIG\_EDAC=y

+

+#

+# Reporting subsystems

+#

+# CONFIG\_EDAC\_DEBUG is not set

+CONFIG\_EDAC\_MM\_EDAC=m

+# CONFIG\_EDAC\_AMD76X is not set

+CONFIG\_EDAC\_E7XXX=m

+CONFIG\_EDAC\_E752X=m

+CONFIG\_EDAC\_I82875P=m

+CONFIG\_EDAC\_I82975X=m

+CONFIG\_EDAC\_I3000=m

+CONFIG\_EDAC\_I82860=m

+CONFIG\_EDAC\_R82600=m

+CONFIG\_EDAC\_I5000=m

+CONFIG\_RTC\_LIB=m

+CONFIG\_RTC\_CLASS=m

+

+#

+# RTC interfaces

+#

+CONFIG\_RTC\_INTF\_SYSFS=y

+CONFIG\_RTC\_INTF\_PROC=y

+CONFIG\_RTC\_INTF\_DEV=y

+CONFIG\_RTC\_INTF\_DEV\_UIE\_EMUL=y

+CONFIG\_RTC\_DRV\_TEST=m

+

+#

+# I2C RTC drivers

+#

+CONFIG\_RTC\_DRV\_DS1307=m

+CONFIG\_RTC\_DRV\_DS1374=m

+CONFIG\_RTC\_DRV\_DS1672=m

+CONFIG\_RTC\_DRV\_MAX6900=m

+CONFIG\_RTC\_DRV\_RS5C372=m

+CONFIG\_RTC\_DRV\_ISL1208=m

+CONFIG\_RTC\_DRV\_X1205=m

+CONFIG\_RTC\_DRV\_PCF8563=m

+CONFIG\_RTC\_DRV\_PCF8583=m

+CONFIG\_RTC\_DRV\_M41T80=m

+CONFIG\_RTC\_DRV\_M41T80\_WDT=y

+

+#

+# SPI RTC drivers

+#

+CONFIG\_RTC\_DRV\_RS5C348=m

+CONFIG\_RTC\_DRV\_MAX6902=m

+

+#

+# Platform RTC drivers

+#

+# CONFIG\_RTC\_DRV\_CMOS is not set

+CONFIG\_RTC\_DRV\_DS1553=m

+CONFIG\_RTC\_DRV\_STK17TA8=m

+CONFIG\_RTC\_DRV\_DS1742=m

+CONFIG\_RTC\_DRV\_M48T86=m

+CONFIG\_RTC\_DRV\_M48T59=m

+CONFIG\_RTC\_DRV\_V3020=m

+

+#

+# on-CPU RTC drivers

+#

+CONFIG\_DMADEVICES=y

+

+#

+# DMA Devices

+#

+CONFIG\_INTEL\_IOATDMA=m

+CONFIG\_DMA\_ENGINE=y

+

+#

+# DMA Clients

+#

+CONFIG\_NET\_DMA=y

+CONFIG\_DCA=m

+CONFIG\_AUXDISPLAY=y

+CONFIG\_KS0108=m

+CONFIG\_KS0108\_PORT=0x378

+CONFIG\_KS0108\_DELAY=2

+CONFIG\_CFAG12864B=m

+CONFIG\_CFAG12864B\_RATE=20

+CONFIG\_VIRTUALIZATION=y

+CONFIG\_KVM=m

+CONFIG\_KVM\_INTEL=m

+CONFIG\_KVM\_AMD=m

+CONFIG\_LGUEST=m

+

+#

+# Userspace I/O

+#

+CONFIG\_UIO=m

+CONFIG\_UIO\_CIF=m

+

+#

+# Firmware Drivers

+#

+CONFIG\_EDD=y

+CONFIG\_EFI\_VARS=y

+CONFIG\_DELL\_RBU=m

+CONFIG\_DCDBAS=m

+CONFIG\_DMIID=y

+

+#

+# File systems

+#

+CONFIG\_EXT2\_FS=m

+CONFIG\_EXT2\_FS\_XATTR=y

+CONFIG\_EXT2\_FS\_POSIX\_ACL=y

+CONFIG\_EXT2\_FS\_SECURITY=y

+# CONFIG\_EXT2\_FS\_XIP is not set

+CONFIG\_EXT3\_FS=m

+CONFIG\_EXT3\_FS\_XATTR=y

+CONFIG\_EXT3\_FS\_POSIX\_ACL=y

+CONFIG\_EXT3\_FS\_SECURITY=y

+# CONFIG\_EXT4DEV\_FS is not set

+CONFIG\_JBD=m

+# CONFIG\_JBD\_DEBUG is not set

+CONFIG\_FS\_MBCACHE=m

+CONFIG\_REISERFS\_FS=m

+# CONFIG\_REISERFS\_CHECK is not set

+# CONFIG\_REISERFS\_PROC\_INFO is not set

+CONFIG\_REISERFS\_FS\_XATTR=y

+CONFIG\_REISERFS\_FS\_POSIX\_ACL=y

+CONFIG\_REISERFS\_FS\_SECURITY=y

+CONFIG\_JFS\_FS=m

+CONFIG\_JFS\_POSIX\_ACL=y

+CONFIG\_JFS\_SECURITY=y

+# CONFIG\_JFS\_DEBUG is not set

+CONFIG\_JFS\_STATISTICS=y

+CONFIG\_FS\_POSIX\_ACL=y

+CONFIG\_XFS\_FS=m

+CONFIG\_XFS\_QUOTA=y

+CONFIG\_XFS\_SECURITY=y

+CONFIG\_XFS\_POSIX\_ACL=y

+CONFIG\_XFS\_RT=y

+CONFIG\_GFS2\_FS=m

+CONFIG\_GFS2\_FS\_LOCKING\_NOLOCK=m

+CONFIG\_GFS2\_FS\_LOCKING\_DLM=m

+CONFIG\_OCFS2\_FS=m

+CONFIG\_OCFS2\_DEBUG\_MASKLOG=y

+# CONFIG\_OCFS2\_DEBUG\_FS is not set

+CONFIG\_MINIX\_FS=m

+CONFIG\_ROMFS\_FS=m

+CONFIG\_INOTIFY=y

+CONFIG\_INOTIFY\_USER=y

+CONFIG\_QUOTA=y

+CONFIG\_QUOTA\_NETLINK\_INTERFACE=y

+CONFIG\_PRINT\_QUOTA\_WARNING=y

+CONFIG\_QFMT\_V1=m

+CONFIG\_QFMT\_V2=m

+CONFIG\_QUOTACTL=y

+CONFIG\_DNOTIFY=y

+CONFIG\_AUTOFS\_FS=m

+CONFIG\_AUTOFS4\_FS=m

+CONFIG\_FUSE\_FS=m

+CONFIG\_GENERIC\_ACL=y

+

+#

+# CD-ROM/DVD Filesystems

+#

+CONFIG\_ISO9660\_FS=m

+CONFIG\_JOLIET=y

+CONFIG\_ZISOFS=y

+CONFIG\_UDF\_FS=m

+CONFIG\_UDF\_NLS=y

+

+#

+# DOS/FAT/NT Filesystems

+#

+CONFIG\_FAT\_FS=m

+CONFIG\_MSDOS\_FS=m

+CONFIG\_VFAT\_FS=m

+CONFIG\_FAT\_DEFAULT\_CODEPAGE=437

+CONFIG\_FAT\_DEFAULT\_IOCHARSET="iso8859-1"

+CONFIG\_NTFS\_FS=m

+# CONFIG\_NTFS\_DEBUG is not set

+# CONFIG\_NTFS\_RW is not set

+

+#

+# Pseudo filesystems

+#

+CONFIG\_PROC\_FS=y

+CONFIG\_PROC\_KCORE=y

+CONFIG\_PROC\_VMCORE=y

+CONFIG\_PROC\_SYSCTL=y

+CONFIG\_SYSFS=y

+CONFIG\_TMPFS=y

+CONFIG\_TMPFS\_POSIX\_ACL=y

+# CONFIG\_HUGETLBFS is not set

+# CONFIG\_HUGETLB\_PAGE is not set

+CONFIG\_CONFIGFS\_FS=m

+

+#

+# Miscellaneous filesystems

+#

+CONFIG\_ADFS\_FS=m

+# CONFIG\_ADFS\_FS\_RW is not set

+CONFIG\_AFFS\_FS=m

+CONFIG\_ECRYPT\_FS=m

+CONFIG\_HFS\_FS=m

+CONFIG\_HFSPLUS\_FS=m

+CONFIG\_BEFS\_FS=m

+# CONFIG\_BEFS\_DEBUG is not set

+CONFIG\_BFS\_FS=m

+CONFIG\_EFS\_FS=m

+CONFIG\_JFFS2\_FS=m

+CONFIG\_JFFS2\_FS\_DEBUG=0

+CONFIG\_JFFS2\_FS\_WRITEBUFFER=y

+# CONFIG\_JFFS2\_FS\_WBUF\_VERIFY is not set

+# CONFIG\_JFFS2\_SUMMARY is not set

+# CONFIG\_JFFS2\_FS\_XATTR is not set

+CONFIG\_JFFS2\_COMPRESSION\_OPTIONS=y

+CONFIG\_JFFS2\_ZLIB=y

+CONFIG\_JFFS2\_LZO=y

+CONFIG\_JFFS2\_RTIME=y

+# CONFIG\_JFFS2\_RUBIN is not set

+# CONFIG\_JFFS2\_CMODE\_NONE is not set

+# CONFIG\_JFFS2\_CMODE\_PRIORITY is not set

+# CONFIG\_JFFS2\_CMODE\_SIZE is not set

+CONFIG\_JFFS2\_CMODE\_FAVOURLZO=y

+CONFIG\_CRAMFS=y

+CONFIG\_VXFS\_FS=m

+CONFIG\_HPFS\_FS=m

+CONFIG\_QNX4FS\_FS=m

+CONFIG\_SYSV\_FS=m

+CONFIG\_UFS\_FS=m

+# CONFIG\_UFS\_FS\_WRITE is not set

+# CONFIG\_UFS\_DEBUG is not set

+CONFIG\_NETWORK\_FILESYSTEMS=y

+CONFIG\_NFS\_FS=m

+CONFIG\_NFS\_V3=y

+CONFIG\_NFS\_V3\_ACL=y

+CONFIG\_NFS\_V4=y

+CONFIG\_NFS\_DIRECTIO=y

+CONFIG\_NFSD=m

+CONFIG\_NFSD\_V2\_ACL=y

+CONFIG\_NFSD\_V3=y

+CONFIG\_NFSD\_V3\_ACL=y

+CONFIG\_NFSD\_V4=y

+CONFIG\_NFSD\_TCP=y

+CONFIG\_LOCKD=m

+CONFIG\_LOCKD\_V4=y

+CONFIG\_EXPORTFS=m

+CONFIG\_NFS\_ACL\_SUPPORT=m

+CONFIG\_NFS\_COMMON=y

+CONFIG\_SUNRPC=m

+CONFIG\_SUNRPC\_GSS=m

+CONFIG\_SUNRPC\_XPRT\_RDMA=m

+# CONFIG\_SUNRPC\_BIND34 is not set

+CONFIG\_RPCSEC\_GSS\_KRB5=m

+CONFIG\_RPCSEC\_GSS\_SPKM3=m

+CONFIG\_SMB\_FS=m

+# CONFIG\_SMB\_NLS\_DEFAULT is not set

+CONFIG\_CIFS=m

+# CONFIG\_CIFS\_STATS is not set

+CONFIG\_CIFS\_WEAK\_PW\_HASH=y

+# CONFIG\_CIFS\_XATTR is not set

+# CONFIG\_CIFS\_DEBUG2 is not set

+CONFIG\_CIFS\_EXPERIMENTAL=y

+CONFIG\_CIFS\_UPCALL=y

+CONFIG\_NCP\_FS=m

+CONFIG\_NCPFS\_PACKET\_SIGNING=y

+CONFIG\_NCPFS\_IOCTL\_LOCKING=y

+CONFIG\_NCPFS\_STRONG=y

+CONFIG\_NCPFS\_NFS\_NS=y

+CONFIG\_NCPFS\_OS2\_NS=y

+# CONFIG\_NCPFS\_SMALLDOS is not set

+CONFIG\_NCPFS\_NLS=y

+CONFIG\_NCPFS\_EXTRAS=y

+CONFIG\_CODA\_FS=m

+# CONFIG\_CODA\_FS\_OLD\_API is not set

+CONFIG\_AFS\_FS=m

+# CONFIG\_AFS\_DEBUG is not set

+CONFIG\_9P\_FS=m

+

+#

+# Partition Types

+#

+CONFIG\_PARTITION\_ADVANCED=y

+CONFIG\_ACORN\_PARTITION=y

+# CONFIG\_ACORN\_PARTITION\_CUMANA is not set

+# CONFIG\_ACORN\_PARTITION\_EESOX is not set

+CONFIG\_ACORN\_PARTITION\_ICS=y

+# CONFIG\_ACORN\_PARTITION\_ADFS is not set

+# CONFIG\_ACORN\_PARTITION\_POWERTEC is not set

+CONFIG\_ACORN\_PARTITION\_RISCIX=y

+CONFIG\_OSF\_PARTITION=y

+CONFIG\_AMIGA\_PARTITION=y

+CONFIG\_ATARI\_PARTITION=y

+CONFIG\_MAC\_PARTITION=y

+CONFIG\_MSDOS\_PARTITION=y

+CONFIG\_BSD\_DISKLABEL=y

+CONFIG\_MINIX\_SUBPARTITION=y

+CONFIG\_SOLARIS\_X86\_PARTITION=y

+CONFIG\_UNIXWARE\_DISKLABEL=y

+CONFIG\_LDM\_PARTITION=y

+# CONFIG\_LDM\_DEBUG is not set

+CONFIG\_SGI\_PARTITION=y

+CONFIG\_ULTRIX\_PARTITION=y

+CONFIG\_SUN\_PARTITION=y

+CONFIG\_KARMA\_PARTITION=y

+CONFIG\_EFI\_PARTITION=y

+CONFIG\_SYSV68\_PARTITION=y

+CONFIG\_NLS=y

+CONFIG\_NLS\_DEFAULT="cp437"

+CONFIG\_NLS\_CODEPAGE\_437=m

+CONFIG\_NLS\_CODEPAGE\_737=m

+CONFIG\_NLS\_CODEPAGE\_775=m

+CONFIG\_NLS\_CODEPAGE\_850=m

+CONFIG\_NLS\_CODEPAGE\_852=m

+CONFIG\_NLS\_CODEPAGE\_855=m

+CONFIG\_NLS\_CODEPAGE\_857=m

+CONFIG\_NLS\_CODEPAGE\_860=m

+CONFIG\_NLS\_CODEPAGE\_861=m

+CONFIG\_NLS\_CODEPAGE\_862=m

+CONFIG\_NLS\_CODEPAGE\_863=m

+CONFIG\_NLS\_CODEPAGE\_864=m

+CONFIG\_NLS\_CODEPAGE\_865=m

+CONFIG\_NLS\_CODEPAGE\_866=m

+CONFIG\_NLS\_CODEPAGE\_869=m

+CONFIG\_NLS\_CODEPAGE\_936=m

+CONFIG\_NLS\_CODEPAGE\_950=m

+CONFIG\_NLS\_CODEPAGE\_932=m

+CONFIG\_NLS\_CODEPAGE\_949=m

+CONFIG\_NLS\_CODEPAGE\_874=m

+CONFIG\_NLS\_ISO8859\_8=m

+CONFIG\_NLS\_CODEPAGE\_1250=m

+CONFIG\_NLS\_CODEPAGE\_1251=m

+CONFIG\_NLS\_ASCII=m

+CONFIG\_NLS\_ISO8859\_1=m

+CONFIG\_NLS\_ISO8859\_2=m

+CONFIG\_NLS\_ISO8859\_3=m

+CONFIG\_NLS\_ISO8859\_4=m

+CONFIG\_NLS\_ISO8859\_5=m

+CONFIG\_NLS\_ISO8859\_6=m

+CONFIG\_NLS\_ISO8859\_7=m

+CONFIG\_NLS\_ISO8859\_9=m

+CONFIG\_NLS\_ISO8859\_13=m

+CONFIG\_NLS\_ISO8859\_14=m

+CONFIG\_NLS\_ISO8859\_15=m

+CONFIG\_NLS\_KOI8\_R=m

+CONFIG\_NLS\_KOI8\_U=m

+CONFIG\_NLS\_UTF8=m

+CONFIG\_DLM=m

+# CONFIG\_DLM\_DEBUG is not set

+CONFIG\_INSTRUMENTATION=y

+CONFIG\_PROFILING=y

+CONFIG\_OPROFILE=m

+CONFIG\_KPROBES=y

+# CONFIG\_MARKERS is not set

+

+#

+# Kernel hacking

+#

+CONFIG\_TRACE\_IRQFLAGS\_SUPPORT=y

+CONFIG\_PRINTK\_TIME=y

+# CONFIG\_ENABLE\_WARN\_DEPRECATED is not set

+# CONFIG\_ENABLE\_MUST\_CHECK is not set

+CONFIG\_MAGIC\_SYSRQ=y

+CONFIG\_UNUSED\_SYMBOLS=y

+CONFIG\_DEBUG\_FS=y

+# CONFIG\_HEADERS\_CHECK is not set

+CONFIG\_DEBUG\_KERNEL=y

+# CONFIG\_DEBUG\_SHIRQ is not set

+CONFIG\_DETECT\_SOFTLOCKUP=y

+CONFIG\_SCHED\_DEBUG=y

+# CONFIG\_SCHEDSTATS is not set

+CONFIG\_TIMER\_STATS=y

+# CONFIG\_SLUB\_DEBUG\_ON is not set

+# CONFIG\_DEBUG\_RT\_MUTEXES is not set

+# CONFIG\_RT\_MUTEX\_TESTER is not set

+# CONFIG\_DEBUG\_SPINLOCK is not set

+# CONFIG\_DEBUG\_MUTEXES is not set

+# CONFIG\_DEBUG\_LOCK\_ALLOC is not set

+# CONFIG\_PROVE\_LOCKING is not set

+# CONFIG\_LOCK\_STAT is not set

+# CONFIG\_DEBUG\_SPINLOCK\_SLEEP is not set

+# CONFIG\_DEBUG\_LOCKING\_API\_SELFTESTS is not set

+# CONFIG\_DEBUG\_KOBJECT is not set

+# CONFIG\_DEBUG\_HIGHMEM is not set

+CONFIG\_DEBUG\_BUGVERBOSE=y

+CONFIG\_DEBUG\_INFO=y

+# CONFIG\_DEBUG\_VM is not set

+# CONFIG\_DEBUG\_LIST is not set

+# CONFIG\_DEBUG\_SG is not set

+# CONFIG\_FRAME\_POINTER is not set

+# CONFIG\_FORCED\_INLINING is not set

+# CONFIG\_BOOT\_PRINTK\_DELAY is not set

+# CONFIG\_RCU\_TORTURE\_TEST is not set

+# CONFIG\_LKDTM is not set

+# CONFIG\_FAULT\_INJECTION is not set

+# CONFIG\_SAMPLES is not set

+CONFIG\_EARLY\_PRINTK=y

+# CONFIG\_DEBUG\_STACKOVERFLOW is not set

+# CONFIG\_DEBUG\_STACK\_USAGE is not set

+

+#

+# Page alloc debug is incompatible with Software Suspend on i386

+#

+CONFIG\_DEBUG\_RODATA=y

+# CONFIG\_4KSTACKS is not set

+CONFIG\_X86\_FIND\_SMP\_CONFIG=y

+CONFIG\_X86\_MPPARSE=y

+CONFIG\_DOUBLEFAULT=y

+

+#

+# Security options

+#

+CONFIG\_KEYS=y

+# CONFIG\_KEYS\_DEBUG\_PROC\_KEYS is not set

+CONFIG\_SECURITY=y

+CONFIG\_SECURITY\_NETWORK=y

+# CONFIG\_SECURITY\_NETWORK\_XFRM is not set

+CONFIG\_SECURITY\_CAPABILITIES=y

+# CONFIG\_SECURITY\_FILE\_CAPABILITIES is not set

+CONFIG\_SECURITY\_SELINUX=y

+CONFIG\_SECURITY\_SELINUX\_BOOTPARAM=y

+CONFIG\_SECURITY\_SELINUX\_BOOTPARAM\_VALUE=0

+CONFIG\_SECURITY\_SELINUX\_DISABLE=y

+CONFIG\_SECURITY\_SELINUX\_DEVELOP=y

+CONFIG\_SECURITY\_SELINUX\_AVC\_STATS=y

+CONFIG\_SECURITY\_SELINUX\_CHECKREQPROT\_VALUE=1

+# CONFIG\_SECURITY\_SELINUX\_ENABLE\_SECMARK\_DEFAULT is not set

+# CONFIG\_SECURITY\_SELINUX\_POLICYDB\_VERSION\_MAX is not set

+CONFIG\_XOR\_BLOCKS=m

+CONFIG\_ASYNC\_CORE=m

+CONFIG\_ASYNC\_MEMCPY=m

+CONFIG\_ASYNC\_XOR=m

+CONFIG\_CRYPTO=y

+CONFIG\_CRYPTO\_ALGAPI=y

+CONFIG\_CRYPTO\_ABLKCIPHER=m

+CONFIG\_CRYPTO\_AEAD=m

+CONFIG\_CRYPTO\_BLKCIPHER=m

+CONFIG\_CRYPTO\_HASH=y

+CONFIG\_CRYPTO\_MANAGER=y

+CONFIG\_CRYPTO\_HMAC=y

+CONFIG\_CRYPTO\_XCBC=m

+CONFIG\_CRYPTO\_NULL=m

+CONFIG\_CRYPTO\_MD4=m

+CONFIG\_CRYPTO\_MD5=y

+CONFIG\_CRYPTO\_SHA1=m

+CONFIG\_CRYPTO\_SHA256=m

+CONFIG\_CRYPTO\_SHA512=m

+CONFIG\_CRYPTO\_WP512=m

+CONFIG\_CRYPTO\_TGR192=m

+CONFIG\_CRYPTO\_GF128MUL=m

+CONFIG\_CRYPTO\_ECB=m

+CONFIG\_CRYPTO\_CBC=m

+CONFIG\_CRYPTO\_PCBC=m

+CONFIG\_CRYPTO\_LRW=m

+CONFIG\_CRYPTO\_XTS=m

+CONFIG\_CRYPTO\_CRYPTD=m

+CONFIG\_CRYPTO\_DES=m

+CONFIG\_CRYPTO\_FCRYPT=m

+CONFIG\_CRYPTO\_BLOWFISH=m

+CONFIG\_CRYPTO\_TWOFISH=m

+CONFIG\_CRYPTO\_TWOFISH\_COMMON=m

+CONFIG\_CRYPTO\_TWOFISH\_586=m

+CONFIG\_CRYPTO\_SERPENT=m

+CONFIG\_CRYPTO\_AES=m

+CONFIG\_CRYPTO\_AES\_586=m

+CONFIG\_CRYPTO\_CAST5=m

+CONFIG\_CRYPTO\_CAST6=m

+CONFIG\_CRYPTO\_TEA=m

+CONFIG\_CRYPTO\_ARC4=m

+CONFIG\_CRYPTO\_KHAZAD=m

+CONFIG\_CRYPTO\_ANUBIS=m

+CONFIG\_CRYPTO\_SEED=m

+CONFIG\_CRYPTO\_DEFLATE=m

+CONFIG\_CRYPTO\_MICHAEL\_MIC=m

+CONFIG\_CRYPTO\_CRC32C=m

+CONFIG\_CRYPTO\_CAMELLIA=m

+CONFIG\_CRYPTO\_TEST=m

+CONFIG\_CRYPTO\_AUTHENC=m

+CONFIG\_CRYPTO\_HW=y

+CONFIG\_CRYPTO\_DEV\_PADLOCK=y

+CONFIG\_CRYPTO\_DEV\_PADLOCK\_AES=m

+CONFIG\_CRYPTO\_DEV\_PADLOCK\_SHA=m

+CONFIG\_CRYPTO\_DEV\_GEODE=m

+

+#

+# Library routines

+#

+CONFIG\_BITREVERSE=y

+CONFIG\_CRC\_CCITT=m

+CONFIG\_CRC16=m

+CONFIG\_CRC\_ITU\_T=m

+CONFIG\_CRC32=y

+CONFIG\_CRC7=m

+CONFIG\_LIBCRC32C=m

+CONFIG\_AUDIT\_GENERIC=y

+CONFIG\_ZLIB\_INFLATE=y

+CONFIG\_ZLIB\_DEFLATE=m

+CONFIG\_LZO\_COMPRESS=m

+CONFIG\_LZO\_DECOMPRESS=m

+CONFIG\_GENERIC\_ALLOCATOR=y

+CONFIG\_REED\_SOLOMON=m

+CONFIG\_REED\_SOLOMON\_DEC16=y

+CONFIG\_TEXTSEARCH=y

+CONFIG\_TEXTSEARCH\_KMP=m

+CONFIG\_TEXTSEARCH\_BM=m

+CONFIG\_TEXTSEARCH\_FSM=m

+CONFIG\_PLIST=y

+CONFIG\_HAS\_IOMEM=y

+CONFIG\_HAS\_IOPORT=y

+CONFIG\_HAS\_DMA=y

+CONFIG\_CHECK\_SIGNATURE=y

diff -Nur linux-2.6.24/fs/proc/proc\_misc.c linux-2.6.24-casio/fs/proc/proc\_misc.c

--- linux-2.6.24/fs/proc/proc\_misc.c 2008-01-24 22:58:37.000000000 +0000

+++ linux-2.6.24-casio/fs/proc/proc\_misc.c 2009-06-02 04:11:04.000000000 +0100

@@ -216,6 +216,48 @@

#undef K

}

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#define CASIO\_MAX\_CURSOR\_LINES\_EVENTS 1

+

+static int casio\_open(struct inode \*inode, struct file \*file)

+{

+ return 0;

+}

+static int casio\_read(char \*filp, char \*buf, size\_t count, loff\_t \*f\_pos)

+{

+ char buffer[CASIO\_MSG\_SIZE];

+ unsigned int len=0,k,i;

+ struct casio\_event\_log \*log=NULL;

+ buffer[0]='\0';

+ log=get\_casio\_event\_log();

+ if(log){

+ if(log->cursor < log->lines){

+ k=(log->lines > (log->cursor + CASIO\_MAX\_CURSOR\_LINES\_EVENTS))?(log->cursor + CASIO\_MAX\_CURSOR\_LINES\_EVENTS):(log->lines);

+ for(i=log->cursor; i<k;i++){

+ len = snprintf(buffer, count, "%s%d,%llu,%s\n",

+ buffer,

+ log->casio\_event[i].action,

+ log->casio\_event[i].timestamp,

+ log->casio\_event[i].msg);

+ }

+ log->cursor=k;

+ }

+ if(len)

+ copy\_to\_user(buf,buffer,len);

+

+ }

+ return len;

+}

+static int casio\_release(struct inode \*inode, struct file \*file)

+{

+ return 0;

+}

+static const struct file\_operations proc\_casio\_operations = {

+ .open = casio\_open,

+ .read = casio\_read,

+ .release = casio\_release,

+};

+#endif

extern struct seq\_operations fragmentation\_op;

static int fragmentation\_open(struct inode \*inode, struct file \*file)

{

@@ -768,4 +810,14 @@

entry->proc\_fops = &proc\_sysrq\_trigger\_operations;

}

#endif

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ {

+ struct proc\_dir\_entry \*casio\_entry;

+ casio\_entry = create\_proc\_entry("casio\_event", 0666, &proc\_root);

+ if (casio\_entry){

+ casio\_entry->proc\_fops = &proc\_casio\_operations;

+ casio\_entry->data=NULL;

+ }

+ }

+#endif

}

diff -Nur linux-2.6.24/fs/proc/proc\_misc.c~ linux-2.6.24-casio/fs/proc/proc\_misc.c~

--- linux-2.6.24/fs/proc/proc\_misc.c~ 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/fs/proc/proc\_misc.c~ 2009-06-02 04:11:04.000000000 +0100

@@ -0,0 +1,823 @@

+/\*

+ \* linux/fs/proc/proc\_misc.c

+ \*

+ \* linux/fs/proc/array.c

+ \* Copyright (C) 1992 by Linus Torvalds

+ \* based on ideas by Darren Senn

+ \*

+ \* This used to be the part of array.c. See the rest of history and credits

+ \* there. I took this into a separate file and switched the thing to generic

+ \* proc\_file\_inode\_operations, leaving in array.c only per-process stuff.

+ \* Inumbers allocation made dynamic (via create\_proc\_entry()). AV, May 1999.

+ \*

+ \* Changes:

+ \* Fulton Green : Encapsulated position metric calculations.

+ \* <kernel@FultonGreen.com>

+ \*/

+

+#include <linux/types.h>

+#include <linux/errno.h>

+#include <linux/time.h>

+#include <linux/kernel.h>

+#include <linux/kernel\_stat.h>

+#include <linux/fs.h>

+#include <linux/tty.h>

+#include <linux/string.h>

+#include <linux/mman.h>

+#include <linux/proc\_fs.h>

+#include <linux/ioport.h>

+#include <linux/mm.h>

+#include <linux/mmzone.h>

+#include <linux/pagemap.h>

+#include <linux/swap.h>

+#include <linux/slab.h>

+#include <linux/smp.h>

+#include <linux/signal.h>

+#include <linux/module.h>

+#include <linux/init.h>

+#include <linux/seq\_file.h>

+#include <linux/times.h>

+#include <linux/profile.h>

+#include <linux/utsname.h>

+#include <linux/blkdev.h>

+#include <linux/hugetlb.h>

+#include <linux/jiffies.h>

+#include <linux/sysrq.h>

+#include <linux/vmalloc.h>

+#include <linux/crash\_dump.h>

+#include <linux/pid\_namespace.h>

+#include <asm/uaccess.h>

+#include <asm/pgtable.h>

+#include <asm/io.h>

+#include <asm/tlb.h>

+#include <asm/div64.h>

+#include "internal.h"

+

+#define LOAD\_INT(x) ((x) >> FSHIFT)

+#define LOAD\_FRAC(x) LOAD\_INT(((x) & (FIXED\_1-1)) \* 100)

+/\*

+ \* Warning: stuff below (imported functions) assumes that its output will fit

+ \* into one page. For some of those functions it may be wrong. Moreover, we

+ \* have a way to deal with that gracefully. Right now I used straightforward

+ \* wrappers, but this needs further analysis wrt potential overflows.

+ \*/

+extern int get\_hardware\_list(char \*);

+extern int get\_stram\_list(char \*);

+extern int get\_filesystem\_list(char \*);

+extern int get\_exec\_domain\_list(char \*);

+extern int get\_dma\_list(char \*);

+

+static int proc\_calc\_metrics(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, int len)

+{

+ if (len <= off+count) \*eof = 1;

+ \*start = page + off;

+ len -= off;

+ if (len>count) len = count;

+ if (len<0) len = 0;

+ return len;

+}

+

+static int loadavg\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int a, b, c;

+ int len;

+

+ a = avenrun[0] + (FIXED\_1/200);

+ b = avenrun[1] + (FIXED\_1/200);

+ c = avenrun[2] + (FIXED\_1/200);

+ len = sprintf(page,"%d.%02d %d.%02d %d.%02d %ld/%d %d\n",

+ LOAD\_INT(a), LOAD\_FRAC(a),

+ LOAD\_INT(b), LOAD\_FRAC(b),

+ LOAD\_INT(c), LOAD\_FRAC(c),

+ nr\_running(), nr\_threads,

+ task\_active\_pid\_ns(current)->last\_pid);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+static int uptime\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ struct timespec uptime;

+ struct timespec idle;

+ int len;

+ cputime\_t idletime = cputime\_add(init\_task.utime, init\_task.stime);

+

+ do\_posix\_clock\_monotonic\_gettime(&uptime);

+ monotonic\_to\_bootbased(&uptime);

+ cputime\_to\_timespec(idletime, &idle);

+ len = sprintf(page,"%lu.%02lu %lu.%02lu\n",

+ (unsigned long) uptime.tv\_sec,

+ (uptime.tv\_nsec / (NSEC\_PER\_SEC / 100)),

+ (unsigned long) idle.tv\_sec,

+ (idle.tv\_nsec / (NSEC\_PER\_SEC / 100)));

+

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+static int meminfo\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ struct sysinfo i;

+ int len;

+ unsigned long committed;

+ unsigned long allowed;

+ struct vmalloc\_info vmi;

+ long cached;

+

+/\*

+ \* display in kilobytes.

+ \*/

+#define K(x) ((x) << (PAGE\_SHIFT - 10))

+ si\_meminfo(&i);

+ si\_swapinfo(&i);

+ committed = atomic\_read(&vm\_committed\_space);

+ allowed = ((totalram\_pages - hugetlb\_total\_pages())

+ \* sysctl\_overcommit\_ratio / 100) + total\_swap\_pages;

+

+ cached = global\_page\_state(NR\_FILE\_PAGES) -

+ total\_swapcache\_pages - i.bufferram;

+ if (cached < 0)

+ cached = 0;

+

+ get\_vmalloc\_info(&vmi);

+

+ /\*

+ \* Tagged format, for easy grepping and expansion.

+ \*/

+ len = sprintf(page,

+ "MemTotal: %8lu kB\n"

+ "MemFree: %8lu kB\n"

+ "Buffers: %8lu kB\n"

+ "Cached: %8lu kB\n"

+ "SwapCached: %8lu kB\n"

+ "Active: %8lu kB\n"

+ "Inactive: %8lu kB\n"

+#ifdef CONFIG\_HIGHMEM

+ "HighTotal: %8lu kB\n"

+ "HighFree: %8lu kB\n"

+ "LowTotal: %8lu kB\n"

+ "LowFree: %8lu kB\n"

+#endif

+ "SwapTotal: %8lu kB\n"

+ "SwapFree: %8lu kB\n"

+ "Dirty: %8lu kB\n"

+ "Writeback: %8lu kB\n"

+ "AnonPages: %8lu kB\n"

+ "Mapped: %8lu kB\n"

+ "Slab: %8lu kB\n"

+ "SReclaimable: %8lu kB\n"

+ "SUnreclaim: %8lu kB\n"

+ "PageTables: %8lu kB\n"

+ "NFS\_Unstable: %8lu kB\n"

+ "Bounce: %8lu kB\n"

+ "CommitLimit: %8lu kB\n"

+ "Committed\_AS: %8lu kB\n"

+ "VmallocTotal: %8lu kB\n"

+ "VmallocUsed: %8lu kB\n"

+ "VmallocChunk: %8lu kB\n",

+ K(i.totalram),

+ K(i.freeram),

+ K(i.bufferram),

+ K(cached),

+ K(total\_swapcache\_pages),

+ K(global\_page\_state(NR\_ACTIVE)),

+ K(global\_page\_state(NR\_INACTIVE)),

+#ifdef CONFIG\_HIGHMEM

+ K(i.totalhigh),

+ K(i.freehigh),

+ K(i.totalram-i.totalhigh),

+ K(i.freeram-i.freehigh),

+#endif

+ K(i.totalswap),

+ K(i.freeswap),

+ K(global\_page\_state(NR\_FILE\_DIRTY)),

+ K(global\_page\_state(NR\_WRITEBACK)),

+ K(global\_page\_state(NR\_ANON\_PAGES)),

+ K(global\_page\_state(NR\_FILE\_MAPPED)),

+ K(global\_page\_state(NR\_SLAB\_RECLAIMABLE) +

+ global\_page\_state(NR\_SLAB\_UNRECLAIMABLE)),

+ K(global\_page\_state(NR\_SLAB\_RECLAIMABLE)),

+ K(global\_page\_state(NR\_SLAB\_UNRECLAIMABLE)),

+ K(global\_page\_state(NR\_PAGETABLE)),

+ K(global\_page\_state(NR\_UNSTABLE\_NFS)),

+ K(global\_page\_state(NR\_BOUNCE)),

+ K(allowed),

+ K(committed),

+ (unsigned long)VMALLOC\_TOTAL >> 10,

+ vmi.used >> 10,

+ vmi.largest\_chunk >> 10

+ );

+

+ len += hugetlb\_report\_meminfo(page + len);

+

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+#undef K

+}

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#define CASIO\_MAX\_CURSOR\_LINES\_EVENTS 1

+

+static int casio\_open(struct inode \*inode, struct file \*file)

+{

+ return 0;

+}

+static int casio\_read(char \*filp, char \*buf, size\_t count, loff\_t \*f\_pos)

+{

+ char buffer[CASIO\_MSG\_SIZE];

+ unsigned int len=0,k,i;

+ struct casio\_event\_log \*log=NULL;

+ buffer[0]='\0';

+ log=get\_casio\_event\_log();

+ if(log){

+ if(log->cursor < log->lines){

+ k=(log->lines > (log->cursor + CASIO\_MAX\_CURSOR\_LINES\_EVENTS))?(log->cursor + CASIO\_MAX\_CURSOR\_LINES\_EVENTS):(log->lines);

+ for(i=log->cursor; i<k;i++){

+ len = snprintf(buffer, count, "%s%d,%llu,%s\n",

+ buffer,

+ log->casio\_event[i].action,

+ log->casio\_event[i].timestamp,

+ log->casio\_event[i].msg);

+ }

+ log->cursor=k;

+ }

+ if(len)

+ copy\_to\_user(buf,buffer,len);

+

+ }

+ return len;

+}

+static int casio\_release(struct inode \*inode, struct file \*file)

+{

+ return 0;

+}

+static const struct file\_operations proc\_casio\_operations = {

+ .open = casio\_open,

+ .read = casio\_read,

+ .release = casio\_release,

+};

+#endif

+extern struct seq\_operations fragmentation\_op;

+static int fragmentation\_open(struct inode \*inode, struct file \*file)

+{

+ (void)inode;

+ return seq\_open(file, &fragmentation\_op);

+}

+

+static const struct file\_operations fragmentation\_file\_operations = {

+ .open = fragmentation\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+extern struct seq\_operations pagetypeinfo\_op;

+static int pagetypeinfo\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &pagetypeinfo\_op);

+}

+

+static const struct file\_operations pagetypeinfo\_file\_ops = {

+ .open = pagetypeinfo\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+extern struct seq\_operations zoneinfo\_op;

+static int zoneinfo\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &zoneinfo\_op);

+}

+

+static const struct file\_operations proc\_zoneinfo\_file\_operations = {

+ .open = zoneinfo\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+static int version\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len;

+

+ len = snprintf(page, PAGE\_SIZE, linux\_proc\_banner,

+ utsname()->sysname,

+ utsname()->release,

+ utsname()->version);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+extern struct seq\_operations cpuinfo\_op;

+static int cpuinfo\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &cpuinfo\_op);

+}

+

+static const struct file\_operations proc\_cpuinfo\_operations = {

+ .open = cpuinfo\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+static int devinfo\_show(struct seq\_file \*f, void \*v)

+{

+ int i = \*(loff\_t \*) v;

+

+ if (i < CHRDEV\_MAJOR\_HASH\_SIZE) {

+ if (i == 0)

+ seq\_printf(f, "Character devices:\n");

+ chrdev\_show(f, i);

+ }

+#ifdef CONFIG\_BLOCK

+ else {

+ i -= CHRDEV\_MAJOR\_HASH\_SIZE;

+ if (i == 0)

+ seq\_printf(f, "\nBlock devices:\n");

+ blkdev\_show(f, i);

+ }

+#endif

+ return 0;

+}

+

+static void \*devinfo\_start(struct seq\_file \*f, loff\_t \*pos)

+{

+ if (\*pos < (BLKDEV\_MAJOR\_HASH\_SIZE + CHRDEV\_MAJOR\_HASH\_SIZE))

+ return pos;

+ return NULL;

+}

+

+static void \*devinfo\_next(struct seq\_file \*f, void \*v, loff\_t \*pos)

+{

+ (\*pos)++;

+ if (\*pos >= (BLKDEV\_MAJOR\_HASH\_SIZE + CHRDEV\_MAJOR\_HASH\_SIZE))

+ return NULL;

+ return pos;

+}

+

+static void devinfo\_stop(struct seq\_file \*f, void \*v)

+{

+ /\* Nothing to do \*/

+}

+

+static struct seq\_operations devinfo\_ops = {

+ .start = devinfo\_start,

+ .next = devinfo\_next,

+ .stop = devinfo\_stop,

+ .show = devinfo\_show

+};

+

+static int devinfo\_open(struct inode \*inode, struct file \*filp)

+{

+ return seq\_open(filp, &devinfo\_ops);

+}

+

+static const struct file\_operations proc\_devinfo\_operations = {

+ .open = devinfo\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+extern struct seq\_operations vmstat\_op;

+static int vmstat\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &vmstat\_op);

+}

+static const struct file\_operations proc\_vmstat\_file\_operations = {

+ .open = vmstat\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+#ifdef CONFIG\_PROC\_HARDWARE

+static int hardware\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len = get\_hardware\_list(page);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+#endif

+

+#ifdef CONFIG\_STRAM\_PROC

+static int stram\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len = get\_stram\_list(page);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+#endif

+

+#ifdef CONFIG\_BLOCK

+extern struct seq\_operations partitions\_op;

+static int partitions\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &partitions\_op);

+}

+static const struct file\_operations proc\_partitions\_operations = {

+ .open = partitions\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+extern struct seq\_operations diskstats\_op;

+static int diskstats\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &diskstats\_op);

+}

+static const struct file\_operations proc\_diskstats\_operations = {

+ .open = diskstats\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+#endif

+

+#ifdef CONFIG\_MODULES

+extern struct seq\_operations modules\_op;

+static int modules\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &modules\_op);

+}

+static const struct file\_operations proc\_modules\_operations = {

+ .open = modules\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+#endif

+

+#ifdef CONFIG\_SLABINFO

+static int slabinfo\_open(struct inode \*inode, struct file \*file)

+{

+ return seq\_open(file, &slabinfo\_op);

+}

+static const struct file\_operations proc\_slabinfo\_operations = {

+ .open = slabinfo\_open,

+ .read = seq\_read,

+ .write = slabinfo\_write,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+#ifdef CONFIG\_DEBUG\_SLAB\_LEAK

+extern struct seq\_operations slabstats\_op;

+static int slabstats\_open(struct inode \*inode, struct file \*file)

+{

+ unsigned long \*n = kzalloc(PAGE\_SIZE, GFP\_KERNEL);

+ int ret = -ENOMEM;

+ if (n) {

+ ret = seq\_open(file, &slabstats\_op);

+ if (!ret) {

+ struct seq\_file \*m = file->private\_data;

+ \*n = PAGE\_SIZE / (2 \* sizeof(unsigned long));

+ m->private = n;

+ n = NULL;

+ }

+ kfree(n);

+ }

+ return ret;

+}

+

+static const struct file\_operations proc\_slabstats\_operations = {

+ .open = slabstats\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release\_private,

+};

+#endif

+#endif

+

+static int show\_stat(struct seq\_file \*p, void \*v)

+{

+ int i;

+ unsigned long jif;

+ cputime64\_t user, nice, system, idle, iowait, irq, softirq, steal;

+ cputime64\_t guest;

+ u64 sum = 0;

+ struct timespec boottime;

+ unsigned int \*per\_irq\_sum;

+

+ per\_irq\_sum = kzalloc(sizeof(unsigned int)\*NR\_IRQS, GFP\_KERNEL);

+ if (!per\_irq\_sum)

+ return -ENOMEM;

+

+ user = nice = system = idle = iowait =

+ irq = softirq = steal = cputime64\_zero;

+ guest = cputime64\_zero;

+ getboottime(&boottime);

+ jif = boottime.tv\_sec;

+

+ for\_each\_possible\_cpu(i) {

+ int j;

+

+ user = cputime64\_add(user, kstat\_cpu(i).cpustat.user);

+ nice = cputime64\_add(nice, kstat\_cpu(i).cpustat.nice);

+ system = cputime64\_add(system, kstat\_cpu(i).cpustat.system);

+ idle = cputime64\_add(idle, kstat\_cpu(i).cpustat.idle);

+ iowait = cputime64\_add(iowait, kstat\_cpu(i).cpustat.iowait);

+ irq = cputime64\_add(irq, kstat\_cpu(i).cpustat.irq);

+ softirq = cputime64\_add(softirq, kstat\_cpu(i).cpustat.softirq);

+ steal = cputime64\_add(steal, kstat\_cpu(i).cpustat.steal);

+ guest = cputime64\_add(guest, kstat\_cpu(i).cpustat.guest);

+ for (j = 0; j < NR\_IRQS; j++) {

+ unsigned int temp = kstat\_cpu(i).irqs[j];

+ sum += temp;

+ per\_irq\_sum[j] += temp;

+ }

+ }

+

+ seq\_printf(p, "cpu %llu %llu %llu %llu %llu %llu %llu %llu %llu\n",

+ (unsigned long long)cputime64\_to\_clock\_t(user),

+ (unsigned long long)cputime64\_to\_clock\_t(nice),

+ (unsigned long long)cputime64\_to\_clock\_t(system),

+ (unsigned long long)cputime64\_to\_clock\_t(idle),

+ (unsigned long long)cputime64\_to\_clock\_t(iowait),

+ (unsigned long long)cputime64\_to\_clock\_t(irq),

+ (unsigned long long)cputime64\_to\_clock\_t(softirq),

+ (unsigned long long)cputime64\_to\_clock\_t(steal),

+ (unsigned long long)cputime64\_to\_clock\_t(guest));

+ for\_each\_online\_cpu(i) {

+

+ /\* Copy values here to work around gcc-2.95.3, gcc-2.96 \*/

+ user = kstat\_cpu(i).cpustat.user;

+ nice = kstat\_cpu(i).cpustat.nice;

+ system = kstat\_cpu(i).cpustat.system;

+ idle = kstat\_cpu(i).cpustat.idle;

+ iowait = kstat\_cpu(i).cpustat.iowait;

+ irq = kstat\_cpu(i).cpustat.irq;

+ softirq = kstat\_cpu(i).cpustat.softirq;

+ steal = kstat\_cpu(i).cpustat.steal;

+ guest = kstat\_cpu(i).cpustat.guest;

+ seq\_printf(p,

+ "cpu%d %llu %llu %llu %llu %llu %llu %llu %llu %llu\n",

+ i,

+ (unsigned long long)cputime64\_to\_clock\_t(user),

+ (unsigned long long)cputime64\_to\_clock\_t(nice),

+ (unsigned long long)cputime64\_to\_clock\_t(system),

+ (unsigned long long)cputime64\_to\_clock\_t(idle),

+ (unsigned long long)cputime64\_to\_clock\_t(iowait),

+ (unsigned long long)cputime64\_to\_clock\_t(irq),

+ (unsigned long long)cputime64\_to\_clock\_t(softirq),

+ (unsigned long long)cputime64\_to\_clock\_t(steal),

+ (unsigned long long)cputime64\_to\_clock\_t(guest));

+ }

+ seq\_printf(p, "intr %llu", (unsigned long long)sum);

+

+ for (i = 0; i < NR\_IRQS; i++)

+ seq\_printf(p, " %u", per\_irq\_sum[i]);

+

+ seq\_printf(p,

+ "\nctxt %llu\n"

+ "btime %lu\n"

+ "processes %lu\n"

+ "procs\_running %lu\n"

+ "procs\_blocked %lu\n",

+ nr\_context\_switches(),

+ (unsigned long)jif,

+ total\_forks,

+ nr\_running(),

+ nr\_iowait());

+

+ kfree(per\_irq\_sum);

+ return 0;

+}

+

+static int stat\_open(struct inode \*inode, struct file \*file)

+{

+ unsigned size = 4096 \* (1 + num\_possible\_cpus() / 32);

+ char \*buf;

+ struct seq\_file \*m;

+ int res;

+

+ /\* don't ask for more than the kmalloc() max size, currently 128 KB \*/

+ if (size > 128 \* 1024)

+ size = 128 \* 1024;

+ buf = kmalloc(size, GFP\_KERNEL);

+ if (!buf)

+ return -ENOMEM;

+

+ res = single\_open(file, show\_stat, NULL);

+ if (!res) {

+ m = file->private\_data;

+ m->buf = buf;

+ m->size = size;

+ } else

+ kfree(buf);

+ return res;

+}

+static const struct file\_operations proc\_stat\_operations = {

+ .open = stat\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = single\_release,

+};

+

+/\*

+ \* /proc/interrupts

+ \*/

+static void \*int\_seq\_start(struct seq\_file \*f, loff\_t \*pos)

+{

+ return (\*pos <= NR\_IRQS) ? pos : NULL;

+}

+

+static void \*int\_seq\_next(struct seq\_file \*f, void \*v, loff\_t \*pos)

+{

+ (\*pos)++;

+ if (\*pos > NR\_IRQS)

+ return NULL;

+ return pos;

+}

+

+static void int\_seq\_stop(struct seq\_file \*f, void \*v)

+{

+ /\* Nothing to do \*/

+}

+

+

+extern int show\_interrupts(struct seq\_file \*f, void \*v); /\* In arch code \*/

+static struct seq\_operations int\_seq\_ops = {

+ .start = int\_seq\_start,

+ .next = int\_seq\_next,

+ .stop = int\_seq\_stop,

+ .show = show\_interrupts

+};

+

+static int interrupts\_open(struct inode \*inode, struct file \*filp)

+{

+ return seq\_open(filp, &int\_seq\_ops);

+}

+

+static const struct file\_operations proc\_interrupts\_operations = {

+ .open = interrupts\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+static int filesystems\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len = get\_filesystem\_list(page);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+static int cmdline\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len;

+

+ len = sprintf(page, "%s\n", saved\_command\_line);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+static int locks\_open(struct inode \*inode, struct file \*filp)

+{

+ return seq\_open(filp, &locks\_seq\_operations);

+}

+

+static const struct file\_operations proc\_locks\_operations = {

+ .open = locks\_open,

+ .read = seq\_read,

+ .llseek = seq\_lseek,

+ .release = seq\_release,

+};

+

+static int execdomains\_read\_proc(char \*page, char \*\*start, off\_t off,

+ int count, int \*eof, void \*data)

+{

+ int len = get\_exec\_domain\_list(page);

+ return proc\_calc\_metrics(page, start, off, count, eof, len);

+}

+

+#ifdef CONFIG\_MAGIC\_SYSRQ

+/\*

+ \* writing 'C' to /proc/sysrq-trigger is like sysrq-C

+ \*/

+static ssize\_t write\_sysrq\_trigger(struct file \*file, const char \_\_user \*buf,

+ size\_t count, loff\_t \*ppos)

+{

+ if (count) {

+ char c;

+

+ if (get\_user(c, buf))

+ return -EFAULT;

+ \_\_handle\_sysrq(c, NULL, 0);

+ }

+ return count;

+}

+

+static const struct file\_operations proc\_sysrq\_trigger\_operations = {

+ .write = write\_sysrq\_trigger,

+};

+#endif

+

+struct proc\_dir\_entry \*proc\_root\_kcore;

+

+void create\_seq\_entry(char \*name, mode\_t mode, const struct file\_operations \*f)

+{

+ struct proc\_dir\_entry \*entry;

+ entry = create\_proc\_entry(name, mode, NULL);

+ if (entry)

+ entry->proc\_fops = f;

+}

+

+void \_\_init proc\_misc\_init(void)

+{

+ static struct {

+ char \*name;

+ int (\*read\_proc)(char\*,char\*\*,off\_t,int,int\*,void\*);

+ } \*p, simple\_ones[] = {

+ {"loadavg", loadavg\_read\_proc},

+ {"uptime", uptime\_read\_proc},

+ {"meminfo", meminfo\_read\_proc},

+ {"version", version\_read\_proc},

+#ifdef CONFIG\_PROC\_HARDWARE

+ {"hardware", hardware\_read\_proc},

+#endif

+#ifdef CONFIG\_STRAM\_PROC

+ {"stram", stram\_read\_proc},

+#endif

+ {"filesystems", filesystems\_read\_proc},

+ {"cmdline", cmdline\_read\_proc},

+ {"execdomains", execdomains\_read\_proc},

+ {NULL,}

+ };

+ for (p = simple\_ones; p->name; p++)

+ create\_proc\_read\_entry(p->name, 0, NULL, p->read\_proc, NULL);

+

+ proc\_symlink("mounts", NULL, "self/mounts");

+

+ /\* And now for trickier ones \*/

+#ifdef CONFIG\_PRINTK

+ {

+ struct proc\_dir\_entry \*entry;

+ entry = create\_proc\_entry("kmsg", S\_IRUSR, &proc\_root);

+ if (entry)

+ entry->proc\_fops = &proc\_kmsg\_operations;

+ }

+#endif

+ create\_seq\_entry("locks", 0, &proc\_locks\_operations);

+ create\_seq\_entry("devices", 0, &proc\_devinfo\_operations);

+ create\_seq\_entry("cpuinfo", 0, &proc\_cpuinfo\_operations);

+#ifdef CONFIG\_BLOCK

+ create\_seq\_entry("partitions", 0, &proc\_partitions\_operations);

+#endif

+ create\_seq\_entry("stat", 0, &proc\_stat\_operations);

+ create\_seq\_entry("interrupts", 0, &proc\_interrupts\_operations);

+#ifdef CONFIG\_SLABINFO

+ create\_seq\_entry("slabinfo",S\_IWUSR|S\_IRUGO,&proc\_slabinfo\_operations);

+#ifdef CONFIG\_DEBUG\_SLAB\_LEAK

+ create\_seq\_entry("slab\_allocators", 0 ,&proc\_slabstats\_operations);

+#endif

+#endif

+ create\_seq\_entry("buddyinfo",S\_IRUGO, &fragmentation\_file\_operations);

+ create\_seq\_entry("pagetypeinfo", S\_IRUGO, &pagetypeinfo\_file\_ops);

+ create\_seq\_entry("vmstat",S\_IRUGO, &proc\_vmstat\_file\_operations);

+ create\_seq\_entry("zoneinfo",S\_IRUGO, &proc\_zoneinfo\_file\_operations);

+#ifdef CONFIG\_BLOCK

+ create\_seq\_entry("diskstats", 0, &proc\_diskstats\_operations);

+#endif

+#ifdef CONFIG\_MODULES

+ create\_seq\_entry("modules", 0, &proc\_modules\_operations);

+#endif

+#ifdef CONFIG\_SCHEDSTATS

+ create\_seq\_entry("schedstat", 0, &proc\_schedstat\_operations);

+#endif

+#ifdef CONFIG\_PROC\_KCORE

+ proc\_root\_kcore = create\_proc\_entry("kcore", S\_IRUSR, NULL);

+ if (proc\_root\_kcore) {

+ proc\_root\_kcore->proc\_fops = &proc\_kcore\_operations;

+ proc\_root\_kcore->size =

+ (size\_t)high\_memory - PAGE\_OFFSET + PAGE\_SIZE;

+ }

+#endif

+#ifdef CONFIG\_PROC\_VMCORE

+ proc\_vmcore = create\_proc\_entry("vmcore", S\_IRUSR, NULL);

+ if (proc\_vmcore)

+ proc\_vmcore->proc\_fops = &proc\_vmcore\_operations;

+#endif

+#ifdef CONFIG\_MAGIC\_SYSRQ

+ {

+ struct proc\_dir\_entry \*entry;

+ entry = create\_proc\_entry("sysrq-trigger", S\_IWUSR, NULL);

+ if (entry)

+ entry->proc\_fops = &proc\_sysrq\_trigger\_operations;

+ }

+#endif

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ {

+ struct proc\_dir\_entry \*casio\_entry;

+ casio\_entry = create\_proc\_entry("casio\_event", 0666, &proc\_root);

+ if (casio\_entry){

+ casio\_entry->proc\_fops = &proc\_casio\_operations;

+ casio\_entry->data=NULL;

+ }

+ }

+#endif

+}

diff -Nur linux-2.6.24/include/linux/sched.h linux-2.6.24-casio/include/linux/sched.h

--- linux-2.6.24/include/linux/sched.h 2008-01-24 22:58:37.000000000 +0000

+++ linux-2.6.24-casio/include/linux/sched.h 2009-06-02 04:11:04.000000000 +0100

@@ -38,10 +38,19 @@

/\* SCHED\_ISO: reserved but not implemented yet \*/

#define SCHED\_IDLE 5

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#define SCHED\_CASIO 6

+#endif

+

#ifdef \_\_KERNEL\_\_

struct sched\_param {

int sched\_priority;

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ unsigned int casio\_id;

+ unsigned long long deadline;

+#endif

};

#include <asm/param.h> /\* for HZ \*/

@@ -1178,6 +1187,11 @@

int make\_it\_fail;

#endif

struct prop\_local\_single dirties;

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ unsigned int casio\_id;

+ unsigned long long deadline;

+#endif

};

/\*

@@ -1989,4 +2003,32 @@

#endif /\* \_\_KERNEL\_\_ \*/

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+

+#define CASIO\_MSG\_SIZE 400

+#define CASIO\_MAX\_EVENT\_LINES 10000

+

+#define CASIO\_ENQUEUE 1

+#define CASIO\_DEQUEUE 2

+#define CASIO\_CONTEXT\_SWITCH 3

+#define CASIO\_MSG 4

+

+struct casio\_event{

+ int action;

+ unsigned long long timestamp;

+ char msg[CASIO\_MSG\_SIZE];

+};

+

+struct casio\_event\_log{

+ struct casio\_event casio\_event[CASIO\_MAX\_EVENT\_LINES];

+ unsigned long lines;

+ unsigned long cursor;

+};

+void init\_casio\_event\_log();

+struct casio\_event\_log \* get\_casio\_event\_log();

+void register\_casio\_event(unsigned long long t, char \*m, int a);

+

+#endif

+

#endif

diff -Nur linux-2.6.24/include/linux/sched.h~ linux-2.6.24-casio/include/linux/sched.h~

--- linux-2.6.24/include/linux/sched.h~ 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/include/linux/sched.h~ 2009-06-02 04:11:04.000000000 +0100

@@ -0,0 +1,2034 @@

+#ifndef \_LINUX\_SCHED\_H

+#define \_LINUX\_SCHED\_H

+

+/\*

+ \* cloning flags:

+ \*/

+#define CSIGNAL 0x000000ff /\* signal mask to be sent at exit \*/

+#define CLONE\_VM 0x00000100 /\* set if VM shared between processes \*/

+#define CLONE\_FS 0x00000200 /\* set if fs info shared between processes \*/

+#define CLONE\_FILES 0x00000400 /\* set if open files shared between processes \*/

+#define CLONE\_SIGHAND 0x00000800 /\* set if signal handlers and blocked signals shared \*/

+#define CLONE\_PTRACE 0x00002000 /\* set if we want to let tracing continue on the child too \*/

+#define CLONE\_VFORK 0x00004000 /\* set if the parent wants the child to wake it up on mm\_release \*/

+#define CLONE\_PARENT 0x00008000 /\* set if we want to have the same parent as the cloner \*/

+#define CLONE\_THREAD 0x00010000 /\* Same thread group? \*/

+#define CLONE\_NEWNS 0x00020000 /\* New namespace group? \*/

+#define CLONE\_SYSVSEM 0x00040000 /\* share system V SEM\_UNDO semantics \*/

+#define CLONE\_SETTLS 0x00080000 /\* create a new TLS for the child \*/

+#define CLONE\_PARENT\_SETTID 0x00100000 /\* set the TID in the parent \*/

+#define CLONE\_CHILD\_CLEARTID 0x00200000 /\* clear the TID in the child \*/

+#define CLONE\_DETACHED 0x00400000 /\* Unused, ignored \*/

+#define CLONE\_UNTRACED 0x00800000 /\* set if the tracing process can't force CLONE\_PTRACE on this clone \*/

+#define CLONE\_CHILD\_SETTID 0x01000000 /\* set the TID in the child \*/

+#define CLONE\_STOPPED 0x02000000 /\* Start in stopped state \*/

+#define CLONE\_NEWUTS 0x04000000 /\* New utsname group? \*/

+#define CLONE\_NEWIPC 0x08000000 /\* New ipcs \*/

+#define CLONE\_NEWUSER 0x10000000 /\* New user namespace \*/

+#define CLONE\_NEWPID 0x20000000 /\* New pid namespace \*/

+#define CLONE\_NEWNET 0x40000000 /\* New network namespace \*/

+

+/\*

+ \* Scheduling policies

+ \*/

+#define SCHED\_NORMAL 0

+#define SCHED\_FIFO 1

+#define SCHED\_RR 2

+#define SCHED\_BATCH 3

+/\* SCHED\_ISO: reserved but not implemented yet \*/

+#define SCHED\_IDLE 5

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#define SCHED\_CASIO 6

+#endif

+

+#ifdef \_\_KERNEL\_\_

+

+struct sched\_param {

+ int sched\_priority;

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ unsigned int casio\_id;

+ unsigned long long deadline;

+#endif

+};

+

+#include <asm/param.h> /\* for HZ \*/

+

+#include <linux/capability.h>

+#include <linux/threads.h>

+#include <linux/kernel.h>

+#include <linux/types.h>

+#include <linux/timex.h>

+#include <linux/jiffies.h>

+#include <linux/rbtree.h>

+#include <linux/thread\_info.h>

+#include <linux/cpumask.h>

+#include <linux/errno.h>

+#include <linux/nodemask.h>

+#include <linux/mm\_types.h>

+

+#include <asm/system.h>

+#include <asm/semaphore.h>

+#include <asm/page.h>

+#include <asm/ptrace.h>

+#include <asm/cputime.h>

+

+#include <linux/smp.h>

+#include <linux/sem.h>

+#include <linux/signal.h>

+#include <linux/securebits.h>

+#include <linux/fs\_struct.h>

+#include <linux/compiler.h>

+#include <linux/completion.h>

+#include <linux/pid.h>

+#include <linux/percpu.h>

+#include <linux/topology.h>

+#include <linux/proportions.h>

+#include <linux/seccomp.h>

+#include <linux/rcupdate.h>

+#include <linux/futex.h>

+#include <linux/rtmutex.h>

+

+#include <linux/time.h>

+#include <linux/param.h>

+#include <linux/resource.h>

+#include <linux/timer.h>

+#include <linux/hrtimer.h>

+#include <linux/task\_io\_accounting.h>

+#include <linux/kobject.h>

+

+#include <asm/processor.h>

+

+struct exec\_domain;

+struct futex\_pi\_state;

+struct bio;

+

+/\*

+ \* List of flags we want to share for kernel threads,

+ \* if only because they are not used by them anyway.

+ \*/

+#define CLONE\_KERNEL (CLONE\_FS | CLONE\_FILES | CLONE\_SIGHAND)

+

+/\*

+ \* These are the constant used to fake the fixed-point load-average

+ \* counting. Some notes:

+ \* - 11 bit fractions expand to 22 bits by the multiplies: this gives

+ \* a load-average precision of 10 bits integer + 11 bits fractional

+ \* - if you want to count load-averages more often, you need more

+ \* precision, or rounding will get you. With 2-second counting freq,

+ \* the EXP\_n values would be 1981, 2034 and 2043 if still using only

+ \* 11 bit fractions.

+ \*/

+extern unsigned long avenrun[]; /\* Load averages \*/

+

+#define FSHIFT 11 /\* nr of bits of precision \*/

+#define FIXED\_1 (1<<FSHIFT) /\* 1.0 as fixed-point \*/

+#define LOAD\_FREQ (5\*HZ+1) /\* 5 sec intervals \*/

+#define EXP\_1 1884 /\* 1/exp(5sec/1min) as fixed-point \*/

+#define EXP\_5 2014 /\* 1/exp(5sec/5min) \*/

+#define EXP\_15 2037 /\* 1/exp(5sec/15min) \*/

+

+#define CALC\_LOAD(load,exp,n) \

+ load \*= exp; \

+ load += n\*(FIXED\_1-exp); \

+ load >>= FSHIFT;

+

+extern unsigned long total\_forks;

+extern int nr\_threads;

+DECLARE\_PER\_CPU(unsigned long, process\_counts);

+extern int nr\_processes(void);

+extern unsigned long nr\_running(void);

+extern unsigned long nr\_uninterruptible(void);

+extern unsigned long nr\_active(void);

+extern unsigned long nr\_iowait(void);

+extern unsigned long weighted\_cpuload(const int cpu);

+

+struct seq\_file;

+struct cfs\_rq;

+struct task\_group;

+#ifdef CONFIG\_SCHED\_DEBUG

+extern void proc\_sched\_show\_task(struct task\_struct \*p, struct seq\_file \*m);

+extern void proc\_sched\_set\_task(struct task\_struct \*p);

+extern void

+print\_cfs\_rq(struct seq\_file \*m, int cpu, struct cfs\_rq \*cfs\_rq);

+#else

+static inline void

+proc\_sched\_show\_task(struct task\_struct \*p, struct seq\_file \*m)

+{

+}

+static inline void proc\_sched\_set\_task(struct task\_struct \*p)

+{

+}

+static inline void

+print\_cfs\_rq(struct seq\_file \*m, int cpu, struct cfs\_rq \*cfs\_rq)

+{

+}

+#endif

+

+/\*

+ \* Task state bitmask. NOTE! These bits are also

+ \* encoded in fs/proc/array.c: get\_task\_state().

+ \*

+ \* We have two separate sets of flags: task->state

+ \* is about runnability, while task->exit\_state are

+ \* about the task exiting. Confusing, but this way

+ \* modifying one set can't modify the other one by

+ \* mistake.

+ \*/

+#define TASK\_RUNNING 0

+#define TASK\_INTERRUPTIBLE 1

+#define TASK\_UNINTERRUPTIBLE 2

+#define TASK\_STOPPED 4

+#define TASK\_TRACED 8

+/\* in tsk->exit\_state \*/

+#define EXIT\_ZOMBIE 16

+#define EXIT\_DEAD 32

+/\* in tsk->state again \*/

+#define TASK\_DEAD 64

+

+#define \_\_set\_task\_state(tsk, state\_value) \

+ do { (tsk)->state = (state\_value); } while (0)

+#define set\_task\_state(tsk, state\_value) \

+ set\_mb((tsk)->state, (state\_value))

+

+/\*

+ \* set\_current\_state() includes a barrier so that the write of current->state

+ \* is correctly serialised wrt the caller's subsequent test of whether to

+ \* actually sleep:

+ \*

+ \* set\_current\_state(TASK\_UNINTERRUPTIBLE);

+ \* if (do\_i\_need\_to\_sleep())

+ \* schedule();

+ \*

+ \* If the caller does not need such serialisation then use \_\_set\_current\_state()

+ \*/

+#define \_\_set\_current\_state(state\_value) \

+ do { current->state = (state\_value); } while (0)

+#define set\_current\_state(state\_value) \

+ set\_mb(current->state, (state\_value))

+

+/\* Task command name length \*/

+#define TASK\_COMM\_LEN 16

+

+#include <linux/spinlock.h>

+

+/\*

+ \* This serializes "schedule()" and also protects

+ \* the run-queue from deletions/modifications (but

+ \* \_adding\_ to the beginning of the run-queue has

+ \* a separate lock).

+ \*/

+extern rwlock\_t tasklist\_lock;

+extern spinlock\_t mmlist\_lock;

+

+struct task\_struct;

+

+extern void sched\_init(void);

+extern void sched\_init\_smp(void);

+extern void init\_idle(struct task\_struct \*idle, int cpu);

+extern void init\_idle\_bootup\_task(struct task\_struct \*idle);

+

+extern cpumask\_t nohz\_cpu\_mask;

+#if defined(CONFIG\_SMP) && defined(CONFIG\_NO\_HZ)

+extern int select\_nohz\_load\_balancer(int cpu);

+#else

+static inline int select\_nohz\_load\_balancer(int cpu)

+{

+ return 0;

+}

+#endif

+

+/\*

+ \* Only dump TASK\_\* tasks. (0 for all tasks)

+ \*/

+extern void show\_state\_filter(unsigned long state\_filter);

+

+static inline void show\_state(void)

+{

+ show\_state\_filter(0);

+}

+

+extern void show\_regs(struct pt\_regs \*);

+

+/\*

+ \* TASK is a pointer to the task whose backtrace we want to see (or NULL for current

+ \* task), SP is the stack pointer of the first frame that should be shown in the back

+ \* trace (or NULL if the entire call-chain of the task should be shown).

+ \*/

+extern void show\_stack(struct task\_struct \*task, unsigned long \*sp);

+

+void io\_schedule(void);

+long io\_schedule\_timeout(long timeout);

+

+extern void cpu\_init (void);

+extern void trap\_init(void);

+extern void account\_process\_tick(struct task\_struct \*task, int user);

+extern void update\_process\_times(int user);

+extern void scheduler\_tick(void);

+

+#ifdef CONFIG\_DETECT\_SOFTLOCKUP

+extern void softlockup\_tick(void);

+extern void spawn\_softlockup\_task(void);

+extern void touch\_softlockup\_watchdog(void);

+extern void touch\_all\_softlockup\_watchdogs(void);

+extern int softlockup\_thresh;

+#else

+static inline void softlockup\_tick(void)

+{

+}

+static inline void spawn\_softlockup\_task(void)

+{

+}

+static inline void touch\_softlockup\_watchdog(void)

+{

+}

+static inline void touch\_all\_softlockup\_watchdogs(void)

+{

+}

+#endif

+

+

+/\* Attach to any functions which should be ignored in wchan output. \*/

+#define \_\_sched \_\_attribute\_\_((\_\_section\_\_(".sched.text")))

+

+/\* Linker adds these: start and end of \_\_sched functions \*/

+extern char \_\_sched\_text\_start[], \_\_sched\_text\_end[];

+

+/\* Is this address in the \_\_sched functions? \*/

+extern int in\_sched\_functions(unsigned long addr);

+

+#define MAX\_SCHEDULE\_TIMEOUT LONG\_MAX

+extern signed long FASTCALL(schedule\_timeout(signed long timeout));

+extern signed long schedule\_timeout\_interruptible(signed long timeout);

+extern signed long schedule\_timeout\_uninterruptible(signed long timeout);

+asmlinkage void schedule(void);

+

+struct nsproxy;

+struct user\_namespace;

+

+/\* Maximum number of active map areas.. This is a random (large) number \*/

+#define DEFAULT\_MAX\_MAP\_COUNT 65536

+

+extern int sysctl\_max\_map\_count;

+

+#include <linux/aio.h>

+

+extern unsigned long

+arch\_get\_unmapped\_area(struct file \*, unsigned long, unsigned long,

+ unsigned long, unsigned long);

+extern unsigned long

+arch\_get\_unmapped\_area\_topdown(struct file \*filp, unsigned long addr,

+ unsigned long len, unsigned long pgoff,

+ unsigned long flags);

+extern void arch\_unmap\_area(struct mm\_struct \*, unsigned long);

+extern void arch\_unmap\_area\_topdown(struct mm\_struct \*, unsigned long);

+

+#if NR\_CPUS >= CONFIG\_SPLIT\_PTLOCK\_CPUS

+/\*

+ \* The mm counters are not protected by its page\_table\_lock,

+ \* so must be incremented atomically.

+ \*/

+#define set\_mm\_counter(mm, member, value) atomic\_long\_set(&(mm)->\_##member, value)

+#define get\_mm\_counter(mm, member) ((unsigned long)atomic\_long\_read(&(mm)->\_##member))

+#define add\_mm\_counter(mm, member, value) atomic\_long\_add(value, &(mm)->\_##member)

+#define inc\_mm\_counter(mm, member) atomic\_long\_inc(&(mm)->\_##member)

+#define dec\_mm\_counter(mm, member) atomic\_long\_dec(&(mm)->\_##member)

+

+#else /\* NR\_CPUS < CONFIG\_SPLIT\_PTLOCK\_CPUS \*/

+/\*

+ \* The mm counters are protected by its page\_table\_lock,

+ \* so can be incremented directly.

+ \*/

+#define set\_mm\_counter(mm, member, value) (mm)->\_##member = (value)

+#define get\_mm\_counter(mm, member) ((mm)->\_##member)

+#define add\_mm\_counter(mm, member, value) (mm)->\_##member += (value)

+#define inc\_mm\_counter(mm, member) (mm)->\_##member++

+#define dec\_mm\_counter(mm, member) (mm)->\_##member--

+

+#endif /\* NR\_CPUS < CONFIG\_SPLIT\_PTLOCK\_CPUS \*/

+

+#define get\_mm\_rss(mm) \

+ (get\_mm\_counter(mm, file\_rss) + get\_mm\_counter(mm, anon\_rss))

+#define update\_hiwater\_rss(mm) do { \

+ unsigned long \_rss = get\_mm\_rss(mm); \

+ if ((mm)->hiwater\_rss < \_rss) \

+ (mm)->hiwater\_rss = \_rss; \

+} while (0)

+#define update\_hiwater\_vm(mm) do { \

+ if ((mm)->hiwater\_vm < (mm)->total\_vm) \

+ (mm)->hiwater\_vm = (mm)->total\_vm; \

+} while (0)

+

+extern void set\_dumpable(struct mm\_struct \*mm, int value);

+extern int get\_dumpable(struct mm\_struct \*mm);

+

+/\* mm flags \*/

+/\* dumpable bits \*/

+#define MMF\_DUMPABLE 0 /\* core dump is permitted \*/

+#define MMF\_DUMP\_SECURELY 1 /\* core file is readable only by root \*/

+#define MMF\_DUMPABLE\_BITS 2

+

+/\* coredump filter bits \*/

+#define MMF\_DUMP\_ANON\_PRIVATE 2

+#define MMF\_DUMP\_ANON\_SHARED 3

+#define MMF\_DUMP\_MAPPED\_PRIVATE 4

+#define MMF\_DUMP\_MAPPED\_SHARED 5

+#define MMF\_DUMP\_ELF\_HEADERS 6

+#define MMF\_DUMP\_FILTER\_SHIFT MMF\_DUMPABLE\_BITS

+#define MMF\_DUMP\_FILTER\_BITS 5

+#define MMF\_DUMP\_FILTER\_MASK \

+ (((1 << MMF\_DUMP\_FILTER\_BITS) - 1) << MMF\_DUMP\_FILTER\_SHIFT)

+#define MMF\_DUMP\_FILTER\_DEFAULT \

+ ((1 << MMF\_DUMP\_ANON\_PRIVATE) | (1 << MMF\_DUMP\_ANON\_SHARED))

+

+struct sighand\_struct {

+ atomic\_t count;

+ struct k\_sigaction action[\_NSIG];

+ spinlock\_t siglock;

+ wait\_queue\_head\_t signalfd\_wqh;

+};

+

+struct pacct\_struct {

+ int ac\_flag;

+ long ac\_exitcode;

+ unsigned long ac\_mem;

+ cputime\_t ac\_utime, ac\_stime;

+ unsigned long ac\_minflt, ac\_majflt;

+};

+

+/\*

+ \* NOTE! "signal\_struct" does not have it's own

+ \* locking, because a shared signal\_struct always

+ \* implies a shared sighand\_struct, so locking

+ \* sighand\_struct is always a proper superset of

+ \* the locking of signal\_struct.

+ \*/

+struct signal\_struct {

+ atomic\_t count;

+ atomic\_t live;

+

+ wait\_queue\_head\_t wait\_chldexit; /\* for wait4() \*/

+

+ /\* current thread group signal load-balancing target: \*/

+ struct task\_struct \*curr\_target;

+

+ /\* shared signal handling: \*/

+ struct sigpending shared\_pending;

+

+ /\* thread group exit support \*/

+ int group\_exit\_code;

+ /\* overloaded:

+ \* - notify group\_exit\_task when ->count is equal to notify\_count

+ \* - everyone except group\_exit\_task is stopped during signal delivery

+ \* of fatal signals, group\_exit\_task processes the signal.

+ \*/

+ struct task\_struct \*group\_exit\_task;

+ int notify\_count;

+

+ /\* thread group stop support, overloads group\_exit\_code too \*/

+ int group\_stop\_count;

+ unsigned int flags; /\* see SIGNAL\_\* flags below \*/

+

+ /\* POSIX.1b Interval Timers \*/

+ struct list\_head posix\_timers;

+

+ /\* ITIMER\_REAL timer for the process \*/

+ struct hrtimer real\_timer;

+ struct task\_struct \*tsk;

+ ktime\_t it\_real\_incr;

+

+ /\* ITIMER\_PROF and ITIMER\_VIRTUAL timers for the process \*/

+ cputime\_t it\_prof\_expires, it\_virt\_expires;

+ cputime\_t it\_prof\_incr, it\_virt\_incr;

+

+ /\* job control IDs \*/

+

+ /\*

+ \* pgrp and session fields are deprecated.

+ \* use the task\_session\_Xnr and task\_pgrp\_Xnr routines below

+ \*/

+

+ union {

+ pid\_t pgrp \_\_deprecated;

+ pid\_t \_\_pgrp;

+ };

+

+ struct pid \*tty\_old\_pgrp;

+

+ union {

+ pid\_t session \_\_deprecated;

+ pid\_t \_\_session;

+ };

+

+ /\* boolean value for session group leader \*/

+ int leader;

+

+ struct tty\_struct \*tty; /\* NULL if no tty \*/

+

+ /\*

+ \* Cumulative resource counters for dead threads in the group,

+ \* and for reaped dead child processes forked by this group.

+ \* Live threads maintain their own counters and add to these

+ \* in \_\_exit\_signal, except for the group leader.

+ \*/

+ cputime\_t utime, stime, cutime, cstime;

+ cputime\_t gtime;

+ cputime\_t cgtime;

+ unsigned long nvcsw, nivcsw, cnvcsw, cnivcsw;

+ unsigned long min\_flt, maj\_flt, cmin\_flt, cmaj\_flt;

+ unsigned long inblock, oublock, cinblock, coublock;

+

+ /\*

+ \* Cumulative ns of scheduled CPU time for dead threads in the

+ \* group, not including a zombie group leader. (This only differs

+ \* from jiffies\_to\_ns(utime + stime) if sched\_clock uses something

+ \* other than jiffies.)

+ \*/

+ unsigned long long sum\_sched\_runtime;

+

+ /\*

+ \* We don't bother to synchronize most readers of this at all,

+ \* because there is no reader checking a limit that actually needs

+ \* to get both rlim\_cur and rlim\_max atomically, and either one

+ \* alone is a single word that can safely be read normally.

+ \* getrlimit/setrlimit use task\_lock(current->group\_leader) to

+ \* protect this instead of the siglock, because they really

+ \* have no need to disable irqs.

+ \*/

+ struct rlimit rlim[RLIM\_NLIMITS];

+

+ struct list\_head cpu\_timers[3];

+

+ /\* keep the process-shared keyrings here so that they do the right

+ \* thing in threads created with CLONE\_THREAD \*/

+#ifdef CONFIG\_KEYS

+ struct key \*session\_keyring; /\* keyring inherited over fork \*/

+ struct key \*process\_keyring; /\* keyring private to this process \*/

+#endif

+#ifdef CONFIG\_BSD\_PROCESS\_ACCT

+ struct pacct\_struct pacct; /\* per-process accounting information \*/

+#endif

+#ifdef CONFIG\_TASKSTATS

+ struct taskstats \*stats;

+#endif

+#ifdef CONFIG\_AUDIT

+ unsigned audit\_tty;

+ struct tty\_audit\_buf \*tty\_audit\_buf;

+#endif

+};

+

+/\* Context switch must be unlocked if interrupts are to be enabled \*/

+#ifdef \_\_ARCH\_WANT\_INTERRUPTS\_ON\_CTXSW

+# define \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

+#endif

+

+/\*

+ \* Bits in flags field of signal\_struct.

+ \*/

+#define SIGNAL\_STOP\_STOPPED 0x00000001 /\* job control stop in effect \*/

+#define SIGNAL\_STOP\_DEQUEUED 0x00000002 /\* stop signal dequeued \*/

+#define SIGNAL\_STOP\_CONTINUED 0x00000004 /\* SIGCONT since WCONTINUED reap \*/

+#define SIGNAL\_GROUP\_EXIT 0x00000008 /\* group exit in progress \*/

+

+/\*

+ \* Some day this will be a full-fledged user tracking system..

+ \*/

+struct user\_struct {

+ atomic\_t \_\_count; /\* reference count \*/

+ atomic\_t processes; /\* How many processes does this user have? \*/

+ atomic\_t files; /\* How many open files does this user have? \*/

+ atomic\_t sigpending; /\* How many pending signals does this user have? \*/

+#ifdef CONFIG\_INOTIFY\_USER

+ atomic\_t inotify\_watches; /\* How many inotify watches does this user have? \*/

+ atomic\_t inotify\_devs; /\* How many inotify devs does this user have opened? \*/

+#endif

+#ifdef CONFIG\_POSIX\_MQUEUE

+ /\* protected by mq\_lock \*/

+ unsigned long mq\_bytes; /\* How many bytes can be allocated to mqueue? \*/

+#endif

+ unsigned long locked\_shm; /\* How many pages of mlocked shm ? \*/

+

+#ifdef CONFIG\_KEYS

+ struct key \*uid\_keyring; /\* UID specific keyring \*/

+ struct key \*session\_keyring; /\* UID's default session keyring \*/

+#endif

+

+ /\* Hash table maintenance information \*/

+ struct hlist\_node uidhash\_node;

+ uid\_t uid;

+

+#ifdef CONFIG\_FAIR\_USER\_SCHED

+ struct task\_group \*tg;

+#ifdef CONFIG\_SYSFS

+ struct kset kset;

+ struct subsys\_attribute user\_attr;

+ struct work\_struct work;

+#endif

+#endif

+};

+

+#ifdef CONFIG\_FAIR\_USER\_SCHED

+extern int uids\_kobject\_init(void);

+#else

+static inline int uids\_kobject\_init(void) { return 0; }

+#endif

+

+extern struct user\_struct \*find\_user(uid\_t);

+

+extern struct user\_struct root\_user;

+#define INIT\_USER (&root\_user)

+

+struct backing\_dev\_info;

+struct reclaim\_state;

+

+#if defined(CONFIG\_SCHEDSTATS) || defined(CONFIG\_TASK\_DELAY\_ACCT)

+struct sched\_info {

+ /\* cumulative counters \*/

+ unsigned long pcount; /\* # of times run on this cpu \*/

+ unsigned long long cpu\_time, /\* time spent on the cpu \*/

+ run\_delay; /\* time spent waiting on a runqueue \*/

+

+ /\* timestamps \*/

+ unsigned long long last\_arrival,/\* when we last ran on a cpu \*/

+ last\_queued; /\* when we were last queued to run \*/

+#ifdef CONFIG\_SCHEDSTATS

+ /\* BKL stats \*/

+ unsigned int bkl\_count;

+#endif

+};

+#endif /\* defined(CONFIG\_SCHEDSTATS) || defined(CONFIG\_TASK\_DELAY\_ACCT) \*/

+

+#ifdef CONFIG\_SCHEDSTATS

+extern const struct file\_operations proc\_schedstat\_operations;

+#endif /\* CONFIG\_SCHEDSTATS \*/

+

+#ifdef CONFIG\_TASK\_DELAY\_ACCT

+struct task\_delay\_info {

+ spinlock\_t lock;

+ unsigned int flags; /\* Private per-task flags \*/

+

+ /\* For each stat XXX, add following, aligned appropriately

+ \*

+ \* struct timespec XXX\_start, XXX\_end;

+ \* u64 XXX\_delay;

+ \* u32 XXX\_count;

+ \*

+ \* Atomicity of updates to XXX\_delay, XXX\_count protected by

+ \* single lock above (split into XXX\_lock if contention is an issue).

+ \*/

+

+ /\*

+ \* XXX\_count is incremented on every XXX operation, the delay

+ \* associated with the operation is added to XXX\_delay.

+ \* XXX\_delay contains the accumulated delay time in nanoseconds.

+ \*/

+ struct timespec blkio\_start, blkio\_end; /\* Shared by blkio, swapin \*/

+ u64 blkio\_delay; /\* wait for sync block io completion \*/

+ u64 swapin\_delay; /\* wait for swapin block io completion \*/

+ u32 blkio\_count; /\* total count of the number of sync block \*/

+ /\* io operations performed \*/

+ u32 swapin\_count; /\* total count of the number of swapin block \*/

+ /\* io operations performed \*/

+};

+#endif /\* CONFIG\_TASK\_DELAY\_ACCT \*/

+

+static inline int sched\_info\_on(void)

+{

+#ifdef CONFIG\_SCHEDSTATS

+ return 1;

+#elif defined(CONFIG\_TASK\_DELAY\_ACCT)

+ extern int delayacct\_on;

+ return delayacct\_on;

+#else

+ return 0;

+#endif

+}

+

+enum cpu\_idle\_type {

+ CPU\_IDLE,

+ CPU\_NOT\_IDLE,

+ CPU\_NEWLY\_IDLE,

+ CPU\_MAX\_IDLE\_TYPES

+};

+

+/\*

+ \* sched-domains (multiprocessor balancing) declarations:

+ \*/

+

+/\*

+ \* Increase resolution of nice-level calculations:

+ \*/

+#define SCHED\_LOAD\_SHIFT 10

+#define SCHED\_LOAD\_SCALE (1L << SCHED\_LOAD\_SHIFT)

+

+#define SCHED\_LOAD\_SCALE\_FUZZ SCHED\_LOAD\_SCALE

+

+#ifdef CONFIG\_SMP

+#define SD\_LOAD\_BALANCE 1 /\* Do load balancing on this domain. \*/

+#define SD\_BALANCE\_NEWIDLE 2 /\* Balance when about to become idle \*/

+#define SD\_BALANCE\_EXEC 4 /\* Balance on exec \*/

+#define SD\_BALANCE\_FORK 8 /\* Balance on fork, clone \*/

+#define SD\_WAKE\_IDLE 16 /\* Wake to idle CPU on task wakeup \*/

+#define SD\_WAKE\_AFFINE 32 /\* Wake task to waking CPU \*/

+#define SD\_WAKE\_BALANCE 64 /\* Perform balancing at task wakeup \*/

+#define SD\_SHARE\_CPUPOWER 128 /\* Domain members share cpu power \*/

+#define SD\_POWERSAVINGS\_BALANCE 256 /\* Balance for power savings \*/

+#define SD\_SHARE\_PKG\_RESOURCES 512 /\* Domain members share cpu pkg resources \*/

+#define SD\_SERIALIZE 1024 /\* Only a single load balancing instance \*/

+

+#define BALANCE\_FOR\_MC\_POWER \

+ (sched\_smt\_power\_savings ? SD\_POWERSAVINGS\_BALANCE : 0)

+

+#define BALANCE\_FOR\_PKG\_POWER \

+ ((sched\_mc\_power\_savings || sched\_smt\_power\_savings) ? \

+ SD\_POWERSAVINGS\_BALANCE : 0)

+

+#define test\_sd\_parent(sd, flag) ((sd->parent && \

+ (sd->parent->flags & flag)) ? 1 : 0)

+

+

+struct sched\_group {

+ struct sched\_group \*next; /\* Must be a circular list \*/

+ cpumask\_t cpumask;

+

+ /\*

+ \* CPU power of this group, SCHED\_LOAD\_SCALE being max power for a

+ \* single CPU. This is read only (except for setup, hotplug CPU).

+ \* Note : Never change cpu\_power without recompute its reciprocal

+ \*/

+ unsigned int \_\_cpu\_power;

+ /\*

+ \* reciprocal value of cpu\_power to avoid expensive divides

+ \* (see include/linux/reciprocal\_div.h)

+ \*/

+ u32 reciprocal\_cpu\_power;

+};

+

+struct sched\_domain {

+ /\* These fields must be setup \*/

+ struct sched\_domain \*parent; /\* top domain must be null terminated \*/

+ struct sched\_domain \*child; /\* bottom domain must be null terminated \*/

+ struct sched\_group \*groups; /\* the balancing groups of the domain \*/

+ cpumask\_t span; /\* span of all CPUs in this domain \*/

+ unsigned long min\_interval; /\* Minimum balance interval ms \*/

+ unsigned long max\_interval; /\* Maximum balance interval ms \*/

+ unsigned int busy\_factor; /\* less balancing by factor if busy \*/

+ unsigned int imbalance\_pct; /\* No balance until over watermark \*/

+ unsigned int cache\_nice\_tries; /\* Leave cache hot tasks for # tries \*/

+ unsigned int busy\_idx;

+ unsigned int idle\_idx;

+ unsigned int newidle\_idx;

+ unsigned int wake\_idx;

+ unsigned int forkexec\_idx;

+ int flags; /\* See SD\_\* \*/

+

+ /\* Runtime fields. \*/

+ unsigned long last\_balance; /\* init to jiffies. units in jiffies \*/

+ unsigned int balance\_interval; /\* initialise to 1. units in ms. \*/

+ unsigned int nr\_balance\_failed; /\* initialise to 0 \*/

+

+#ifdef CONFIG\_SCHEDSTATS

+ /\* load\_balance() stats \*/

+ unsigned int lb\_count[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_failed[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_balanced[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_imbalance[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_gained[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_hot\_gained[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_nobusyg[CPU\_MAX\_IDLE\_TYPES];

+ unsigned int lb\_nobusyq[CPU\_MAX\_IDLE\_TYPES];

+

+ /\* Active load balancing \*/

+ unsigned int alb\_count;

+ unsigned int alb\_failed;

+ unsigned int alb\_pushed;

+

+ /\* SD\_BALANCE\_EXEC stats \*/

+ unsigned int sbe\_count;

+ unsigned int sbe\_balanced;

+ unsigned int sbe\_pushed;

+

+ /\* SD\_BALANCE\_FORK stats \*/

+ unsigned int sbf\_count;

+ unsigned int sbf\_balanced;

+ unsigned int sbf\_pushed;

+

+ /\* try\_to\_wake\_up() stats \*/

+ unsigned int ttwu\_wake\_remote;

+ unsigned int ttwu\_move\_affine;

+ unsigned int ttwu\_move\_balance;

+#endif

+};

+

+extern void partition\_sched\_domains(int ndoms\_new, cpumask\_t \*doms\_new);

+

+#endif /\* CONFIG\_SMP \*/

+

+/\*

+ \* A runqueue laden with a single nice 0 task scores a weighted\_cpuload of

+ \* SCHED\_LOAD\_SCALE. This function returns 1 if any cpu is laden with a

+ \* task of nice 0 or enough lower priority tasks to bring up the

+ \* weighted\_cpuload

+ \*/

+static inline int above\_background\_load(void)

+{

+ unsigned long cpu;

+

+ for\_each\_online\_cpu(cpu) {

+ if (weighted\_cpuload(cpu) >= SCHED\_LOAD\_SCALE)

+ return 1;

+ }

+ return 0;

+}

+

+struct io\_context; /\* See blkdev.h \*/

+#define NGROUPS\_SMALL 32

+#define NGROUPS\_PER\_BLOCK ((int)(PAGE\_SIZE / sizeof(gid\_t)))

+struct group\_info {

+ int ngroups;

+ atomic\_t usage;

+ gid\_t small\_block[NGROUPS\_SMALL];

+ int nblocks;

+ gid\_t \*blocks[0];

+};

+

+/\*

+ \* get\_group\_info() must be called with the owning task locked (via task\_lock())

+ \* when task != current. The reason being that the vast majority of callers are

+ \* looking at current->group\_info, which can not be changed except by the

+ \* current task. Changing current->group\_info requires the task lock, too.

+ \*/

+#define get\_group\_info(group\_info) do { \

+ atomic\_inc(&(group\_info)->usage); \

+} while (0)

+

+#define put\_group\_info(group\_info) do { \

+ if (atomic\_dec\_and\_test(&(group\_info)->usage)) \

+ groups\_free(group\_info); \

+} while (0)

+

+extern struct group\_info \*groups\_alloc(int gidsetsize);

+extern void groups\_free(struct group\_info \*group\_info);

+extern int set\_current\_groups(struct group\_info \*group\_info);

+extern int groups\_search(struct group\_info \*group\_info, gid\_t grp);

+/\* access the groups "array" with this macro \*/

+#define GROUP\_AT(gi, i) \

+ ((gi)->blocks[(i)/NGROUPS\_PER\_BLOCK][(i)%NGROUPS\_PER\_BLOCK])

+

+#ifdef ARCH\_HAS\_PREFETCH\_SWITCH\_STACK

+extern void prefetch\_stack(struct task\_struct \*t);

+#else

+static inline void prefetch\_stack(struct task\_struct \*t) { }

+#endif

+

+struct audit\_context; /\* See audit.c \*/

+struct mempolicy;

+struct pipe\_inode\_info;

+struct uts\_namespace;

+

+struct rq;

+struct sched\_domain;

+

+struct sched\_class {

+ const struct sched\_class \*next;

+

+ void (\*enqueue\_task) (struct rq \*rq, struct task\_struct \*p, int wakeup);

+ void (\*dequeue\_task) (struct rq \*rq, struct task\_struct \*p, int sleep);

+ void (\*yield\_task) (struct rq \*rq);

+

+ void (\*check\_preempt\_curr) (struct rq \*rq, struct task\_struct \*p);

+

+ struct task\_struct \* (\*pick\_next\_task) (struct rq \*rq);

+ void (\*put\_prev\_task) (struct rq \*rq, struct task\_struct \*p);

+

+#ifdef CONFIG\_SMP

+ unsigned long (\*load\_balance) (struct rq \*this\_rq, int this\_cpu,

+ struct rq \*busiest, unsigned long max\_load\_move,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*all\_pinned, int \*this\_best\_prio);

+

+ int (\*move\_one\_task) (struct rq \*this\_rq, int this\_cpu,

+ struct rq \*busiest, struct sched\_domain \*sd,

+ enum cpu\_idle\_type idle);

+#endif

+

+ void (\*set\_curr\_task) (struct rq \*rq);

+ void (\*task\_tick) (struct rq \*rq, struct task\_struct \*p);

+ void (\*task\_new) (struct rq \*rq, struct task\_struct \*p);

+};

+

+struct load\_weight {

+ unsigned long weight, inv\_weight;

+};

+

+/\*

+ \* CFS stats for a schedulable entity (task, task-group etc)

+ \*

+ \* Current field usage histogram:

+ \*

+ \* 4 se->block\_start

+ \* 4 se->run\_node

+ \* 4 se->sleep\_start

+ \* 6 se->load.weight

+ \*/

+struct sched\_entity {

+ struct load\_weight load; /\* for load-balancing \*/

+ struct rb\_node run\_node;

+ unsigned int on\_rq;

+

+ u64 exec\_start;

+ u64 sum\_exec\_runtime;

+ u64 vruntime;

+ u64 prev\_sum\_exec\_runtime;

+

+#ifdef CONFIG\_SCHEDSTATS

+ u64 wait\_start;

+ u64 wait\_max;

+

+ u64 sleep\_start;

+ u64 sleep\_max;

+ s64 sum\_sleep\_runtime;

+

+ u64 block\_start;

+ u64 block\_max;

+ u64 exec\_max;

+ u64 slice\_max;

+

+ u64 nr\_migrations;

+ u64 nr\_migrations\_cold;

+ u64 nr\_failed\_migrations\_affine;

+ u64 nr\_failed\_migrations\_running;

+ u64 nr\_failed\_migrations\_hot;

+ u64 nr\_forced\_migrations;

+ u64 nr\_forced2\_migrations;

+

+ u64 nr\_wakeups;

+ u64 nr\_wakeups\_sync;

+ u64 nr\_wakeups\_migrate;

+ u64 nr\_wakeups\_local;

+ u64 nr\_wakeups\_remote;

+ u64 nr\_wakeups\_affine;

+ u64 nr\_wakeups\_affine\_attempts;

+ u64 nr\_wakeups\_passive;

+ u64 nr\_wakeups\_idle;

+#endif

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+ struct sched\_entity \*parent;

+ /\* rq on which this entity is (to be) queued: \*/

+ struct cfs\_rq \*cfs\_rq;

+ /\* rq "owned" by this entity/group: \*/

+ struct cfs\_rq \*my\_q;

+#endif

+};

+

+struct task\_struct {

+ volatile long state; /\* -1 unrunnable, 0 runnable, >0 stopped \*/

+ void \*stack;

+ atomic\_t usage;

+ unsigned int flags; /\* per process flags, defined below \*/

+ unsigned int ptrace;

+

+ int lock\_depth; /\* BKL lock depth \*/

+

+#ifdef CONFIG\_SMP

+#ifdef \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

+ int oncpu;

+#endif

+#endif

+

+ int prio, static\_prio, normal\_prio;

+ struct list\_head run\_list;

+ const struct sched\_class \*sched\_class;

+ struct sched\_entity se;

+

+#ifdef CONFIG\_PREEMPT\_NOTIFIERS

+ /\* list of struct preempt\_notifier: \*/

+ struct hlist\_head preempt\_notifiers;

+#endif

+

+ unsigned short ioprio;

+ /\*

+ \* fpu\_counter contains the number of consecutive context switches

+ \* that the FPU is used. If this is over a threshold, the lazy fpu

+ \* saving becomes unlazy to save the trap. This is an unsigned char

+ \* so that after 256 times the counter wraps and the behavior turns

+ \* lazy again; this to deal with bursty apps that only use FPU for

+ \* a short time

+ \*/

+ unsigned char fpu\_counter;

+ s8 oomkilladj; /\* OOM kill score adjustment (bit shift). \*/

+#ifdef CONFIG\_BLK\_DEV\_IO\_TRACE

+ unsigned int btrace\_seq;

+#endif

+

+ unsigned int policy;

+ cpumask\_t cpus\_allowed;

+ unsigned int time\_slice;

+

+#if defined(CONFIG\_SCHEDSTATS) || defined(CONFIG\_TASK\_DELAY\_ACCT)

+ struct sched\_info sched\_info;

+#endif

+

+ struct list\_head tasks;

+ /\*

+ \* ptrace\_list/ptrace\_children forms the list of my children

+ \* that were stolen by a ptracer.

+ \*/

+ struct list\_head ptrace\_children;

+ struct list\_head ptrace\_list;

+

+ struct mm\_struct \*mm, \*active\_mm;

+

+/\* task state \*/

+ struct linux\_binfmt \*binfmt;

+ int exit\_state;

+ int exit\_code, exit\_signal;

+ int pdeath\_signal; /\* The signal sent when the parent dies \*/

+ /\* ??? \*/

+ unsigned int personality;

+ unsigned did\_exec:1;

+ pid\_t pid;

+ pid\_t tgid;

+

+#ifdef CONFIG\_CC\_STACKPROTECTOR

+ /\* Canary value for the -fstack-protector gcc feature \*/

+ unsigned long stack\_canary;

+#endif

+ /\*

+ \* pointers to (original) parent process, youngest child, younger sibling,

+ \* older sibling, respectively. (p->father can be replaced with

+ \* p->parent->pid)

+ \*/

+ struct task\_struct \*real\_parent; /\* real parent process (when being debugged) \*/

+ struct task\_struct \*parent; /\* parent process \*/

+ /\*

+ \* children/sibling forms the list of my children plus the

+ \* tasks I'm ptracing.

+ \*/

+ struct list\_head children; /\* list of my children \*/

+ struct list\_head sibling; /\* linkage in my parent's children list \*/

+ struct task\_struct \*group\_leader; /\* threadgroup leader \*/

+

+ /\* PID/PID hash table linkage. \*/

+ struct pid\_link pids[PIDTYPE\_MAX];

+ struct list\_head thread\_group;

+

+ struct completion \*vfork\_done; /\* for vfork() \*/

+ int \_\_user \*set\_child\_tid; /\* CLONE\_CHILD\_SETTID \*/

+ int \_\_user \*clear\_child\_tid; /\* CLONE\_CHILD\_CLEARTID \*/

+

+ unsigned int rt\_priority;

+ cputime\_t utime, stime, utimescaled, stimescaled;

+ cputime\_t gtime;

+ cputime\_t prev\_utime, prev\_stime;

+ unsigned long nvcsw, nivcsw; /\* context switch counts \*/

+ struct timespec start\_time; /\* monotonic time \*/

+ struct timespec real\_start\_time; /\* boot based time \*/

+/\* mm fault and swap info: this can arguably be seen as either mm-specific or thread-specific \*/

+ unsigned long min\_flt, maj\_flt;

+

+ cputime\_t it\_prof\_expires, it\_virt\_expires;

+ unsigned long long it\_sched\_expires;

+ struct list\_head cpu\_timers[3];

+

+/\* process credentials \*/

+ uid\_t uid,euid,suid,fsuid;

+ gid\_t gid,egid,sgid,fsgid;

+ struct group\_info \*group\_info;

+ kernel\_cap\_t cap\_effective, cap\_inheritable, cap\_permitted;

+ unsigned keep\_capabilities:1;

+ struct user\_struct \*user;

+#ifdef CONFIG\_KEYS

+ struct key \*request\_key\_auth; /\* assumed request\_key authority \*/

+ struct key \*thread\_keyring; /\* keyring private to this thread \*/

+ unsigned char jit\_keyring; /\* default keyring to attach requested keys to \*/

+#endif

+ char comm[TASK\_COMM\_LEN]; /\* executable name excluding path

+ - access with [gs]et\_task\_comm (which lock

+ it with task\_lock())

+ - initialized normally by flush\_old\_exec \*/

+/\* file system info \*/

+ int link\_count, total\_link\_count;

+#ifdef CONFIG\_SYSVIPC

+/\* ipc stuff \*/

+ struct sysv\_sem sysvsem;

+#endif

+/\* CPU-specific state of this task \*/

+ struct thread\_struct thread;

+/\* filesystem information \*/

+ struct fs\_struct \*fs;

+/\* open file information \*/

+ struct files\_struct \*files;

+/\* namespaces \*/

+ struct nsproxy \*nsproxy;

+/\* signal handlers \*/

+ struct signal\_struct \*signal;

+ struct sighand\_struct \*sighand;

+

+ sigset\_t blocked, real\_blocked;

+ sigset\_t saved\_sigmask; /\* To be restored with TIF\_RESTORE\_SIGMASK \*/

+ struct sigpending pending;

+

+ unsigned long sas\_ss\_sp;

+ size\_t sas\_ss\_size;

+ int (\*notifier)(void \*priv);

+ void \*notifier\_data;

+ sigset\_t \*notifier\_mask;

+#ifdef CONFIG\_SECURITY

+ void \*security;

+#endif

+ struct audit\_context \*audit\_context;

+ seccomp\_t seccomp;

+

+/\* Thread group tracking \*/

+ u32 parent\_exec\_id;

+ u32 self\_exec\_id;

+/\* Protection of (de-)allocation: mm, files, fs, tty, keyrings \*/

+ spinlock\_t alloc\_lock;

+

+ /\* Protection of the PI data structures: \*/

+ spinlock\_t pi\_lock;

+

+#ifdef CONFIG\_RT\_MUTEXES

+ /\* PI waiters blocked on a rt\_mutex held by this task \*/

+ struct plist\_head pi\_waiters;

+ /\* Deadlock detection and priority inheritance handling \*/

+ struct rt\_mutex\_waiter \*pi\_blocked\_on;

+#endif

+

+#ifdef CONFIG\_DEBUG\_MUTEXES

+ /\* mutex deadlock detection \*/

+ struct mutex\_waiter \*blocked\_on;

+#endif

+#ifdef CONFIG\_TRACE\_IRQFLAGS

+ unsigned int irq\_events;

+ int hardirqs\_enabled;

+ unsigned long hardirq\_enable\_ip;

+ unsigned int hardirq\_enable\_event;

+ unsigned long hardirq\_disable\_ip;

+ unsigned int hardirq\_disable\_event;

+ int softirqs\_enabled;

+ unsigned long softirq\_disable\_ip;

+ unsigned int softirq\_disable\_event;

+ unsigned long softirq\_enable\_ip;

+ unsigned int softirq\_enable\_event;

+ int hardirq\_context;

+ int softirq\_context;

+#endif

+#ifdef CONFIG\_LOCKDEP

+# define MAX\_LOCK\_DEPTH 30UL

+ u64 curr\_chain\_key;

+ int lockdep\_depth;

+ struct held\_lock held\_locks[MAX\_LOCK\_DEPTH];

+ unsigned int lockdep\_recursion;

+#endif

+

+/\* journalling filesystem info \*/

+ void \*journal\_info;

+

+/\* stacked block device info \*/

+ struct bio \*bio\_list, \*\*bio\_tail;

+

+/\* VM state \*/

+ struct reclaim\_state \*reclaim\_state;

+

+ struct backing\_dev\_info \*backing\_dev\_info;

+

+ struct io\_context \*io\_context;

+

+ unsigned long ptrace\_message;

+ siginfo\_t \*last\_siginfo; /\* For ptrace use. \*/

+#ifdef CONFIG\_TASK\_XACCT

+/\* i/o counters(bytes read/written, #syscalls \*/

+ u64 rchar, wchar, syscr, syscw;

+#endif

+ struct task\_io\_accounting ioac;

+#if defined(CONFIG\_TASK\_XACCT)

+ u64 acct\_rss\_mem1; /\* accumulated rss usage \*/

+ u64 acct\_vm\_mem1; /\* accumulated virtual memory usage \*/

+ cputime\_t acct\_stimexpd;/\* stime since last update \*/

+#endif

+#ifdef CONFIG\_NUMA

+ struct mempolicy \*mempolicy;

+ short il\_next;

+#endif

+#ifdef CONFIG\_CPUSETS

+ nodemask\_t mems\_allowed;

+ int cpuset\_mems\_generation;

+ int cpuset\_mem\_spread\_rotor;

+#endif

+#ifdef CONFIG\_CGROUPS

+ /\* Control Group info protected by css\_set\_lock \*/

+ struct css\_set \*cgroups;

+ /\* cg\_list protected by css\_set\_lock and tsk->alloc\_lock \*/

+ struct list\_head cg\_list;

+#endif

+#ifdef CONFIG\_FUTEX

+ struct robust\_list\_head \_\_user \*robust\_list;

+#ifdef CONFIG\_COMPAT

+ struct compat\_robust\_list\_head \_\_user \*compat\_robust\_list;

+#endif

+ struct list\_head pi\_state\_list;

+ struct futex\_pi\_state \*pi\_state\_cache;

+#endif

+ atomic\_t fs\_excl; /\* holding fs exclusive resources \*/

+ struct rcu\_head rcu;

+

+ /\*

+ \* cache last used pipe for splice

+ \*/

+ struct pipe\_inode\_info \*splice\_pipe;

+#ifdef CONFIG\_TASK\_DELAY\_ACCT

+ struct task\_delay\_info \*delays;

+#endif

+#ifdef CONFIG\_FAULT\_INJECTION

+ int make\_it\_fail;

+#endif

+ struct prop\_local\_single dirties;

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ unsigned int casio\_id;

+ unsigned long long deadline;

+#endif

+};

+

+/\*

+ \* Priority of a process goes from 0..MAX\_PRIO-1, valid RT

+ \* priority is 0..MAX\_RT\_PRIO-1, and SCHED\_NORMAL/SCHED\_BATCH

+ \* tasks are in the range MAX\_RT\_PRIO..MAX\_PRIO-1. Priority

+ \* values are inverted: lower p->prio value means higher priority.

+ \*

+ \* The MAX\_USER\_RT\_PRIO value allows the actual maximum

+ \* RT priority to be separate from the value exported to

+ \* user-space. This allows kernel threads to set their

+ \* priority to a value higher than any user task. Note:

+ \* MAX\_RT\_PRIO must not be smaller than MAX\_USER\_RT\_PRIO.

+ \*/

+

+#define MAX\_USER\_RT\_PRIO 100

+#define MAX\_RT\_PRIO MAX\_USER\_RT\_PRIO

+

+#define MAX\_PRIO (MAX\_RT\_PRIO + 40)

+#define DEFAULT\_PRIO (MAX\_RT\_PRIO + 20)

+

+static inline int rt\_prio(int prio)

+{

+ if (unlikely(prio < MAX\_RT\_PRIO))

+ return 1;

+ return 0;

+}

+

+static inline int rt\_task(struct task\_struct \*p)

+{

+ return rt\_prio(p->prio);

+}

+

+static inline void set\_task\_session(struct task\_struct \*tsk, pid\_t session)

+{

+ tsk->signal->\_\_session = session;

+}

+

+static inline void set\_task\_pgrp(struct task\_struct \*tsk, pid\_t pgrp)

+{

+ tsk->signal->\_\_pgrp = pgrp;

+}

+

+static inline struct pid \*task\_pid(struct task\_struct \*task)

+{

+ return task->pids[PIDTYPE\_PID].pid;

+}

+

+static inline struct pid \*task\_tgid(struct task\_struct \*task)

+{

+ return task->group\_leader->pids[PIDTYPE\_PID].pid;

+}

+

+static inline struct pid \*task\_pgrp(struct task\_struct \*task)

+{

+ return task->group\_leader->pids[PIDTYPE\_PGID].pid;

+}

+

+static inline struct pid \*task\_session(struct task\_struct \*task)

+{

+ return task->group\_leader->pids[PIDTYPE\_SID].pid;

+}

+

+struct pid\_namespace;

+

+/\*

+ \* the helpers to get the task's different pids as they are seen

+ \* from various namespaces

+ \*

+ \* task\_xid\_nr() : global id, i.e. the id seen from the init namespace;

+ \* task\_xid\_vnr() : virtual id, i.e. the id seen from the namespace the task

+ \* belongs to. this only makes sence when called in the

+ \* context of the task that belongs to the same namespace;

+ \* task\_xid\_nr\_ns() : id seen from the ns specified;

+ \*

+ \* set\_task\_vxid() : assigns a virtual id to a task;

+ \*

+ \* see also pid\_nr() etc in include/linux/pid.h

+ \*/

+

+static inline pid\_t task\_pid\_nr(struct task\_struct \*tsk)

+{

+ return tsk->pid;

+}

+

+pid\_t task\_pid\_nr\_ns(struct task\_struct \*tsk, struct pid\_namespace \*ns);

+

+static inline pid\_t task\_pid\_vnr(struct task\_struct \*tsk)

+{

+ return pid\_vnr(task\_pid(tsk));

+}

+

+

+static inline pid\_t task\_tgid\_nr(struct task\_struct \*tsk)

+{

+ return tsk->tgid;

+}

+

+pid\_t task\_tgid\_nr\_ns(struct task\_struct \*tsk, struct pid\_namespace \*ns);

+

+static inline pid\_t task\_tgid\_vnr(struct task\_struct \*tsk)

+{

+ return pid\_vnr(task\_tgid(tsk));

+}

+

+

+static inline pid\_t task\_pgrp\_nr(struct task\_struct \*tsk)

+{

+ return tsk->signal->\_\_pgrp;

+}

+

+pid\_t task\_pgrp\_nr\_ns(struct task\_struct \*tsk, struct pid\_namespace \*ns);

+

+static inline pid\_t task\_pgrp\_vnr(struct task\_struct \*tsk)

+{

+ return pid\_vnr(task\_pgrp(tsk));

+}

+

+

+static inline pid\_t task\_session\_nr(struct task\_struct \*tsk)

+{

+ return tsk->signal->\_\_session;

+}

+

+pid\_t task\_session\_nr\_ns(struct task\_struct \*tsk, struct pid\_namespace \*ns);

+

+static inline pid\_t task\_session\_vnr(struct task\_struct \*tsk)

+{

+ return pid\_vnr(task\_session(tsk));

+}

+

+

+/\*\*

+ \* pid\_alive - check that a task structure is not stale

+ \* @p: Task structure to be checked.

+ \*

+ \* Test if a process is not yet dead (at most zombie state)

+ \* If pid\_alive fails, then pointers within the task structure

+ \* can be stale and must not be dereferenced.

+ \*/

+static inline int pid\_alive(struct task\_struct \*p)

+{

+ return p->pids[PIDTYPE\_PID].pid != NULL;

+}

+

+/\*\*

+ \* is\_global\_init - check if a task structure is init

+ \* @tsk: Task structure to be checked.

+ \*

+ \* Check if a task structure is the first user space task the kernel created.

+ \*/

+static inline int is\_global\_init(struct task\_struct \*tsk)

+{

+ return tsk->pid == 1;

+}

+

+/\*

+ \* is\_container\_init:

+ \* check whether in the task is init in its own pid namespace.

+ \*/

+extern int is\_container\_init(struct task\_struct \*tsk);

+

+extern struct pid \*cad\_pid;

+

+extern void free\_task(struct task\_struct \*tsk);

+#define get\_task\_struct(tsk) do { atomic\_inc(&(tsk)->usage); } while(0)

+

+extern void \_\_put\_task\_struct(struct task\_struct \*t);

+

+static inline void put\_task\_struct(struct task\_struct \*t)

+{

+ if (atomic\_dec\_and\_test(&t->usage))

+ \_\_put\_task\_struct(t);

+}

+

+/\*

+ \* Per process flags

+ \*/

+#define PF\_ALIGNWARN 0x00000001 /\* Print alignment warning msgs \*/

+ /\* Not implemented yet, only for 486\*/

+#define PF\_STARTING 0x00000002 /\* being created \*/

+#define PF\_EXITING 0x00000004 /\* getting shut down \*/

+#define PF\_EXITPIDONE 0x00000008 /\* pi exit done on shut down \*/

+#define PF\_VCPU 0x00000010 /\* I'm a virtual CPU \*/

+#define PF\_FORKNOEXEC 0x00000040 /\* forked but didn't exec \*/

+#define PF\_SUPERPRIV 0x00000100 /\* used super-user privileges \*/

+#define PF\_DUMPCORE 0x00000200 /\* dumped core \*/

+#define PF\_SIGNALED 0x00000400 /\* killed by a signal \*/

+#define PF\_MEMALLOC 0x00000800 /\* Allocating memory \*/

+#define PF\_FLUSHER 0x00001000 /\* responsible for disk writeback \*/

+#define PF\_USED\_MATH 0x00002000 /\* if unset the fpu must be initialized before use \*/

+#define PF\_NOFREEZE 0x00008000 /\* this thread should not be frozen \*/

+#define PF\_FROZEN 0x00010000 /\* frozen for system suspend \*/

+#define PF\_FSTRANS 0x00020000 /\* inside a filesystem transaction \*/

+#define PF\_KSWAPD 0x00040000 /\* I am kswapd \*/

+#define PF\_SWAPOFF 0x00080000 /\* I am in swapoff \*/

+#define PF\_LESS\_THROTTLE 0x00100000 /\* Throttle me less: I clean memory \*/

+#define PF\_BORROWED\_MM 0x00200000 /\* I am a kthread doing use\_mm \*/

+#define PF\_RANDOMIZE 0x00400000 /\* randomize virtual address space \*/

+#define PF\_SWAPWRITE 0x00800000 /\* Allowed to write to swap \*/

+#define PF\_SPREAD\_PAGE 0x01000000 /\* Spread page cache over cpuset \*/

+#define PF\_SPREAD\_SLAB 0x02000000 /\* Spread some slab caches over cpuset \*/

+#define PF\_MEMPOLICY 0x10000000 /\* Non-default NUMA mempolicy \*/

+#define PF\_MUTEX\_TESTER 0x20000000 /\* Thread belongs to the rt mutex tester \*/

+#define PF\_FREEZER\_SKIP 0x40000000 /\* Freezer should not count it as freezeable \*/

+

+/\*

+ \* Only the \_current\_ task can read/write to tsk->flags, but other

+ \* tasks can access tsk->flags in readonly mode for example

+ \* with tsk\_used\_math (like during threaded core dumping).

+ \* There is however an exception to this rule during ptrace

+ \* or during fork: the ptracer task is allowed to write to the

+ \* child->flags of its traced child (same goes for fork, the parent

+ \* can write to the child->flags), because we're guaranteed the

+ \* child is not running and in turn not changing child->flags

+ \* at the same time the parent does it.

+ \*/

+#define clear\_stopped\_child\_used\_math(child) do { (child)->flags &= ~PF\_USED\_MATH; } while (0)

+#define set\_stopped\_child\_used\_math(child) do { (child)->flags |= PF\_USED\_MATH; } while (0)

+#define clear\_used\_math() clear\_stopped\_child\_used\_math(current)

+#define set\_used\_math() set\_stopped\_child\_used\_math(current)

+#define conditional\_stopped\_child\_used\_math(condition, child) \

+ do { (child)->flags &= ~PF\_USED\_MATH, (child)->flags |= (condition) ? PF\_USED\_MATH : 0; } while (0)

+#define conditional\_used\_math(condition) \

+ conditional\_stopped\_child\_used\_math(condition, current)

+#define copy\_to\_stopped\_child\_used\_math(child) \

+ do { (child)->flags &= ~PF\_USED\_MATH, (child)->flags |= current->flags & PF\_USED\_MATH; } while (0)

+/\* NOTE: this will return 0 or PF\_USED\_MATH, it will never return 1 \*/

+#define tsk\_used\_math(p) ((p)->flags & PF\_USED\_MATH)

+#define used\_math() tsk\_used\_math(current)

+

+#ifdef CONFIG\_SMP

+extern int set\_cpus\_allowed(struct task\_struct \*p, cpumask\_t new\_mask);

+#else

+static inline int set\_cpus\_allowed(struct task\_struct \*p, cpumask\_t new\_mask)

+{

+ if (!cpu\_isset(0, new\_mask))

+ return -EINVAL;

+ return 0;

+}

+#endif

+

+extern unsigned long long sched\_clock(void);

+

+/\*

+ \* For kernel-internal use: high-speed (but slightly incorrect) per-cpu

+ \* clock constructed from sched\_clock():

+ \*/

+extern unsigned long long cpu\_clock(int cpu);

+

+extern unsigned long long

+task\_sched\_runtime(struct task\_struct \*task);

+

+/\* sched\_exec is called by processes performing an exec \*/

+#ifdef CONFIG\_SMP

+extern void sched\_exec(void);

+#else

+#define sched\_exec() {}

+#endif

+

+extern void sched\_clock\_idle\_sleep\_event(void);

+extern void sched\_clock\_idle\_wakeup\_event(u64 delta\_ns);

+

+#ifdef CONFIG\_HOTPLUG\_CPU

+extern void idle\_task\_exit(void);

+#else

+static inline void idle\_task\_exit(void) {}

+#endif

+

+extern void sched\_idle\_next(void);

+

+#ifdef CONFIG\_SCHED\_DEBUG

+extern unsigned int sysctl\_sched\_latency;

+extern unsigned int sysctl\_sched\_min\_granularity;

+extern unsigned int sysctl\_sched\_wakeup\_granularity;

+extern unsigned int sysctl\_sched\_batch\_wakeup\_granularity;

+extern unsigned int sysctl\_sched\_child\_runs\_first;

+extern unsigned int sysctl\_sched\_features;

+extern unsigned int sysctl\_sched\_migration\_cost;

+extern unsigned int sysctl\_sched\_nr\_migrate;

+

+int sched\_nr\_latency\_handler(struct ctl\_table \*table, int write,

+ struct file \*file, void \_\_user \*buffer, size\_t \*length,

+ loff\_t \*ppos);

+#endif

+

+extern unsigned int sysctl\_sched\_compat\_yield;

+

+#ifdef CONFIG\_RT\_MUTEXES

+extern int rt\_mutex\_getprio(struct task\_struct \*p);

+extern void rt\_mutex\_setprio(struct task\_struct \*p, int prio);

+extern void rt\_mutex\_adjust\_pi(struct task\_struct \*p);

+#else

+static inline int rt\_mutex\_getprio(struct task\_struct \*p)

+{

+ return p->normal\_prio;

+}

+# define rt\_mutex\_adjust\_pi(p) do { } while (0)

+#endif

+

+extern void set\_user\_nice(struct task\_struct \*p, long nice);

+extern int task\_prio(const struct task\_struct \*p);

+extern int task\_nice(const struct task\_struct \*p);

+extern int can\_nice(const struct task\_struct \*p, const int nice);

+extern int task\_curr(const struct task\_struct \*p);

+extern int idle\_cpu(int cpu);

+extern int sched\_setscheduler(struct task\_struct \*, int, struct sched\_param \*);

+extern struct task\_struct \*idle\_task(int cpu);

+extern struct task\_struct \*curr\_task(int cpu);

+extern void set\_curr\_task(int cpu, struct task\_struct \*p);

+

+void yield(void);

+

+/\*

+ \* The default (Linux) execution domain.

+ \*/

+extern struct exec\_domain default\_exec\_domain;

+

+union thread\_union {

+ struct thread\_info thread\_info;

+ unsigned long stack[THREAD\_SIZE/sizeof(long)];

+};

+

+#ifndef \_\_HAVE\_ARCH\_KSTACK\_END

+static inline int kstack\_end(void \*addr)

+{

+ /\* Reliable end of stack detection:

+ \* Some APM bios versions misalign the stack

+ \*/

+ return !(((unsigned long)addr+sizeof(void\*)-1) & (THREAD\_SIZE-sizeof(void\*)));

+}

+#endif

+

+extern union thread\_union init\_thread\_union;

+extern struct task\_struct init\_task;

+

+extern struct mm\_struct init\_mm;

+

+extern struct pid\_namespace init\_pid\_ns;

+

+/\*

+ \* find a task by one of its numerical ids

+ \*

+ \* find\_task\_by\_pid\_type\_ns():

+ \* it is the most generic call - it finds a task by all id,

+ \* type and namespace specified

+ \* find\_task\_by\_pid\_ns():

+ \* finds a task by its pid in the specified namespace

+ \* find\_task\_by\_vpid():

+ \* finds a task by its virtual pid

+ \* find\_task\_by\_pid():

+ \* finds a task by its global pid

+ \*

+ \* see also find\_pid() etc in include/linux/pid.h

+ \*/

+

+extern struct task\_struct \*find\_task\_by\_pid\_type\_ns(int type, int pid,

+ struct pid\_namespace \*ns);

+

+extern struct task\_struct \*find\_task\_by\_pid(pid\_t nr);

+extern struct task\_struct \*find\_task\_by\_vpid(pid\_t nr);

+extern struct task\_struct \*find\_task\_by\_pid\_ns(pid\_t nr,

+ struct pid\_namespace \*ns);

+

+extern void \_\_set\_special\_pids(pid\_t session, pid\_t pgrp);

+

+/\* per-UID process charging. \*/

+extern struct user\_struct \* alloc\_uid(struct user\_namespace \*, uid\_t);

+static inline struct user\_struct \*get\_uid(struct user\_struct \*u)

+{

+ atomic\_inc(&u->\_\_count);

+ return u;

+}

+extern void free\_uid(struct user\_struct \*);

+extern void switch\_uid(struct user\_struct \*);

+extern void release\_uids(struct user\_namespace \*ns);

+

+#include <asm/current.h>

+

+extern void do\_timer(unsigned long ticks);

+

+extern int FASTCALL(wake\_up\_state(struct task\_struct \* tsk, unsigned int state));

+extern int FASTCALL(wake\_up\_process(struct task\_struct \* tsk));

+extern void FASTCALL(wake\_up\_new\_task(struct task\_struct \* tsk,

+ unsigned long clone\_flags));

+#ifdef CONFIG\_SMP

+ extern void kick\_process(struct task\_struct \*tsk);

+#else

+ static inline void kick\_process(struct task\_struct \*tsk) { }

+#endif

+extern void sched\_fork(struct task\_struct \*p, int clone\_flags);

+extern void sched\_dead(struct task\_struct \*p);

+

+extern int in\_group\_p(gid\_t);

+extern int in\_egroup\_p(gid\_t);

+

+extern void proc\_caches\_init(void);

+extern void flush\_signals(struct task\_struct \*);

+extern void ignore\_signals(struct task\_struct \*);

+extern void flush\_signal\_handlers(struct task\_struct \*, int force\_default);

+extern int dequeue\_signal(struct task\_struct \*tsk, sigset\_t \*mask, siginfo\_t \*info);

+

+static inline int dequeue\_signal\_lock(struct task\_struct \*tsk, sigset\_t \*mask, siginfo\_t \*info)

+{

+ unsigned long flags;

+ int ret;

+

+ spin\_lock\_irqsave(&tsk->sighand->siglock, flags);

+ ret = dequeue\_signal(tsk, mask, info);

+ spin\_unlock\_irqrestore(&tsk->sighand->siglock, flags);

+

+ return ret;

+}

+

+extern void block\_all\_signals(int (\*notifier)(void \*priv), void \*priv,

+ sigset\_t \*mask);

+extern void unblock\_all\_signals(void);

+extern void release\_task(struct task\_struct \* p);

+extern int send\_sig\_info(int, struct siginfo \*, struct task\_struct \*);

+extern int send\_group\_sig\_info(int, struct siginfo \*, struct task\_struct \*);

+extern int force\_sigsegv(int, struct task\_struct \*);

+extern int force\_sig\_info(int, struct siginfo \*, struct task\_struct \*);

+extern int \_\_kill\_pgrp\_info(int sig, struct siginfo \*info, struct pid \*pgrp);

+extern int kill\_pgrp\_info(int sig, struct siginfo \*info, struct pid \*pgrp);

+extern int kill\_pid\_info(int sig, struct siginfo \*info, struct pid \*pid);

+extern int kill\_pid\_info\_as\_uid(int, struct siginfo \*, struct pid \*, uid\_t, uid\_t, u32);

+extern int kill\_pgrp(struct pid \*pid, int sig, int priv);

+extern int kill\_pid(struct pid \*pid, int sig, int priv);

+extern int kill\_proc\_info(int, struct siginfo \*, pid\_t);

+extern void do\_notify\_parent(struct task\_struct \*, int);

+extern void force\_sig(int, struct task\_struct \*);

+extern void force\_sig\_specific(int, struct task\_struct \*);

+extern int send\_sig(int, struct task\_struct \*, int);

+extern void zap\_other\_threads(struct task\_struct \*p);

+extern int kill\_proc(pid\_t, int, int);

+extern struct sigqueue \*sigqueue\_alloc(void);

+extern void sigqueue\_free(struct sigqueue \*);

+extern int send\_sigqueue(int, struct sigqueue \*, struct task\_struct \*);

+extern int send\_group\_sigqueue(int, struct sigqueue \*, struct task\_struct \*);

+extern int do\_sigaction(int, struct k\_sigaction \*, struct k\_sigaction \*);

+extern int do\_sigaltstack(const stack\_t \_\_user \*, stack\_t \_\_user \*, unsigned long);

+

+static inline int kill\_cad\_pid(int sig, int priv)

+{

+ return kill\_pid(cad\_pid, sig, priv);

+}

+

+/\* These can be the second arg to send\_sig\_info/send\_group\_sig\_info. \*/

+#define SEND\_SIG\_NOINFO ((struct siginfo \*) 0)

+#define SEND\_SIG\_PRIV ((struct siginfo \*) 1)

+#define SEND\_SIG\_FORCED ((struct siginfo \*) 2)

+

+static inline int is\_si\_special(const struct siginfo \*info)

+{

+ return info <= SEND\_SIG\_FORCED;

+}

+

+/\* True if we are on the alternate signal stack. \*/

+

+static inline int on\_sig\_stack(unsigned long sp)

+{

+ return (sp - current->sas\_ss\_sp < current->sas\_ss\_size);

+}

+

+static inline int sas\_ss\_flags(unsigned long sp)

+{

+ return (current->sas\_ss\_size == 0 ? SS\_DISABLE

+ : on\_sig\_stack(sp) ? SS\_ONSTACK : 0);

+}

+

+/\*

+ \* Routines for handling mm\_structs

+ \*/

+extern struct mm\_struct \* mm\_alloc(void);

+

+/\* mmdrop drops the mm and the page tables \*/

+extern void FASTCALL(\_\_mmdrop(struct mm\_struct \*));

+static inline void mmdrop(struct mm\_struct \* mm)

+{

+ if (unlikely(atomic\_dec\_and\_test(&mm->mm\_count)))

+ \_\_mmdrop(mm);

+}

+

+/\* mmput gets rid of the mappings and all user-space \*/

+extern void mmput(struct mm\_struct \*);

+/\* Grab a reference to a task's mm, if it is not already going away \*/

+extern struct mm\_struct \*get\_task\_mm(struct task\_struct \*task);

+/\* Remove the current tasks stale references to the old mm\_struct \*/

+extern void mm\_release(struct task\_struct \*, struct mm\_struct \*);

+

+extern int copy\_thread(int, unsigned long, unsigned long, unsigned long, struct task\_struct \*, struct pt\_regs \*);

+extern void flush\_thread(void);

+extern void exit\_thread(void);

+

+extern void exit\_files(struct task\_struct \*);

+extern void \_\_cleanup\_signal(struct signal\_struct \*);

+extern void \_\_cleanup\_sighand(struct sighand\_struct \*);

+extern void exit\_itimers(struct signal\_struct \*);

+

+extern NORET\_TYPE void do\_group\_exit(int);

+

+extern void daemonize(const char \*, ...);

+extern int allow\_signal(int);

+extern int disallow\_signal(int);

+

+extern int do\_execve(char \*, char \_\_user \* \_\_user \*, char \_\_user \* \_\_user \*, struct pt\_regs \*);

+extern long do\_fork(unsigned long, unsigned long, struct pt\_regs \*, unsigned long, int \_\_user \*, int \_\_user \*);

+struct task\_struct \*fork\_idle(int);

+

+extern void set\_task\_comm(struct task\_struct \*tsk, char \*from);

+extern void get\_task\_comm(char \*to, struct task\_struct \*tsk);

+

+#ifdef CONFIG\_SMP

+extern void wait\_task\_inactive(struct task\_struct \* p);

+#else

+#define wait\_task\_inactive(p) do { } while (0)

+#endif

+

+#define remove\_parent(p) list\_del\_init(&(p)->sibling)

+#define add\_parent(p) list\_add\_tail(&(p)->sibling,&(p)->parent->children)

+

+#define next\_task(p) list\_entry(rcu\_dereference((p)->tasks.next), struct task\_struct, tasks)

+

+#define for\_each\_process(p) \

+ for (p = &init\_task ; (p = next\_task(p)) != &init\_task ; )

+

+/\*

+ \* Careful: do\_each\_thread/while\_each\_thread is a double loop so

+ \* 'break' will not work as expected - use goto instead.

+ \*/

+#define do\_each\_thread(g, t) \

+ for (g = t = &init\_task ; (g = t = next\_task(g)) != &init\_task ; ) do

+

+#define while\_each\_thread(g, t) \

+ while ((t = next\_thread(t)) != g)

+

+/\* de\_thread depends on thread\_group\_leader not being a pid based check \*/

+#define thread\_group\_leader(p) (p == p->group\_leader)

+

+/\* Do to the insanities of de\_thread it is possible for a process

+ \* to have the pid of the thread group leader without actually being

+ \* the thread group leader. For iteration through the pids in proc

+ \* all we care about is that we have a task with the appropriate

+ \* pid, we don't actually care if we have the right task.

+ \*/

+static inline int has\_group\_leader\_pid(struct task\_struct \*p)

+{

+ return p->pid == p->tgid;

+}

+

+static inline

+int same\_thread\_group(struct task\_struct \*p1, struct task\_struct \*p2)

+{

+ return p1->tgid == p2->tgid;

+}

+

+static inline struct task\_struct \*next\_thread(const struct task\_struct \*p)

+{

+ return list\_entry(rcu\_dereference(p->thread\_group.next),

+ struct task\_struct, thread\_group);

+}

+

+static inline int thread\_group\_empty(struct task\_struct \*p)

+{

+ return list\_empty(&p->thread\_group);

+}

+

+#define delay\_group\_leader(p) \

+ (thread\_group\_leader(p) && !thread\_group\_empty(p))

+

+/\*

+ \* Protects ->fs, ->files, ->mm, ->group\_info, ->comm, keyring

+ \* subscriptions and synchronises with wait4(). Also used in procfs. Also

+ \* pins the final release of task.io\_context. Also protects ->cpuset and

+ \* ->cgroup.subsys[].

+ \*

+ \* Nests both inside and outside of read\_lock(&tasklist\_lock).

+ \* It must not be nested with write\_lock\_irq(&tasklist\_lock),

+ \* neither inside nor outside.

+ \*/

+static inline void task\_lock(struct task\_struct \*p)

+{

+ spin\_lock(&p->alloc\_lock);

+}

+

+static inline void task\_unlock(struct task\_struct \*p)

+{

+ spin\_unlock(&p->alloc\_lock);

+}

+

+extern struct sighand\_struct \*lock\_task\_sighand(struct task\_struct \*tsk,

+ unsigned long \*flags);

+

+static inline void unlock\_task\_sighand(struct task\_struct \*tsk,

+ unsigned long \*flags)

+{

+ spin\_unlock\_irqrestore(&tsk->sighand->siglock, \*flags);

+}

+

+#ifndef \_\_HAVE\_THREAD\_FUNCTIONS

+

+#define task\_thread\_info(task) ((struct thread\_info \*)(task)->stack)

+#define task\_stack\_page(task) ((task)->stack)

+

+static inline void setup\_thread\_stack(struct task\_struct \*p, struct task\_struct \*org)

+{

+ \*task\_thread\_info(p) = \*task\_thread\_info(org);

+ task\_thread\_info(p)->task = p;

+}

+

+static inline unsigned long \*end\_of\_stack(struct task\_struct \*p)

+{

+ return (unsigned long \*)(task\_thread\_info(p) + 1);

+}

+

+#endif

+

+/\* set thread flags in other task's structures

+ \* - see asm/thread\_info.h for TIF\_xxxx flags available

+ \*/

+static inline void set\_tsk\_thread\_flag(struct task\_struct \*tsk, int flag)

+{

+ set\_ti\_thread\_flag(task\_thread\_info(tsk), flag);

+}

+

+static inline void clear\_tsk\_thread\_flag(struct task\_struct \*tsk, int flag)

+{

+ clear\_ti\_thread\_flag(task\_thread\_info(tsk), flag);

+}

+

+static inline int test\_and\_set\_tsk\_thread\_flag(struct task\_struct \*tsk, int flag)

+{

+ return test\_and\_set\_ti\_thread\_flag(task\_thread\_info(tsk), flag);

+}

+

+static inline int test\_and\_clear\_tsk\_thread\_flag(struct task\_struct \*tsk, int flag)

+{

+ return test\_and\_clear\_ti\_thread\_flag(task\_thread\_info(tsk), flag);

+}

+

+static inline int test\_tsk\_thread\_flag(struct task\_struct \*tsk, int flag)

+{

+ return test\_ti\_thread\_flag(task\_thread\_info(tsk), flag);

+}

+

+static inline void set\_tsk\_need\_resched(struct task\_struct \*tsk)

+{

+ set\_tsk\_thread\_flag(tsk,TIF\_NEED\_RESCHED);

+}

+

+static inline void clear\_tsk\_need\_resched(struct task\_struct \*tsk)

+{

+ clear\_tsk\_thread\_flag(tsk,TIF\_NEED\_RESCHED);

+}

+

+static inline int signal\_pending(struct task\_struct \*p)

+{

+ return unlikely(test\_tsk\_thread\_flag(p,TIF\_SIGPENDING));

+}

+

+static inline int need\_resched(void)

+{

+ return unlikely(test\_thread\_flag(TIF\_NEED\_RESCHED));

+}

+

+/\*

+ \* cond\_resched() and cond\_resched\_lock(): latency reduction via

+ \* explicit rescheduling in places that are safe. The return

+ \* value indicates whether a reschedule was done in fact.

+ \* cond\_resched\_lock() will drop the spinlock before scheduling,

+ \* cond\_resched\_softirq() will enable bhs before scheduling.

+ \*/

+extern int cond\_resched(void);

+extern int cond\_resched\_lock(spinlock\_t \* lock);

+extern int cond\_resched\_softirq(void);

+

+/\*

+ \* Does a critical section need to be broken due to another

+ \* task waiting?:

+ \*/

+#if defined(CONFIG\_PREEMPT) && defined(CONFIG\_SMP)

+# define need\_lockbreak(lock) ((lock)->break\_lock)

+#else

+# define need\_lockbreak(lock) 0

+#endif

+

+/\*

+ \* Does a critical section need to be broken due to another

+ \* task waiting or preemption being signalled:

+ \*/

+static inline int lock\_need\_resched(spinlock\_t \*lock)

+{

+ if (need\_lockbreak(lock) || need\_resched())

+ return 1;

+ return 0;

+}

+

+/\*

+ \* Reevaluate whether the task has signals pending delivery.

+ \* Wake the task if so.

+ \* This is required every time the blocked sigset\_t changes.

+ \* callers must hold sighand->siglock.

+ \*/

+extern void recalc\_sigpending\_and\_wake(struct task\_struct \*t);

+extern void recalc\_sigpending(void);

+

+extern void signal\_wake\_up(struct task\_struct \*t, int resume\_stopped);

+

+/\*

+ \* Wrappers for p->thread\_info->cpu access. No-op on UP.

+ \*/

+#ifdef CONFIG\_SMP

+

+static inline unsigned int task\_cpu(const struct task\_struct \*p)

+{

+ return task\_thread\_info(p)->cpu;

+}

+

+extern void set\_task\_cpu(struct task\_struct \*p, unsigned int cpu);

+

+#else

+

+static inline unsigned int task\_cpu(const struct task\_struct \*p)

+{

+ return 0;

+}

+

+static inline void set\_task\_cpu(struct task\_struct \*p, unsigned int cpu)

+{

+}

+

+#endif /\* CONFIG\_SMP \*/

+

+#ifdef HAVE\_ARCH\_PICK\_MMAP\_LAYOUT

+extern void arch\_pick\_mmap\_layout(struct mm\_struct \*mm);

+#else

+static inline void arch\_pick\_mmap\_layout(struct mm\_struct \*mm)

+{

+ mm->mmap\_base = TASK\_UNMAPPED\_BASE;

+ mm->get\_unmapped\_area = arch\_get\_unmapped\_area;

+ mm->unmap\_area = arch\_unmap\_area;

+}

+#endif

+

+extern long sched\_setaffinity(pid\_t pid, cpumask\_t new\_mask);

+extern long sched\_getaffinity(pid\_t pid, cpumask\_t \*mask);

+

+extern int sched\_mc\_power\_savings, sched\_smt\_power\_savings;

+

+extern void normalize\_rt\_tasks(void);

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+

+extern struct task\_group init\_task\_group;

+

+extern struct task\_group \*sched\_create\_group(void);

+extern void sched\_destroy\_group(struct task\_group \*tg);

+extern void sched\_move\_task(struct task\_struct \*tsk);

+extern int sched\_group\_set\_shares(struct task\_group \*tg, unsigned long shares);

+extern unsigned long sched\_group\_shares(struct task\_group \*tg);

+

+#endif

+

+#ifdef CONFIG\_TASK\_XACCT

+static inline void add\_rchar(struct task\_struct \*tsk, ssize\_t amt)

+{

+ tsk->rchar += amt;

+}

+

+static inline void add\_wchar(struct task\_struct \*tsk, ssize\_t amt)

+{

+ tsk->wchar += amt;

+}

+

+static inline void inc\_syscr(struct task\_struct \*tsk)

+{

+ tsk->syscr++;

+}

+

+static inline void inc\_syscw(struct task\_struct \*tsk)

+{

+ tsk->syscw++;

+}

+#else

+static inline void add\_rchar(struct task\_struct \*tsk, ssize\_t amt)

+{

+}

+

+static inline void add\_wchar(struct task\_struct \*tsk, ssize\_t amt)

+{

+}

+

+static inline void inc\_syscr(struct task\_struct \*tsk)

+{

+}

+

+static inline void inc\_syscw(struct task\_struct \*tsk)

+{

+}

+#endif

+

+#ifdef CONFIG\_SMP

+void migration\_init(void);

+#else

+static inline void migration\_init(void)

+{

+}

+#endif

+

+#endif /\* \_\_KERNEL\_\_ \*/

+

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+

+#define CASIO\_MSG\_SIZE 400

+#define CASIO\_MAX\_EVENT\_LINES 10000

+

+#define CASIO\_ENQUEUE 1

+#define CASIO\_DEQUEUE 2

+#define CASIO\_CONTEXT\_SWITCH 3

+#define CASIO\_MSG 4

+

+struct casio\_event{

+ int action;

+ unsigned long long timestamp;

+ char msg[CASIO\_MSG\_SIZE];

+};

+

+struct casio\_event\_log{

+ struct casio\_event casio\_event[CASIO\_MAX\_EVENT\_LINES];

+ unsigned long lines;

+ unsigned long cursor;

+};

+void init\_casio\_event\_log();

+struct casio\_event\_log \* get\_casio\_event\_log();

+void register\_casio\_event(unsigned long long t, char \*m, int a);

+

+#endif

+

+#endif

diff -Nur linux-2.6.24/kernel/sched.c linux-2.6.24-casio/kernel/sched.c

--- linux-2.6.24/kernel/sched.c 2008-01-24 22:58:37.000000000 +0000

+++ linux-2.6.24-casio/kernel/sched.c 2009-06-28 18:13:18.000000000 +0100

@@ -135,8 +135,13 @@

static inline int rt\_policy(int policy)

{

- if (unlikely(policy == SCHED\_FIFO) || unlikely(policy == SCHED\_RR))

+ if (unlikely(policy == SCHED\_FIFO) || unlikely(policy == SCHED\_RR)

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ || unlikely(policy == SCHED\_CASIO)

+#endif

+ ){

return 1;

+ }

return 0;

}

@@ -267,7 +272,20 @@

int rt\_load\_balance\_idx;

struct list\_head \*rt\_load\_balance\_head, \*rt\_load\_balance\_curr;

};

-

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+struct casio\_task{

+ struct rb\_node casio\_rb\_node;

+ unsigned long long absolute\_deadline;

+ struct list\_head casio\_list\_node;

+ struct task\_struct \*task;

+

+};

+struct casio\_rq {

+ struct rb\_root casio\_rb\_root;

+ struct list\_head casio\_list\_head;

+ atomic\_t nr\_running;

+};

+#endif

/\*

\* This is the main, per-CPU runqueue data structure.

\*

@@ -302,6 +320,10 @@

#endif

struct rt\_rq rt;

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ struct casio\_rq casio\_rq;

+#endif

+

/\*

\* This is part of a global counter where only the total sum

\* over all CPUs matters. A task can increase this counter on

@@ -878,8 +900,16 @@

#ifdef CONFIG\_SCHED\_DEBUG

# include "sched\_debug.c"

#endif

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#include "sched\_casio.c"

+#endif

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ #define sched\_class\_highest (&casio\_sched\_class)

+#else

+ #define sched\_class\_highest (&rt\_sched\_class)

+#endif

-#define sched\_class\_highest (&rt\_sched\_class)

/\*

\* Update delta\_exec, delta\_fair fields for rq.

@@ -3594,12 +3624,13 @@

\* Optimization: we know that if all tasks are in

\* the fair class we can call that function directly:

\*/

+/\*

if (likely(rq->nr\_running == rq->cfs.nr\_running)) {

p = fair\_sched\_class.pick\_next\_task(rq);

if (likely(p))

return p;

}

-

+\*/

class = sched\_class\_highest;

for ( ; ; ) {

p = class->pick\_next\_task(rq);

@@ -3660,6 +3691,25 @@

prev->sched\_class->put\_prev\_task(rq, prev);

next = pick\_next\_task(rq, prev);

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ char msg[CASIO\_MSG\_SIZE];

+ if(prev->policy==SCHED\_CASIO || next->policy==SCHED\_CASIO){

+ if(prev->policy==SCHED\_CASIO && next->policy==SCHED\_CASIO){

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(%d:%d),next->(%d:%d)",prev->casio\_id,prev->pid,next->casio\_id,next->pid);

+ }

+ else{

+ if(prev->policy==SCHED\_CASIO){

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(%d:%d),next->(-1:%d)",prev->casio\_id,prev->pid,next->pid);

+ }else{

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(-1:%d),next->(%d:%d)",prev->pid,next->casio\_id,next->pid);

+ }

+ }

+ register\_casio\_event(sched\_clock(), msg, CASIO\_CONTEXT\_SWITCH);

+

+

+ }

+#endif

+

sched\_info\_switch(prev, next);

if (likely(prev != next)) {

@@ -4236,6 +4286,11 @@

case SCHED\_RR:

p->sched\_class = &rt\_sched\_class;

break;

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ case SCHED\_CASIO:

+ p->sched\_class = &casio\_sched\_class;

+ break;

+#endif

}

p->rt\_priority = prio;

@@ -4259,7 +4314,6 @@

int retval, oldprio, oldpolicy = -1, on\_rq, running;

unsigned long flags;

struct rq \*rq;

-

/\* may grab non-irq protected spin\_locks \*/

BUG\_ON(in\_interrupt());

recheck:

@@ -4268,7 +4322,11 @@

policy = oldpolicy = p->policy;

else if (policy != SCHED\_FIFO && policy != SCHED\_RR &&

policy != SCHED\_NORMAL && policy != SCHED\_BATCH &&

- policy != SCHED\_IDLE)

+ policy != SCHED\_IDLE

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ && policy!=SCHED\_CASIO

+#endif

+ )

return -EINVAL;

/\*

\* Valid priorities for SCHED\_FIFO and SCHED\_RR are

@@ -4315,6 +4373,12 @@

(current->euid != p->uid))

return -EPERM;

}

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ if(policy==SCHED\_CASIO){

+ p->deadline=param->deadline;

+ p->casio\_id=param->casio\_id;

+ }

+#endif

retval = security\_task\_setscheduler(p, policy, param);

if (retval)

@@ -4336,6 +4400,11 @@

spin\_unlock\_irqrestore(&p->pi\_lock, flags);

goto recheck;

}

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ if(policy==SCHED\_CASIO){

+ add\_casio\_task\_2\_list(&rq->casio\_rq, p);

+ }

+#endif

update\_rq\_clock(rq);

on\_rq = p->se.on\_rq;

running = task\_current(rq, p);

@@ -6760,7 +6829,13 @@

lockdep\_set\_class(&rq->lock, &rq->rq\_lock\_key);

rq->nr\_running = 0;

rq->clock = 1;

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ init\_casio\_rq(&rq->casio\_rq);

+#endif

init\_cfs\_rq(&rq->cfs, rq);

+

+

+

#ifdef CONFIG\_FAIR\_GROUP\_SCHED

INIT\_LIST\_HEAD(&rq->leaf\_cfs\_rq\_list);

{

@@ -6782,10 +6857,16 @@

div64\_64(1ULL<<32, init\_task\_group\_load);

se->parent = NULL;

}

+

init\_task\_group.shares = init\_task\_group\_load;

spin\_lock\_init(&init\_task\_group.lock);

#endif

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+

+ init\_casio\_event\_log();

+#endif

+

for (j = 0; j < CPU\_LOAD\_IDX\_MAX; j++)

rq->cpu\_load[j] = 0;

#ifdef CONFIG\_SMP

diff -Nur linux-2.6.24/kernel/sched.c~ linux-2.6.24-casio/kernel/sched.c~

--- linux-2.6.24/kernel/sched.c~ 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/kernel/sched.c~ 2009-06-28 18:02:48.000000000 +0100

@@ -0,0 +1,7485 @@

+/\*

+ \* kernel/sched.c

+ \*

+ \* Kernel scheduler and related syscalls

+ \*

+ \* Copyright (C) 1991-2002 Linus Torvalds

+ \*

+ \* 1996-12-23 Modified by Dave Grothe to fix bugs in semaphores and

+ \* make semaphores SMP safe

+ \* 1998-11-19 Implemented schedule\_timeout() and related stuff

+ \* by Andrea Arcangeli

+ \* 2002-01-04 New ultra-scalable O(1) scheduler by Ingo Molnar:

+ \* hybrid priority-list and round-robin design with

+ \* an array-switch method of distributing timeslices

+ \* and per-CPU runqueues. Cleanups and useful suggestions

+ \* by Davide Libenzi, preemptible kernel bits by Robert Love.

+ \* 2003-09-03 Interactivity tuning by Con Kolivas.

+ \* 2004-04-02 Scheduler domains code by Nick Piggin

+ \* 2007-04-15 Work begun on replacing all interactivity tuning with a

+ \* fair scheduling design by Con Kolivas.

+ \* 2007-05-05 Load balancing (smp-nice) and other improvements

+ \* by Peter Williams

+ \* 2007-05-06 Interactivity improvements to CFS by Mike Galbraith

+ \* 2007-07-01 Group scheduling enhancements by Srivatsa Vaddagiri

+ \*/

+

+#include <linux/mm.h>

+#include <linux/module.h>

+#include <linux/nmi.h>

+#include <linux/init.h>

+#include <linux/uaccess.h>

+#include <linux/highmem.h>

+#include <linux/smp\_lock.h>

+#include <asm/mmu\_context.h>

+#include <linux/interrupt.h>

+#include <linux/capability.h>

+#include <linux/completion.h>

+#include <linux/kernel\_stat.h>

+#include <linux/debug\_locks.h>

+#include <linux/security.h>

+#include <linux/notifier.h>

+#include <linux/profile.h>

+#include <linux/freezer.h>

+#include <linux/vmalloc.h>

+#include <linux/blkdev.h>

+#include <linux/delay.h>

+#include <linux/pid\_namespace.h>

+#include <linux/smp.h>

+#include <linux/threads.h>

+#include <linux/timer.h>

+#include <linux/rcupdate.h>

+#include <linux/cpu.h>

+#include <linux/cpuset.h>

+#include <linux/percpu.h>

+#include <linux/kthread.h>

+#include <linux/seq\_file.h>

+#include <linux/sysctl.h>

+#include <linux/syscalls.h>

+#include <linux/times.h>

+#include <linux/tsacct\_kern.h>

+#include <linux/kprobes.h>

+#include <linux/delayacct.h>

+#include <linux/reciprocal\_div.h>

+#include <linux/unistd.h>

+#include <linux/pagemap.h>

+

+#include <asm/tlb.h>

+#include <asm/irq\_regs.h>

+

+/\*

+ \* Scheduler clock - returns current time in nanosec units.

+ \* This is default implementation.

+ \* Architectures and sub-architectures can override this.

+ \*/

+unsigned long long \_\_attribute\_\_((weak)) sched\_clock(void)

+{

+ return (unsigned long long)jiffies \* (NSEC\_PER\_SEC / HZ);

+}

+

+/\*

+ \* Convert user-nice values [ -20 ... 0 ... 19 ]

+ \* to static priority [ MAX\_RT\_PRIO..MAX\_PRIO-1 ],

+ \* and back.

+ \*/

+#define NICE\_TO\_PRIO(nice) (MAX\_RT\_PRIO + (nice) + 20)

+#define PRIO\_TO\_NICE(prio) ((prio) - MAX\_RT\_PRIO - 20)

+#define TASK\_NICE(p) PRIO\_TO\_NICE((p)->static\_prio)

+

+/\*

+ \* 'User priority' is the nice value converted to something we

+ \* can work with better when scaling various scheduler parameters,

+ \* it's a [ 0 ... 39 ] range.

+ \*/

+#define USER\_PRIO(p) ((p)-MAX\_RT\_PRIO)

+#define TASK\_USER\_PRIO(p) USER\_PRIO((p)->static\_prio)

+#define MAX\_USER\_PRIO (USER\_PRIO(MAX\_PRIO))

+

+/\*

+ \* Some helpers for converting nanosecond timing to jiffy resolution

+ \*/

+#define NS\_TO\_JIFFIES(TIME) ((unsigned long)(TIME) / (NSEC\_PER\_SEC / HZ))

+#define JIFFIES\_TO\_NS(TIME) ((TIME) \* (NSEC\_PER\_SEC / HZ))

+

+#define NICE\_0\_LOAD SCHED\_LOAD\_SCALE

+#define NICE\_0\_SHIFT SCHED\_LOAD\_SHIFT

+

+/\*

+ \* These are the 'tuning knobs' of the scheduler:

+ \*

+ \* default timeslice is 100 msecs (used only for SCHED\_RR tasks).

+ \* Timeslices get refilled after they expire.

+ \*/

+#define DEF\_TIMESLICE (100 \* HZ / 1000)

+

+#ifdef CONFIG\_SMP

+/\*

+ \* Divide a load by a sched group cpu\_power : (load / sg->\_\_cpu\_power)

+ \* Since cpu\_power is a 'constant', we can use a reciprocal divide.

+ \*/

+static inline u32 sg\_div\_cpu\_power(const struct sched\_group \*sg, u32 load)

+{

+ return reciprocal\_divide(load, sg->reciprocal\_cpu\_power);

+}

+

+/\*

+ \* Each time a sched group cpu\_power is changed,

+ \* we must compute its reciprocal value

+ \*/

+static inline void sg\_inc\_cpu\_power(struct sched\_group \*sg, u32 val)

+{

+ sg->\_\_cpu\_power += val;

+ sg->reciprocal\_cpu\_power = reciprocal\_value(sg->\_\_cpu\_power);

+}

+#endif

+

+static inline int rt\_policy(int policy)

+{

+ if (unlikely(policy == SCHED\_FIFO) || unlikely(policy == SCHED\_RR)

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ || unlikely(policy == SCHED\_CASIO)

+#endif

+ ){

+ return 1;

+ }

+ return 0;

+}

+

+static inline int task\_has\_rt\_policy(struct task\_struct \*p)

+{

+ return rt\_policy(p->policy);

+}

+

+/\*

+ \* This is the priority-queue data structure of the RT scheduling class:

+ \*/

+struct rt\_prio\_array {

+ DECLARE\_BITMAP(bitmap, MAX\_RT\_PRIO+1); /\* include 1 bit for delimiter \*/

+ struct list\_head queue[MAX\_RT\_PRIO];

+};

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+

+#include <linux/cgroup.h>

+

+struct cfs\_rq;

+

+/\* task group related information \*/

+struct task\_group {

+#ifdef CONFIG\_FAIR\_CGROUP\_SCHED

+ struct cgroup\_subsys\_state css;

+#endif

+ /\* schedulable entities of this group on each cpu \*/

+ struct sched\_entity \*\*se;

+ /\* runqueue "owned" by this group on each cpu \*/

+ struct cfs\_rq \*\*cfs\_rq;

+ unsigned long shares;

+ /\* spinlock to serialize modification to shares \*/

+ spinlock\_t lock;

+ struct rcu\_head rcu;

+};

+

+/\* Default task group's sched entity on each cpu \*/

+static DEFINE\_PER\_CPU(struct sched\_entity, init\_sched\_entity);

+/\* Default task group's cfs\_rq on each cpu \*/

+static DEFINE\_PER\_CPU(struct cfs\_rq, init\_cfs\_rq) \_\_\_\_cacheline\_aligned\_in\_smp;

+

+static struct sched\_entity \*init\_sched\_entity\_p[NR\_CPUS];

+static struct cfs\_rq \*init\_cfs\_rq\_p[NR\_CPUS];

+

+/\* Default task group.

+ \* Every task in system belong to this group at bootup.

+ \*/

+struct task\_group init\_task\_group = {

+ .se = init\_sched\_entity\_p,

+ .cfs\_rq = init\_cfs\_rq\_p,

+};

+

+#ifdef CONFIG\_FAIR\_USER\_SCHED

+# define INIT\_TASK\_GRP\_LOAD 2\*NICE\_0\_LOAD

+#else

+# define INIT\_TASK\_GRP\_LOAD NICE\_0\_LOAD

+#endif

+

+static int init\_task\_group\_load = INIT\_TASK\_GRP\_LOAD;

+

+/\* return group to which a task belongs \*/

+static inline struct task\_group \*task\_group(struct task\_struct \*p)

+{

+ struct task\_group \*tg;

+

+#ifdef CONFIG\_FAIR\_USER\_SCHED

+ tg = p->user->tg;

+#elif defined(CONFIG\_FAIR\_CGROUP\_SCHED)

+ tg = container\_of(task\_subsys\_state(p, cpu\_cgroup\_subsys\_id),

+ struct task\_group, css);

+#else

+ tg = &init\_task\_group;

+#endif

+ return tg;

+}

+

+/\* Change a task's cfs\_rq and parent entity if it moves across CPUs/groups \*/

+static inline void set\_task\_cfs\_rq(struct task\_struct \*p, unsigned int cpu)

+{

+ p->se.cfs\_rq = task\_group(p)->cfs\_rq[cpu];

+ p->se.parent = task\_group(p)->se[cpu];

+}

+

+#else

+

+static inline void set\_task\_cfs\_rq(struct task\_struct \*p, unsigned int cpu) { }

+

+#endif /\* CONFIG\_FAIR\_GROUP\_SCHED \*/

+

+/\* CFS-related fields in a runqueue \*/

+struct cfs\_rq {

+ struct load\_weight load;

+ unsigned long nr\_running;

+

+ u64 exec\_clock;

+ u64 min\_vruntime;

+

+ struct rb\_root tasks\_timeline;

+ struct rb\_node \*rb\_leftmost;

+ struct rb\_node \*rb\_load\_balance\_curr;

+ /\* 'curr' points to currently running entity on this cfs\_rq.

+ \* It is set to NULL otherwise (i.e when none are currently running).

+ \*/

+ struct sched\_entity \*curr;

+

+ unsigned long nr\_spread\_over;

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+ struct rq \*rq; /\* cpu runqueue to which this cfs\_rq is attached \*/

+

+ /\*

+ \* leaf cfs\_rqs are those that hold tasks (lowest schedulable entity in

+ \* a hierarchy). Non-leaf lrqs hold other higher schedulable entities

+ \* (like users, containers etc.)

+ \*

+ \* leaf\_cfs\_rq\_list ties together list of leaf cfs\_rq's in a cpu. This

+ \* list is used during load balance.

+ \*/

+ struct list\_head leaf\_cfs\_rq\_list;

+ struct task\_group \*tg; /\* group that "owns" this runqueue \*/

+#endif

+};

+

+/\* Real-Time classes' related field in a runqueue: \*/

+struct rt\_rq {

+ struct rt\_prio\_array active;

+ int rt\_load\_balance\_idx;

+ struct list\_head \*rt\_load\_balance\_head, \*rt\_load\_balance\_curr;

+};

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+struct casio\_task{

+ struct rb\_node casio\_rb\_node;

+ unsigned long long absolute\_deadline;

+ struct list\_head casio\_list\_node;

+ struct task\_struct \*task;

+

+};

+struct casio\_rq {

+ struct rb\_root casio\_rb\_root;

+ struct list\_head \*casio\_list\_head;

+ atomic\_t nr\_running;

+};

+#endif

+/\*

+ \* This is the main, per-CPU runqueue data structure.

+ \*

+ \* Locking rule: those places that want to lock multiple runqueues

+ \* (such as the load balancing or the thread migration code), lock

+ \* acquire operations must be ordered by ascending &runqueue.

+ \*/

+struct rq {

+ /\* runqueue lock: \*/

+ spinlock\_t lock;

+

+ /\*

+ \* nr\_running and cpu\_load should be in the same cacheline because

+ \* remote CPUs use both these fields when doing load calculation.

+ \*/

+ unsigned long nr\_running;

+ #define CPU\_LOAD\_IDX\_MAX 5

+ unsigned long cpu\_load[CPU\_LOAD\_IDX\_MAX];

+ unsigned char idle\_at\_tick;

+#ifdef CONFIG\_NO\_HZ

+ unsigned char in\_nohz\_recently;

+#endif

+ /\* capture load from \*all\* tasks on this cpu: \*/

+ struct load\_weight load;

+ unsigned long nr\_load\_updates;

+ u64 nr\_switches;

+

+ struct cfs\_rq cfs;

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+ /\* list of leaf cfs\_rq on this cpu: \*/

+ struct list\_head leaf\_cfs\_rq\_list;

+#endif

+ struct rt\_rq rt;

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ struct casio\_rq casio\_rq;

+#endif

+

+ /\*

+ \* This is part of a global counter where only the total sum

+ \* over all CPUs matters. A task can increase this counter on

+ \* one CPU and if it got migrated afterwards it may decrease

+ \* it on another CPU. Always updated under the runqueue lock:

+ \*/

+ unsigned long nr\_uninterruptible;

+

+ struct task\_struct \*curr, \*idle;

+ unsigned long next\_balance;

+ struct mm\_struct \*prev\_mm;

+

+ u64 clock, prev\_clock\_raw;

+ s64 clock\_max\_delta;

+

+ unsigned int clock\_warps, clock\_overflows;

+ u64 idle\_clock;

+ unsigned int clock\_deep\_idle\_events;

+ u64 tick\_timestamp;

+

+ atomic\_t nr\_iowait;

+

+#ifdef CONFIG\_SMP

+ struct sched\_domain \*sd;

+

+ /\* For active balancing \*/

+ int active\_balance;

+ int push\_cpu;

+ /\* cpu of this runqueue: \*/

+ int cpu;

+

+ struct task\_struct \*migration\_thread;

+ struct list\_head migration\_queue;

+#endif

+

+#ifdef CONFIG\_SCHEDSTATS

+ /\* latency stats \*/

+ struct sched\_info rq\_sched\_info;

+

+ /\* sys\_sched\_yield() stats \*/

+ unsigned int yld\_exp\_empty;

+ unsigned int yld\_act\_empty;

+ unsigned int yld\_both\_empty;

+ unsigned int yld\_count;

+

+ /\* schedule() stats \*/

+ unsigned int sched\_switch;

+ unsigned int sched\_count;

+ unsigned int sched\_goidle;

+

+ /\* try\_to\_wake\_up() stats \*/

+ unsigned int ttwu\_count;

+ unsigned int ttwu\_local;

+

+ /\* BKL stats \*/

+ unsigned int bkl\_count;

+#endif

+ struct lock\_class\_key rq\_lock\_key;

+};

+

+static DEFINE\_PER\_CPU\_SHARED\_ALIGNED(struct rq, runqueues);

+static DEFINE\_MUTEX(sched\_hotcpu\_mutex);

+

+static inline void check\_preempt\_curr(struct rq \*rq, struct task\_struct \*p)

+{

+ rq->curr->sched\_class->check\_preempt\_curr(rq, p);

+}

+

+static inline int cpu\_of(struct rq \*rq)

+{

+#ifdef CONFIG\_SMP

+ return rq->cpu;

+#else

+ return 0;

+#endif

+}

+

+/\*

+ \* Update the per-runqueue clock, as finegrained as the platform can give

+ \* us, but without assuming monotonicity, etc.:

+ \*/

+static void \_\_update\_rq\_clock(struct rq \*rq)

+{

+ u64 prev\_raw = rq->prev\_clock\_raw;

+ u64 now = sched\_clock();

+ s64 delta = now - prev\_raw;

+ u64 clock = rq->clock;

+

+#ifdef CONFIG\_SCHED\_DEBUG

+ WARN\_ON\_ONCE(cpu\_of(rq) != smp\_processor\_id());

+#endif

+ /\*

+ \* Protect against sched\_clock() occasionally going backwards:

+ \*/

+ if (unlikely(delta < 0)) {

+ clock++;

+ rq->clock\_warps++;

+ } else {

+ /\*

+ \* Catch too large forward jumps too:

+ \*/

+ if (unlikely(clock + delta > rq->tick\_timestamp + TICK\_NSEC)) {

+ if (clock < rq->tick\_timestamp + TICK\_NSEC)

+ clock = rq->tick\_timestamp + TICK\_NSEC;

+ else

+ clock++;

+ rq->clock\_overflows++;

+ } else {

+ if (unlikely(delta > rq->clock\_max\_delta))

+ rq->clock\_max\_delta = delta;

+ clock += delta;

+ }

+ }

+

+ rq->prev\_clock\_raw = now;

+ rq->clock = clock;

+}

+

+static void update\_rq\_clock(struct rq \*rq)

+{

+ if (likely(smp\_processor\_id() == cpu\_of(rq)))

+ \_\_update\_rq\_clock(rq);

+}

+

+/\*

+ \* The domain tree (rq->sd) is protected by RCU's quiescent state transition.

+ \* See detach\_destroy\_domains: synchronize\_sched for details.

+ \*

+ \* The domain tree of any CPU may only be accessed from within

+ \* preempt-disabled sections.

+ \*/

+#define for\_each\_domain(cpu, \_\_sd) \

+ for (\_\_sd = rcu\_dereference(cpu\_rq(cpu)->sd); \_\_sd; \_\_sd = \_\_sd->parent)

+

+#define cpu\_rq(cpu) (&per\_cpu(runqueues, (cpu)))

+#define this\_rq() (&\_\_get\_cpu\_var(runqueues))

+#define task\_rq(p) cpu\_rq(task\_cpu(p))

+#define cpu\_curr(cpu) (cpu\_rq(cpu)->curr)

+

+/\*

+ \* Tunables that become constants when CONFIG\_SCHED\_DEBUG is off:

+ \*/

+#ifdef CONFIG\_SCHED\_DEBUG

+# define const\_debug \_\_read\_mostly

+#else

+# define const\_debug static const

+#endif

+

+/\*

+ \* Debugging: various feature bits

+ \*/

+enum {

+ SCHED\_FEAT\_NEW\_FAIR\_SLEEPERS = 1,

+ SCHED\_FEAT\_WAKEUP\_PREEMPT = 2,

+ SCHED\_FEAT\_START\_DEBIT = 4,

+ SCHED\_FEAT\_TREE\_AVG = 8,

+ SCHED\_FEAT\_APPROX\_AVG = 16,

+};

+

+const\_debug unsigned int sysctl\_sched\_features =

+ SCHED\_FEAT\_NEW\_FAIR\_SLEEPERS \* 1 |

+ SCHED\_FEAT\_WAKEUP\_PREEMPT \* 1 |

+ SCHED\_FEAT\_START\_DEBIT \* 1 |

+ SCHED\_FEAT\_TREE\_AVG \* 0 |

+ SCHED\_FEAT\_APPROX\_AVG \* 0;

+

+#define sched\_feat(x) (sysctl\_sched\_features & SCHED\_FEAT\_##x)

+

+/\*

+ \* Number of tasks to iterate in a single balance run.

+ \* Limited because this is done with IRQs disabled.

+ \*/

+const\_debug unsigned int sysctl\_sched\_nr\_migrate = 32;

+

+/\*

+ \* For kernel-internal use: high-speed (but slightly incorrect) per-cpu

+ \* clock constructed from sched\_clock():

+ \*/

+unsigned long long cpu\_clock(int cpu)

+{

+ unsigned long long now;

+ unsigned long flags;

+ struct rq \*rq;

+

+ local\_irq\_save(flags);

+ rq = cpu\_rq(cpu);

+ /\*

+ \* Only call sched\_clock() if the scheduler has already been

+ \* initialized (some code might call cpu\_clock() very early):

+ \*/

+ if (rq->idle)

+ update\_rq\_clock(rq);

+ now = rq->clock;

+ local\_irq\_restore(flags);

+

+ return now;

+}

+EXPORT\_SYMBOL\_GPL(cpu\_clock);

+

+#ifndef prepare\_arch\_switch

+# define prepare\_arch\_switch(next) do { } while (0)

+#endif

+#ifndef finish\_arch\_switch

+# define finish\_arch\_switch(prev) do { } while (0)

+#endif

+

+static inline int task\_current(struct rq \*rq, struct task\_struct \*p)

+{

+ return rq->curr == p;

+}

+

+#ifndef \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

+static inline int task\_running(struct rq \*rq, struct task\_struct \*p)

+{

+ return task\_current(rq, p);

+}

+

+static inline void prepare\_lock\_switch(struct rq \*rq, struct task\_struct \*next)

+{

+}

+

+static inline void finish\_lock\_switch(struct rq \*rq, struct task\_struct \*prev)

+{

+#ifdef CONFIG\_DEBUG\_SPINLOCK

+ /\* this is a valid case when another task releases the spinlock \*/

+ rq->lock.owner = current;

+#endif

+ /\*

+ \* If we are tracking spinlock dependencies then we have to

+ \* fix up the runqueue lock - which gets 'carried over' from

+ \* prev into current:

+ \*/

+ spin\_acquire(&rq->lock.dep\_map, 0, 0, \_THIS\_IP\_);

+

+ spin\_unlock\_irq(&rq->lock);

+}

+

+#else /\* \_\_ARCH\_WANT\_UNLOCKED\_CTXSW \*/

+static inline int task\_running(struct rq \*rq, struct task\_struct \*p)

+{

+#ifdef CONFIG\_SMP

+ return p->oncpu;

+#else

+ return task\_current(rq, p);

+#endif

+}

+

+static inline void prepare\_lock\_switch(struct rq \*rq, struct task\_struct \*next)

+{

+#ifdef CONFIG\_SMP

+ /\*

+ \* We can optimise this out completely for !SMP, because the

+ \* SMP rebalancing from interrupt is the only thing that cares

+ \* here.

+ \*/

+ next->oncpu = 1;

+#endif

+#ifdef \_\_ARCH\_WANT\_INTERRUPTS\_ON\_CTXSW

+ spin\_unlock\_irq(&rq->lock);

+#else

+ spin\_unlock(&rq->lock);

+#endif

+}

+

+static inline void finish\_lock\_switch(struct rq \*rq, struct task\_struct \*prev)

+{

+#ifdef CONFIG\_SMP

+ /\*

+ \* After ->oncpu is cleared, the task can be moved to a different CPU.

+ \* We must ensure this doesn't happen until the switch is completely

+ \* finished.

+ \*/

+ smp\_wmb();

+ prev->oncpu = 0;

+#endif

+#ifndef \_\_ARCH\_WANT\_INTERRUPTS\_ON\_CTXSW

+ local\_irq\_enable();

+#endif

+}

+#endif /\* \_\_ARCH\_WANT\_UNLOCKED\_CTXSW \*/

+

+/\*

+ \* \_\_task\_rq\_lock - lock the runqueue a given task resides on.

+ \* Must be called interrupts disabled.

+ \*/

+static inline struct rq \*\_\_task\_rq\_lock(struct task\_struct \*p)

+ \_\_acquires(rq->lock)

+{

+ for (;;) {

+ struct rq \*rq = task\_rq(p);

+ spin\_lock(&rq->lock);

+ if (likely(rq == task\_rq(p)))

+ return rq;

+ spin\_unlock(&rq->lock);

+ }

+}

+

+/\*

+ \* task\_rq\_lock - lock the runqueue a given task resides on and disable

+ \* interrupts. Note the ordering: we can safely lookup the task\_rq without

+ \* explicitly disabling preemption.

+ \*/

+static struct rq \*task\_rq\_lock(struct task\_struct \*p, unsigned long \*flags)

+ \_\_acquires(rq->lock)

+{

+ struct rq \*rq;

+

+ for (;;) {

+ local\_irq\_save(\*flags);

+ rq = task\_rq(p);

+ spin\_lock(&rq->lock);

+ if (likely(rq == task\_rq(p)))

+ return rq;

+ spin\_unlock\_irqrestore(&rq->lock, \*flags);

+ }

+}

+

+static void \_\_task\_rq\_unlock(struct rq \*rq)

+ \_\_releases(rq->lock)

+{

+ spin\_unlock(&rq->lock);

+}

+

+static inline void task\_rq\_unlock(struct rq \*rq, unsigned long \*flags)

+ \_\_releases(rq->lock)

+{

+ spin\_unlock\_irqrestore(&rq->lock, \*flags);

+}

+

+/\*

+ \* this\_rq\_lock - lock this runqueue and disable interrupts.

+ \*/

+static struct rq \*this\_rq\_lock(void)

+ \_\_acquires(rq->lock)

+{

+ struct rq \*rq;

+

+ local\_irq\_disable();

+ rq = this\_rq();

+ spin\_lock(&rq->lock);

+

+ return rq;

+}

+

+/\*

+ \* We are going deep-idle (irqs are disabled):

+ \*/

+void sched\_clock\_idle\_sleep\_event(void)

+{

+ struct rq \*rq = cpu\_rq(smp\_processor\_id());

+

+ spin\_lock(&rq->lock);

+ \_\_update\_rq\_clock(rq);

+ spin\_unlock(&rq->lock);

+ rq->clock\_deep\_idle\_events++;

+}

+EXPORT\_SYMBOL\_GPL(sched\_clock\_idle\_sleep\_event);

+

+/\*

+ \* We just idled delta nanoseconds (called with irqs disabled):

+ \*/

+void sched\_clock\_idle\_wakeup\_event(u64 delta\_ns)

+{

+ struct rq \*rq = cpu\_rq(smp\_processor\_id());

+ u64 now = sched\_clock();

+

+ touch\_softlockup\_watchdog();

+ rq->idle\_clock += delta\_ns;

+ /\*

+ \* Override the previous timestamp and ignore all

+ \* sched\_clock() deltas that occured while we idled,

+ \* and use the PM-provided delta\_ns to advance the

+ \* rq clock:

+ \*/

+ spin\_lock(&rq->lock);

+ rq->prev\_clock\_raw = now;

+ rq->clock += delta\_ns;

+ spin\_unlock(&rq->lock);

+}

+EXPORT\_SYMBOL\_GPL(sched\_clock\_idle\_wakeup\_event);

+

+/\*

+ \* resched\_task - mark a task 'to be rescheduled now'.

+ \*

+ \* On UP this means the setting of the need\_resched flag, on SMP it

+ \* might also involve a cross-CPU call to trigger the scheduler on

+ \* the target CPU.

+ \*/

+#ifdef CONFIG\_SMP

+

+#ifndef tsk\_is\_polling

+#define tsk\_is\_polling(t) test\_tsk\_thread\_flag(t, TIF\_POLLING\_NRFLAG)

+#endif

+

+static void resched\_task(struct task\_struct \*p)

+{

+ int cpu;

+

+ assert\_spin\_locked(&task\_rq(p)->lock);

+

+ if (unlikely(test\_tsk\_thread\_flag(p, TIF\_NEED\_RESCHED)))

+ return;

+

+ set\_tsk\_thread\_flag(p, TIF\_NEED\_RESCHED);

+

+ cpu = task\_cpu(p);

+ if (cpu == smp\_processor\_id())

+ return;

+

+ /\* NEED\_RESCHED must be visible before we test polling \*/

+ smp\_mb();

+ if (!tsk\_is\_polling(p))

+ smp\_send\_reschedule(cpu);

+}

+

+static void resched\_cpu(int cpu)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long flags;

+

+ if (!spin\_trylock\_irqsave(&rq->lock, flags))

+ return;

+ resched\_task(cpu\_curr(cpu));

+ spin\_unlock\_irqrestore(&rq->lock, flags);

+}

+#else

+static inline void resched\_task(struct task\_struct \*p)

+{

+ assert\_spin\_locked(&task\_rq(p)->lock);

+ set\_tsk\_need\_resched(p);

+}

+#endif

+

+#if BITS\_PER\_LONG == 32

+# define WMULT\_CONST (~0UL)

+#else

+# define WMULT\_CONST (1UL << 32)

+#endif

+

+#define WMULT\_SHIFT 32

+

+/\*

+ \* Shift right and round:

+ \*/

+#define SRR(x, y) (((x) + (1UL << ((y) - 1))) >> (y))

+

+static unsigned long

+calc\_delta\_mine(unsigned long delta\_exec, unsigned long weight,

+ struct load\_weight \*lw)

+{

+ u64 tmp;

+

+ if (unlikely(!lw->inv\_weight))

+ lw->inv\_weight = (WMULT\_CONST - lw->weight/2) / lw->weight + 1;

+

+ tmp = (u64)delta\_exec \* weight;

+ /\*

+ \* Check whether we'd overflow the 64-bit multiplication:

+ \*/

+ if (unlikely(tmp > WMULT\_CONST))

+ tmp = SRR(SRR(tmp, WMULT\_SHIFT/2) \* lw->inv\_weight,

+ WMULT\_SHIFT/2);

+ else

+ tmp = SRR(tmp \* lw->inv\_weight, WMULT\_SHIFT);

+

+ return (unsigned long)min(tmp, (u64)(unsigned long)LONG\_MAX);

+}

+

+static inline unsigned long

+calc\_delta\_fair(unsigned long delta\_exec, struct load\_weight \*lw)

+{

+ return calc\_delta\_mine(delta\_exec, NICE\_0\_LOAD, lw);

+}

+

+static inline void update\_load\_add(struct load\_weight \*lw, unsigned long inc)

+{

+ lw->weight += inc;

+}

+

+static inline void update\_load\_sub(struct load\_weight \*lw, unsigned long dec)

+{

+ lw->weight -= dec;

+}

+

+/\*

+ \* To aid in avoiding the subversion of "niceness" due to uneven distribution

+ \* of tasks with abnormal "nice" values across CPUs the contribution that

+ \* each task makes to its run queue's load is weighted according to its

+ \* scheduling class and "nice" value. For SCHED\_NORMAL tasks this is just a

+ \* scaled version of the new time slice allocation that they receive on time

+ \* slice expiry etc.

+ \*/

+

+#define WEIGHT\_IDLEPRIO 2

+#define WMULT\_IDLEPRIO (1 << 31)

+

+/\*

+ \* Nice levels are multiplicative, with a gentle 10% change for every

+ \* nice level changed. I.e. when a CPU-bound task goes from nice 0 to

+ \* nice 1, it will get ~10% less CPU time than another CPU-bound task

+ \* that remained on nice 0.

+ \*

+ \* The "10% effect" is relative and cumulative: from \_any\_ nice level,

+ \* if you go up 1 level, it's -10% CPU usage, if you go down 1 level

+ \* it's +10% CPU usage. (to achieve that we use a multiplier of 1.25.

+ \* If a task goes up by ~10% and another task goes down by ~10% then

+ \* the relative distance between them is ~25%.)

+ \*/

+static const int prio\_to\_weight[40] = {

+ /\* -20 \*/ 88761, 71755, 56483, 46273, 36291,

+ /\* -15 \*/ 29154, 23254, 18705, 14949, 11916,

+ /\* -10 \*/ 9548, 7620, 6100, 4904, 3906,

+ /\* -5 \*/ 3121, 2501, 1991, 1586, 1277,

+ /\* 0 \*/ 1024, 820, 655, 526, 423,

+ /\* 5 \*/ 335, 272, 215, 172, 137,

+ /\* 10 \*/ 110, 87, 70, 56, 45,

+ /\* 15 \*/ 36, 29, 23, 18, 15,

+};

+

+/\*

+ \* Inverse (2^32/x) values of the prio\_to\_weight[] array, precalculated.

+ \*

+ \* In cases where the weight does not change often, we can use the

+ \* precalculated inverse to speed up arithmetics by turning divisions

+ \* into multiplications:

+ \*/

+static const u32 prio\_to\_wmult[40] = {

+ /\* -20 \*/ 48388, 59856, 76040, 92818, 118348,

+ /\* -15 \*/ 147320, 184698, 229616, 287308, 360437,

+ /\* -10 \*/ 449829, 563644, 704093, 875809, 1099582,

+ /\* -5 \*/ 1376151, 1717300, 2157191, 2708050, 3363326,

+ /\* 0 \*/ 4194304, 5237765, 6557202, 8165337, 10153587,

+ /\* 5 \*/ 12820798, 15790321, 19976592, 24970740, 31350126,

+ /\* 10 \*/ 39045157, 49367440, 61356676, 76695844, 95443717,

+ /\* 15 \*/ 119304647, 148102320, 186737708, 238609294, 286331153,

+};

+

+static void activate\_task(struct rq \*rq, struct task\_struct \*p, int wakeup);

+

+/\*

+ \* runqueue iterator, to support SMP load-balancing between different

+ \* scheduling classes, without having to expose their internal data

+ \* structures to the load-balancing proper:

+ \*/

+struct rq\_iterator {

+ void \*arg;

+ struct task\_struct \*(\*start)(void \*);

+ struct task\_struct \*(\*next)(void \*);

+};

+

+#ifdef CONFIG\_SMP

+static unsigned long

+balance\_tasks(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ unsigned long max\_load\_move, struct sched\_domain \*sd,

+ enum cpu\_idle\_type idle, int \*all\_pinned,

+ int \*this\_best\_prio, struct rq\_iterator \*iterator);

+

+static int

+iter\_move\_one\_task(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ struct rq\_iterator \*iterator);

+#endif

+

+#ifdef CONFIG\_CGROUP\_CPUACCT

+static void cpuacct\_charge(struct task\_struct \*tsk, u64 cputime);

+#else

+static inline void cpuacct\_charge(struct task\_struct \*tsk, u64 cputime) {}

+#endif

+

+#include "sched\_stats.h"

+#include "sched\_idletask.c"

+#include "sched\_fair.c"

+#include "sched\_rt.c"

+#ifdef CONFIG\_SCHED\_DEBUG

+# include "sched\_debug.c"

+#endif

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+#include "sched\_casio.c"

+#endif

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ #define sched\_class\_highest (&casio\_sched\_class)

+#else

+ #define sched\_class\_highest (&rt\_sched\_class)

+#endif

+

+

+/\*

+ \* Update delta\_exec, delta\_fair fields for rq.

+ \*

+ \* delta\_fair clock advances at a rate inversely proportional to

+ \* total load (rq->load.weight) on the runqueue, while

+ \* delta\_exec advances at the same rate as wall-clock (provided

+ \* cpu is not idle).

+ \*

+ \* delta\_exec / delta\_fair is a measure of the (smoothened) load on this

+ \* runqueue over any given interval. This (smoothened) load is used

+ \* during load balance.

+ \*

+ \* This function is called /before/ updating rq->load

+ \* and when switching tasks.

+ \*/

+static inline void inc\_load(struct rq \*rq, const struct task\_struct \*p)

+{

+ update\_load\_add(&rq->load, p->se.load.weight);

+}

+

+static inline void dec\_load(struct rq \*rq, const struct task\_struct \*p)

+{

+ update\_load\_sub(&rq->load, p->se.load.weight);

+}

+

+static void inc\_nr\_running(struct task\_struct \*p, struct rq \*rq)

+{

+ rq->nr\_running++;

+ inc\_load(rq, p);

+}

+

+static void dec\_nr\_running(struct task\_struct \*p, struct rq \*rq)

+{

+ rq->nr\_running--;

+ dec\_load(rq, p);

+}

+

+static void set\_load\_weight(struct task\_struct \*p)

+{

+ if (task\_has\_rt\_policy(p)) {

+ p->se.load.weight = prio\_to\_weight[0] \* 2;

+ p->se.load.inv\_weight = prio\_to\_wmult[0] >> 1;

+ return;

+ }

+

+ /\*

+ \* SCHED\_IDLE tasks get minimal weight:

+ \*/

+ if (p->policy == SCHED\_IDLE) {

+ p->se.load.weight = WEIGHT\_IDLEPRIO;

+ p->se.load.inv\_weight = WMULT\_IDLEPRIO;

+ return;

+ }

+

+ p->se.load.weight = prio\_to\_weight[p->static\_prio - MAX\_RT\_PRIO];

+ p->se.load.inv\_weight = prio\_to\_wmult[p->static\_prio - MAX\_RT\_PRIO];

+}

+

+static void enqueue\_task(struct rq \*rq, struct task\_struct \*p, int wakeup)

+{

+ sched\_info\_queued(p);

+ p->sched\_class->enqueue\_task(rq, p, wakeup);

+ p->se.on\_rq = 1;

+}

+

+static void dequeue\_task(struct rq \*rq, struct task\_struct \*p, int sleep)

+{

+ p->sched\_class->dequeue\_task(rq, p, sleep);

+ p->se.on\_rq = 0;

+}

+

+/\*

+ \* \_\_normal\_prio - return the priority that is based on the static prio

+ \*/

+static inline int \_\_normal\_prio(struct task\_struct \*p)

+{

+ return p->static\_prio;

+}

+

+/\*

+ \* Calculate the expected normal priority: i.e. priority

+ \* without taking RT-inheritance into account. Might be

+ \* boosted by interactivity modifiers. Changes upon fork,

+ \* setprio syscalls, and whenever the interactivity

+ \* estimator recalculates.

+ \*/

+static inline int normal\_prio(struct task\_struct \*p)

+{

+ int prio;

+

+ if (task\_has\_rt\_policy(p))

+ prio = MAX\_RT\_PRIO-1 - p->rt\_priority;

+ else

+ prio = \_\_normal\_prio(p);

+ return prio;

+}

+

+/\*

+ \* Calculate the current priority, i.e. the priority

+ \* taken into account by the scheduler. This value might

+ \* be boosted by RT tasks, or might be boosted by

+ \* interactivity modifiers. Will be RT if the task got

+ \* RT-boosted. If not then it returns p->normal\_prio.

+ \*/

+static int effective\_prio(struct task\_struct \*p)

+{

+ p->normal\_prio = normal\_prio(p);

+ /\*

+ \* If we are RT tasks or we were boosted to RT priority,

+ \* keep the priority unchanged. Otherwise, update priority

+ \* to the normal priority:

+ \*/

+ if (!rt\_prio(p->prio))

+ return p->normal\_prio;

+ return p->prio;

+}

+

+/\*

+ \* activate\_task - move a task to the runqueue.

+ \*/

+static void activate\_task(struct rq \*rq, struct task\_struct \*p, int wakeup)

+{

+ if (p->state == TASK\_UNINTERRUPTIBLE)

+ rq->nr\_uninterruptible--;

+

+ enqueue\_task(rq, p, wakeup);

+ inc\_nr\_running(p, rq);

+}

+

+/\*

+ \* deactivate\_task - remove a task from the runqueue.

+ \*/

+static void deactivate\_task(struct rq \*rq, struct task\_struct \*p, int sleep)

+{

+ if (p->state == TASK\_UNINTERRUPTIBLE)

+ rq->nr\_uninterruptible++;

+

+ dequeue\_task(rq, p, sleep);

+ dec\_nr\_running(p, rq);

+}

+

+/\*\*

+ \* task\_curr - is this task currently executing on a CPU?

+ \* @p: the task in question.

+ \*/

+inline int task\_curr(const struct task\_struct \*p)

+{

+ return cpu\_curr(task\_cpu(p)) == p;

+}

+

+/\* Used instead of source\_load when we know the type == 0 \*/

+unsigned long weighted\_cpuload(const int cpu)

+{

+ return cpu\_rq(cpu)->load.weight;

+}

+

+static inline void \_\_set\_task\_cpu(struct task\_struct \*p, unsigned int cpu)

+{

+ set\_task\_cfs\_rq(p, cpu);

+#ifdef CONFIG\_SMP

+ /\*

+ \* After ->cpu is set up to a new value, task\_rq\_lock(p, ...) can be

+ \* successfuly executed on another CPU. We must ensure that updates of

+ \* per-task data have been completed by this moment.

+ \*/

+ smp\_wmb();

+ task\_thread\_info(p)->cpu = cpu;

+#endif

+}

+

+#ifdef CONFIG\_SMP

+

+/\*

+ \* Is this task likely cache-hot:

+ \*/

+static inline int

+task\_hot(struct task\_struct \*p, u64 now, struct sched\_domain \*sd)

+{

+ s64 delta;

+

+ if (p->sched\_class != &fair\_sched\_class)

+ return 0;

+

+ if (sysctl\_sched\_migration\_cost == -1)

+ return 1;

+ if (sysctl\_sched\_migration\_cost == 0)

+ return 0;

+

+ delta = now - p->se.exec\_start;

+

+ return delta < (s64)sysctl\_sched\_migration\_cost;

+}

+

+

+void set\_task\_cpu(struct task\_struct \*p, unsigned int new\_cpu)

+{

+ int old\_cpu = task\_cpu(p);

+ struct rq \*old\_rq = cpu\_rq(old\_cpu), \*new\_rq = cpu\_rq(new\_cpu);

+ struct cfs\_rq \*old\_cfsrq = task\_cfs\_rq(p),

+ \*new\_cfsrq = cpu\_cfs\_rq(old\_cfsrq, new\_cpu);

+ u64 clock\_offset;

+

+ clock\_offset = old\_rq->clock - new\_rq->clock;

+

+#ifdef CONFIG\_SCHEDSTATS

+ if (p->se.wait\_start)

+ p->se.wait\_start -= clock\_offset;

+ if (p->se.sleep\_start)

+ p->se.sleep\_start -= clock\_offset;

+ if (p->se.block\_start)

+ p->se.block\_start -= clock\_offset;

+ if (old\_cpu != new\_cpu) {

+ schedstat\_inc(p, se.nr\_migrations);

+ if (task\_hot(p, old\_rq->clock, NULL))

+ schedstat\_inc(p, se.nr\_forced2\_migrations);

+ }

+#endif

+ p->se.vruntime -= old\_cfsrq->min\_vruntime -

+ new\_cfsrq->min\_vruntime;

+

+ \_\_set\_task\_cpu(p, new\_cpu);

+}

+

+struct migration\_req {

+ struct list\_head list;

+

+ struct task\_struct \*task;

+ int dest\_cpu;

+

+ struct completion done;

+};

+

+/\*

+ \* The task's runqueue lock must be held.

+ \* Returns true if you have to wait for migration thread.

+ \*/

+static int

+migrate\_task(struct task\_struct \*p, int dest\_cpu, struct migration\_req \*req)

+{

+ struct rq \*rq = task\_rq(p);

+

+ /\*

+ \* If the task is not on a runqueue (and not running), then

+ \* it is sufficient to simply update the task's cpu field.

+ \*/

+ if (!p->se.on\_rq && !task\_running(rq, p)) {

+ set\_task\_cpu(p, dest\_cpu);

+ return 0;

+ }

+

+ init\_completion(&req->done);

+ req->task = p;

+ req->dest\_cpu = dest\_cpu;

+ list\_add(&req->list, &rq->migration\_queue);

+

+ return 1;

+}

+

+/\*

+ \* wait\_task\_inactive - wait for a thread to unschedule.

+ \*

+ \* The caller must ensure that the task \*will\* unschedule sometime soon,

+ \* else this function might spin for a \*long\* time. This function can't

+ \* be called with interrupts off, or it may introduce deadlock with

+ \* smp\_call\_function() if an IPI is sent by the same process we are

+ \* waiting to become inactive.

+ \*/

+void wait\_task\_inactive(struct task\_struct \*p)

+{

+ unsigned long flags;

+ int running, on\_rq;

+ struct rq \*rq;

+

+ for (;;) {

+ /\*

+ \* We do the initial early heuristics without holding

+ \* any task-queue locks at all. We'll only try to get

+ \* the runqueue lock when things look like they will

+ \* work out!

+ \*/

+ rq = task\_rq(p);

+

+ /\*

+ \* If the task is actively running on another CPU

+ \* still, just relax and busy-wait without holding

+ \* any locks.

+ \*

+ \* NOTE! Since we don't hold any locks, it's not

+ \* even sure that "rq" stays as the right runqueue!

+ \* But we don't care, since "task\_running()" will

+ \* return false if the runqueue has changed and p

+ \* is actually now running somewhere else!

+ \*/

+ while (task\_running(rq, p))

+ cpu\_relax();

+

+ /\*

+ \* Ok, time to look more closely! We need the rq

+ \* lock now, to be \*sure\*. If we're wrong, we'll

+ \* just go back and repeat.

+ \*/

+ rq = task\_rq\_lock(p, &flags);

+ running = task\_running(rq, p);

+ on\_rq = p->se.on\_rq;

+ task\_rq\_unlock(rq, &flags);

+

+ /\*

+ \* Was it really running after all now that we

+ \* checked with the proper locks actually held?

+ \*

+ \* Oops. Go back and try again..

+ \*/

+ if (unlikely(running)) {

+ cpu\_relax();

+ continue;

+ }

+

+ /\*

+ \* It's not enough that it's not actively running,

+ \* it must be off the runqueue \_entirely\_, and not

+ \* preempted!

+ \*

+ \* So if it wa still runnable (but just not actively

+ \* running right now), it's preempted, and we should

+ \* yield - it could be a while.

+ \*/

+ if (unlikely(on\_rq)) {

+ schedule\_timeout\_uninterruptible(1);

+ continue;

+ }

+

+ /\*

+ \* Ahh, all good. It wasn't running, and it wasn't

+ \* runnable, which means that it will never become

+ \* running in the future either. We're all done!

+ \*/

+ break;

+ }

+}

+

+/\*\*\*

+ \* kick\_process - kick a running thread to enter/exit the kernel

+ \* @p: the to-be-kicked thread

+ \*

+ \* Cause a process which is running on another CPU to enter

+ \* kernel-mode, without any delay. (to get signals handled.)

+ \*

+ \* NOTE: this function doesnt have to take the runqueue lock,

+ \* because all it wants to ensure is that the remote task enters

+ \* the kernel. If the IPI races and the task has been migrated

+ \* to another CPU then no harm is done and the purpose has been

+ \* achieved as well.

+ \*/

+void kick\_process(struct task\_struct \*p)

+{

+ int cpu;

+

+ preempt\_disable();

+ cpu = task\_cpu(p);

+ if ((cpu != smp\_processor\_id()) && task\_curr(p))

+ smp\_send\_reschedule(cpu);

+ preempt\_enable();

+}

+

+/\*

+ \* Return a low guess at the load of a migration-source cpu weighted

+ \* according to the scheduling class and "nice" value.

+ \*

+ \* We want to under-estimate the load of migration sources, to

+ \* balance conservatively.

+ \*/

+static unsigned long source\_load(int cpu, int type)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long total = weighted\_cpuload(cpu);

+

+ if (type == 0)

+ return total;

+

+ return min(rq->cpu\_load[type-1], total);

+}

+

+/\*

+ \* Return a high guess at the load of a migration-target cpu weighted

+ \* according to the scheduling class and "nice" value.

+ \*/

+static unsigned long target\_load(int cpu, int type)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long total = weighted\_cpuload(cpu);

+

+ if (type == 0)

+ return total;

+

+ return max(rq->cpu\_load[type-1], total);

+}

+

+/\*

+ \* Return the average load per task on the cpu's run queue

+ \*/

+static inline unsigned long cpu\_avg\_load\_per\_task(int cpu)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long total = weighted\_cpuload(cpu);

+ unsigned long n = rq->nr\_running;

+

+ return n ? total / n : SCHED\_LOAD\_SCALE;

+}

+

+/\*

+ \* find\_idlest\_group finds and returns the least busy CPU group within the

+ \* domain.

+ \*/

+static struct sched\_group \*

+find\_idlest\_group(struct sched\_domain \*sd, struct task\_struct \*p, int this\_cpu)

+{

+ struct sched\_group \*idlest = NULL, \*this = NULL, \*group = sd->groups;

+ unsigned long min\_load = ULONG\_MAX, this\_load = 0;

+ int load\_idx = sd->forkexec\_idx;

+ int imbalance = 100 + (sd->imbalance\_pct-100)/2;

+

+ do {

+ unsigned long load, avg\_load;

+ int local\_group;

+ int i;

+

+ /\* Skip over this group if it has no CPUs allowed \*/

+ if (!cpus\_intersects(group->cpumask, p->cpus\_allowed))

+ continue;

+

+ local\_group = cpu\_isset(this\_cpu, group->cpumask);

+

+ /\* Tally up the load of all CPUs in the group \*/

+ avg\_load = 0;

+

+ for\_each\_cpu\_mask(i, group->cpumask) {

+ /\* Bias balancing toward cpus of our domain \*/

+ if (local\_group)

+ load = source\_load(i, load\_idx);

+ else

+ load = target\_load(i, load\_idx);

+

+ avg\_load += load;

+ }

+

+ /\* Adjust by relative CPU power of the group \*/

+ avg\_load = sg\_div\_cpu\_power(group,

+ avg\_load \* SCHED\_LOAD\_SCALE);

+

+ if (local\_group) {

+ this\_load = avg\_load;

+ this = group;

+ } else if (avg\_load < min\_load) {

+ min\_load = avg\_load;

+ idlest = group;

+ }

+ } while (group = group->next, group != sd->groups);

+

+ if (!idlest || 100\*this\_load < imbalance\*min\_load)

+ return NULL;

+ return idlest;

+}

+

+/\*

+ \* find\_idlest\_cpu - find the idlest cpu among the cpus in group.

+ \*/

+static int

+find\_idlest\_cpu(struct sched\_group \*group, struct task\_struct \*p, int this\_cpu)

+{

+ cpumask\_t tmp;

+ unsigned long load, min\_load = ULONG\_MAX;

+ int idlest = -1;

+ int i;

+

+ /\* Traverse only the allowed CPUs \*/

+ cpus\_and(tmp, group->cpumask, p->cpus\_allowed);

+

+ for\_each\_cpu\_mask(i, tmp) {

+ load = weighted\_cpuload(i);

+

+ if (load < min\_load || (load == min\_load && i == this\_cpu)) {

+ min\_load = load;

+ idlest = i;

+ }

+ }

+

+ return idlest;

+}

+

+/\*

+ \* sched\_balance\_self: balance the current task (running on cpu) in domains

+ \* that have the 'flag' flag set. In practice, this is SD\_BALANCE\_FORK and

+ \* SD\_BALANCE\_EXEC.

+ \*

+ \* Balance, ie. select the least loaded group.

+ \*

+ \* Returns the target CPU number, or the same CPU if no balancing is needed.

+ \*

+ \* preempt must be disabled.

+ \*/

+static int sched\_balance\_self(int cpu, int flag)

+{

+ struct task\_struct \*t = current;

+ struct sched\_domain \*tmp, \*sd = NULL;

+

+ for\_each\_domain(cpu, tmp) {

+ /\*

+ \* If power savings logic is enabled for a domain, stop there.

+ \*/

+ if (tmp->flags & SD\_POWERSAVINGS\_BALANCE)

+ break;

+ if (tmp->flags & flag)

+ sd = tmp;

+ }

+

+ while (sd) {

+ cpumask\_t span;

+ struct sched\_group \*group;

+ int new\_cpu, weight;

+

+ if (!(sd->flags & flag)) {

+ sd = sd->child;

+ continue;

+ }

+

+ span = sd->span;

+ group = find\_idlest\_group(sd, t, cpu);

+ if (!group) {

+ sd = sd->child;

+ continue;

+ }

+

+ new\_cpu = find\_idlest\_cpu(group, t, cpu);

+ if (new\_cpu == -1 || new\_cpu == cpu) {

+ /\* Now try balancing at a lower domain level of cpu \*/

+ sd = sd->child;

+ continue;

+ }

+

+ /\* Now try balancing at a lower domain level of new\_cpu \*/

+ cpu = new\_cpu;

+ sd = NULL;

+ weight = cpus\_weight(span);

+ for\_each\_domain(cpu, tmp) {

+ if (weight <= cpus\_weight(tmp->span))

+ break;

+ if (tmp->flags & flag)

+ sd = tmp;

+ }

+ /\* while loop will break here if sd == NULL \*/

+ }

+

+ return cpu;

+}

+

+#endif /\* CONFIG\_SMP \*/

+

+/\*

+ \* wake\_idle() will wake a task on an idle cpu if task->cpu is

+ \* not idle and an idle cpu is available. The span of cpus to

+ \* search starts with cpus closest then further out as needed,

+ \* so we always favor a closer, idle cpu.

+ \*

+ \* Returns the CPU we should wake onto.

+ \*/

+#if defined(ARCH\_HAS\_SCHED\_WAKE\_IDLE)

+static int wake\_idle(int cpu, struct task\_struct \*p)

+{

+ cpumask\_t tmp;

+ struct sched\_domain \*sd;

+ int i;

+

+ /\*

+ \* If it is idle, then it is the best cpu to run this task.

+ \*

+ \* This cpu is also the best, if it has more than one task already.

+ \* Siblings must be also busy(in most cases) as they didn't already

+ \* pickup the extra load from this cpu and hence we need not check

+ \* sibling runqueue info. This will avoid the checks and cache miss

+ \* penalities associated with that.

+ \*/

+ if (idle\_cpu(cpu) || cpu\_rq(cpu)->nr\_running > 1)

+ return cpu;

+

+ for\_each\_domain(cpu, sd) {

+ if (sd->flags & SD\_WAKE\_IDLE) {

+ cpus\_and(tmp, sd->span, p->cpus\_allowed);

+ for\_each\_cpu\_mask(i, tmp) {

+ if (idle\_cpu(i)) {

+ if (i != task\_cpu(p)) {

+ schedstat\_inc(p,

+ se.nr\_wakeups\_idle);

+ }

+ return i;

+ }

+ }

+ } else {

+ break;

+ }

+ }

+ return cpu;

+}

+#else

+static inline int wake\_idle(int cpu, struct task\_struct \*p)

+{

+ return cpu;

+}

+#endif

+

+/\*\*\*

+ \* try\_to\_wake\_up - wake up a thread

+ \* @p: the to-be-woken-up thread

+ \* @state: the mask of task states that can be woken

+ \* @sync: do a synchronous wakeup?

+ \*

+ \* Put it on the run-queue if it's not already there. The "current"

+ \* thread is always on the run-queue (except when the actual

+ \* re-schedule is in progress), and as such you're allowed to do

+ \* the simpler "current->state = TASK\_RUNNING" to mark yourself

+ \* runnable without the overhead of this.

+ \*

+ \* returns failure only if the task is already active.

+ \*/

+static int try\_to\_wake\_up(struct task\_struct \*p, unsigned int state, int sync)

+{

+ int cpu, orig\_cpu, this\_cpu, success = 0;

+ unsigned long flags;

+ long old\_state;

+ struct rq \*rq;

+#ifdef CONFIG\_SMP

+ struct sched\_domain \*sd, \*this\_sd = NULL;

+ unsigned long load, this\_load;

+ int new\_cpu;

+#endif

+

+ rq = task\_rq\_lock(p, &flags);

+ old\_state = p->state;

+ if (!(old\_state & state))

+ goto out;

+

+ if (p->se.on\_rq)

+ goto out\_running;

+

+ cpu = task\_cpu(p);

+ orig\_cpu = cpu;

+ this\_cpu = smp\_processor\_id();

+

+#ifdef CONFIG\_SMP

+ if (unlikely(task\_running(rq, p)))

+ goto out\_activate;

+

+ new\_cpu = cpu;

+

+ schedstat\_inc(rq, ttwu\_count);

+ if (cpu == this\_cpu) {

+ schedstat\_inc(rq, ttwu\_local);

+ goto out\_set\_cpu;

+ }

+

+ for\_each\_domain(this\_cpu, sd) {

+ if (cpu\_isset(cpu, sd->span)) {

+ schedstat\_inc(sd, ttwu\_wake\_remote);

+ this\_sd = sd;

+ break;

+ }

+ }

+

+ if (unlikely(!cpu\_isset(this\_cpu, p->cpus\_allowed)))

+ goto out\_set\_cpu;

+

+ /\*

+ \* Check for affine wakeup and passive balancing possibilities.

+ \*/

+ if (this\_sd) {

+ int idx = this\_sd->wake\_idx;

+ unsigned int imbalance;

+

+ imbalance = 100 + (this\_sd->imbalance\_pct - 100) / 2;

+

+ load = source\_load(cpu, idx);

+ this\_load = target\_load(this\_cpu, idx);

+

+ new\_cpu = this\_cpu; /\* Wake to this CPU if we can \*/

+

+ if (this\_sd->flags & SD\_WAKE\_AFFINE) {

+ unsigned long tl = this\_load;

+ unsigned long tl\_per\_task;

+

+ /\*

+ \* Attract cache-cold tasks on sync wakeups:

+ \*/

+ if (sync && !task\_hot(p, rq->clock, this\_sd))

+ goto out\_set\_cpu;

+

+ schedstat\_inc(p, se.nr\_wakeups\_affine\_attempts);

+ tl\_per\_task = cpu\_avg\_load\_per\_task(this\_cpu);

+

+ /\*

+ \* If sync wakeup then subtract the (maximum possible)

+ \* effect of the currently running task from the load

+ \* of the current CPU:

+ \*/

+ if (sync)

+ tl -= current->se.load.weight;

+

+ if ((tl <= load &&

+ tl + target\_load(cpu, idx) <= tl\_per\_task) ||

+ 100\*(tl + p->se.load.weight) <= imbalance\*load) {

+ /\*

+ \* This domain has SD\_WAKE\_AFFINE and

+ \* p is cache cold in this domain, and

+ \* there is no bad imbalance.

+ \*/

+ schedstat\_inc(this\_sd, ttwu\_move\_affine);

+ schedstat\_inc(p, se.nr\_wakeups\_affine);

+ goto out\_set\_cpu;

+ }

+ }

+

+ /\*

+ \* Start passive balancing when half the imbalance\_pct

+ \* limit is reached.

+ \*/

+ if (this\_sd->flags & SD\_WAKE\_BALANCE) {

+ if (imbalance\*this\_load <= 100\*load) {

+ schedstat\_inc(this\_sd, ttwu\_move\_balance);

+ schedstat\_inc(p, se.nr\_wakeups\_passive);

+ goto out\_set\_cpu;

+ }

+ }

+ }

+

+ new\_cpu = cpu; /\* Could not wake to this\_cpu. Wake to cpu instead \*/

+out\_set\_cpu:

+ new\_cpu = wake\_idle(new\_cpu, p);

+ if (new\_cpu != cpu) {

+ set\_task\_cpu(p, new\_cpu);

+ task\_rq\_unlock(rq, &flags);

+ /\* might preempt at this point \*/

+ rq = task\_rq\_lock(p, &flags);

+ old\_state = p->state;

+ if (!(old\_state & state))

+ goto out;

+ if (p->se.on\_rq)

+ goto out\_running;

+

+ this\_cpu = smp\_processor\_id();

+ cpu = task\_cpu(p);

+ }

+

+out\_activate:

+#endif /\* CONFIG\_SMP \*/

+ schedstat\_inc(p, se.nr\_wakeups);

+ if (sync)

+ schedstat\_inc(p, se.nr\_wakeups\_sync);

+ if (orig\_cpu != cpu)

+ schedstat\_inc(p, se.nr\_wakeups\_migrate);

+ if (cpu == this\_cpu)

+ schedstat\_inc(p, se.nr\_wakeups\_local);

+ else

+ schedstat\_inc(p, se.nr\_wakeups\_remote);

+ update\_rq\_clock(rq);

+ activate\_task(rq, p, 1);

+ check\_preempt\_curr(rq, p);

+ success = 1;

+

+out\_running:

+ p->state = TASK\_RUNNING;

+out:

+ task\_rq\_unlock(rq, &flags);

+

+ return success;

+}

+

+int fastcall wake\_up\_process(struct task\_struct \*p)

+{

+ return try\_to\_wake\_up(p, TASK\_STOPPED | TASK\_TRACED |

+ TASK\_INTERRUPTIBLE | TASK\_UNINTERRUPTIBLE, 0);

+}

+EXPORT\_SYMBOL(wake\_up\_process);

+

+int fastcall wake\_up\_state(struct task\_struct \*p, unsigned int state)

+{

+ return try\_to\_wake\_up(p, state, 0);

+}

+

+/\*

+ \* Perform scheduler related setup for a newly forked process p.

+ \* p is forked by current.

+ \*

+ \* \_\_sched\_fork() is basic setup used by init\_idle() too:

+ \*/

+static void \_\_sched\_fork(struct task\_struct \*p)

+{

+ p->se.exec\_start = 0;

+ p->se.sum\_exec\_runtime = 0;

+ p->se.prev\_sum\_exec\_runtime = 0;

+

+#ifdef CONFIG\_SCHEDSTATS

+ p->se.wait\_start = 0;

+ p->se.sum\_sleep\_runtime = 0;

+ p->se.sleep\_start = 0;

+ p->se.block\_start = 0;

+ p->se.sleep\_max = 0;

+ p->se.block\_max = 0;

+ p->se.exec\_max = 0;

+ p->se.slice\_max = 0;

+ p->se.wait\_max = 0;

+#endif

+

+ INIT\_LIST\_HEAD(&p->run\_list);

+ p->se.on\_rq = 0;

+

+#ifdef CONFIG\_PREEMPT\_NOTIFIERS

+ INIT\_HLIST\_HEAD(&p->preempt\_notifiers);

+#endif

+

+ /\*

+ \* We mark the process as running here, but have not actually

+ \* inserted it onto the runqueue yet. This guarantees that

+ \* nobody will actually run it, and a signal or other external

+ \* event cannot wake it up and insert it on the runqueue either.

+ \*/

+ p->state = TASK\_RUNNING;

+}

+

+/\*

+ \* fork()/clone()-time setup:

+ \*/

+void sched\_fork(struct task\_struct \*p, int clone\_flags)

+{

+ int cpu = get\_cpu();

+

+ \_\_sched\_fork(p);

+

+#ifdef CONFIG\_SMP

+ cpu = sched\_balance\_self(cpu, SD\_BALANCE\_FORK);

+#endif

+ set\_task\_cpu(p, cpu);

+

+ /\*

+ \* Make sure we do not leak PI boosting priority to the child:

+ \*/

+ p->prio = current->normal\_prio;

+ if (!rt\_prio(p->prio))

+ p->sched\_class = &fair\_sched\_class;

+

+#if defined(CONFIG\_SCHEDSTATS) || defined(CONFIG\_TASK\_DELAY\_ACCT)

+ if (likely(sched\_info\_on()))

+ memset(&p->sched\_info, 0, sizeof(p->sched\_info));

+#endif

+#if defined(CONFIG\_SMP) && defined(\_\_ARCH\_WANT\_UNLOCKED\_CTXSW)

+ p->oncpu = 0;

+#endif

+#ifdef CONFIG\_PREEMPT

+ /\* Want to start with kernel preemption disabled. \*/

+ task\_thread\_info(p)->preempt\_count = 1;

+#endif

+ put\_cpu();

+}

+

+/\*

+ \* wake\_up\_new\_task - wake up a newly created task for the first time.

+ \*

+ \* This function will do some initial scheduler statistics housekeeping

+ \* that must be done for every newly created context, then puts the task

+ \* on the runqueue and wakes it.

+ \*/

+void fastcall wake\_up\_new\_task(struct task\_struct \*p, unsigned long clone\_flags)

+{

+ unsigned long flags;

+ struct rq \*rq;

+

+ rq = task\_rq\_lock(p, &flags);

+ BUG\_ON(p->state != TASK\_RUNNING);

+ update\_rq\_clock(rq);

+

+ p->prio = effective\_prio(p);

+

+ if (!p->sched\_class->task\_new || !current->se.on\_rq) {

+ activate\_task(rq, p, 0);

+ } else {

+ /\*

+ \* Let the scheduling class do new task startup

+ \* management (if any):

+ \*/

+ p->sched\_class->task\_new(rq, p);

+ inc\_nr\_running(p, rq);

+ }

+ check\_preempt\_curr(rq, p);

+ task\_rq\_unlock(rq, &flags);

+}

+

+#ifdef CONFIG\_PREEMPT\_NOTIFIERS

+

+/\*\*

+ \* preempt\_notifier\_register - tell me when current is being being preempted & rescheduled

+ \* @notifier: notifier struct to register

+ \*/

+void preempt\_notifier\_register(struct preempt\_notifier \*notifier)

+{

+ hlist\_add\_head(&notifier->link, &current->preempt\_notifiers);

+}

+EXPORT\_SYMBOL\_GPL(preempt\_notifier\_register);

+

+/\*\*

+ \* preempt\_notifier\_unregister - no longer interested in preemption notifications

+ \* @notifier: notifier struct to unregister

+ \*

+ \* This is safe to call from within a preemption notifier.

+ \*/

+void preempt\_notifier\_unregister(struct preempt\_notifier \*notifier)

+{

+ hlist\_del(&notifier->link);

+}

+EXPORT\_SYMBOL\_GPL(preempt\_notifier\_unregister);

+

+static void fire\_sched\_in\_preempt\_notifiers(struct task\_struct \*curr)

+{

+ struct preempt\_notifier \*notifier;

+ struct hlist\_node \*node;

+

+ hlist\_for\_each\_entry(notifier, node, &curr->preempt\_notifiers, link)

+ notifier->ops->sched\_in(notifier, raw\_smp\_processor\_id());

+}

+

+static void

+fire\_sched\_out\_preempt\_notifiers(struct task\_struct \*curr,

+ struct task\_struct \*next)

+{

+ struct preempt\_notifier \*notifier;

+ struct hlist\_node \*node;

+

+ hlist\_for\_each\_entry(notifier, node, &curr->preempt\_notifiers, link)

+ notifier->ops->sched\_out(notifier, next);

+}

+

+#else

+

+static void fire\_sched\_in\_preempt\_notifiers(struct task\_struct \*curr)

+{

+}

+

+static void

+fire\_sched\_out\_preempt\_notifiers(struct task\_struct \*curr,

+ struct task\_struct \*next)

+{

+}

+

+#endif

+

+/\*\*

+ \* prepare\_task\_switch - prepare to switch tasks

+ \* @rq: the runqueue preparing to switch

+ \* @prev: the current task that is being switched out

+ \* @next: the task we are going to switch to.

+ \*

+ \* This is called with the rq lock held and interrupts off. It must

+ \* be paired with a subsequent finish\_task\_switch after the context

+ \* switch.

+ \*

+ \* prepare\_task\_switch sets up locking and calls architecture specific

+ \* hooks.

+ \*/

+static inline void

+prepare\_task\_switch(struct rq \*rq, struct task\_struct \*prev,

+ struct task\_struct \*next)

+{

+ fire\_sched\_out\_preempt\_notifiers(prev, next);

+ prepare\_lock\_switch(rq, next);

+ prepare\_arch\_switch(next);

+}

+

+/\*\*

+ \* finish\_task\_switch - clean up after a task-switch

+ \* @rq: runqueue associated with task-switch

+ \* @prev: the thread we just switched away from.

+ \*

+ \* finish\_task\_switch must be called after the context switch, paired

+ \* with a prepare\_task\_switch call before the context switch.

+ \* finish\_task\_switch will reconcile locking set up by prepare\_task\_switch,

+ \* and do any other architecture-specific cleanup actions.

+ \*

+ \* Note that we may have delayed dropping an mm in context\_switch(). If

+ \* so, we finish that here outside of the runqueue lock. (Doing it

+ \* with the lock held can cause deadlocks; see schedule() for

+ \* details.)

+ \*/

+static void finish\_task\_switch(struct rq \*rq, struct task\_struct \*prev)

+ \_\_releases(rq->lock)

+{

+ struct mm\_struct \*mm = rq->prev\_mm;

+ long prev\_state;

+

+ rq->prev\_mm = NULL;

+

+ /\*

+ \* A task struct has one reference for the use as "current".

+ \* If a task dies, then it sets TASK\_DEAD in tsk->state and calls

+ \* schedule one last time. The schedule call will never return, and

+ \* the scheduled task must drop that reference.

+ \* The test for TASK\_DEAD must occur while the runqueue locks are

+ \* still held, otherwise prev could be scheduled on another cpu, die

+ \* there before we look at prev->state, and then the reference would

+ \* be dropped twice.

+ \* Manfred Spraul <manfred@colorfullife.com>

+ \*/

+ prev\_state = prev->state;

+ finish\_arch\_switch(prev);

+ finish\_lock\_switch(rq, prev);

+ fire\_sched\_in\_preempt\_notifiers(current);

+ if (mm)

+ mmdrop(mm);

+ if (unlikely(prev\_state == TASK\_DEAD)) {

+ /\*

+ \* Remove function-return probe instances associated with this

+ \* task and put them back on the free list.

+ \*/

+ kprobe\_flush\_task(prev);

+ put\_task\_struct(prev);

+ }

+}

+

+/\*\*

+ \* schedule\_tail - first thing a freshly forked thread must call.

+ \* @prev: the thread we just switched away from.

+ \*/

+asmlinkage void schedule\_tail(struct task\_struct \*prev)

+ \_\_releases(rq->lock)

+{

+ struct rq \*rq = this\_rq();

+

+ finish\_task\_switch(rq, prev);

+#ifdef \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

+ /\* In this case, finish\_task\_switch does not reenable preemption \*/

+ preempt\_enable();

+#endif

+ if (current->set\_child\_tid)

+ put\_user(task\_pid\_vnr(current), current->set\_child\_tid);

+}

+

+/\*

+ \* context\_switch - switch to the new MM and the new

+ \* thread's register state.

+ \*/

+static inline void

+context\_switch(struct rq \*rq, struct task\_struct \*prev,

+ struct task\_struct \*next)

+{

+ struct mm\_struct \*mm, \*oldmm;

+

+ prepare\_task\_switch(rq, prev, next);

+ mm = next->mm;

+ oldmm = prev->active\_mm;

+ /\*

+ \* For paravirt, this is coupled with an exit in switch\_to to

+ \* combine the page table reload and the switch backend into

+ \* one hypercall.

+ \*/

+ arch\_enter\_lazy\_cpu\_mode();

+

+ if (unlikely(!mm)) {

+ next->active\_mm = oldmm;

+ atomic\_inc(&oldmm->mm\_count);

+ enter\_lazy\_tlb(oldmm, next);

+ } else

+ switch\_mm(oldmm, mm, next);

+

+ if (unlikely(!prev->mm)) {

+ prev->active\_mm = NULL;

+ rq->prev\_mm = oldmm;

+ }

+ /\*

+ \* Since the runqueue lock will be released by the next

+ \* task (which is an invalid locking op but in the case

+ \* of the scheduler it's an obvious special-case), so we

+ \* do an early lockdep release here:

+ \*/

+#ifndef \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

+ spin\_release(&rq->lock.dep\_map, 1, \_THIS\_IP\_);

+#endif

+

+ /\* Here we just switch the register state and the stack. \*/

+ switch\_to(prev, next, prev);

+

+ barrier();

+ /\*

+ \* this\_rq must be evaluated again because prev may have moved

+ \* CPUs since it called schedule(), thus the 'rq' on its stack

+ \* frame will be invalid.

+ \*/

+ finish\_task\_switch(this\_rq(), prev);

+}

+

+/\*

+ \* nr\_running, nr\_uninterruptible and nr\_context\_switches:

+ \*

+ \* externally visible scheduler statistics: current number of runnable

+ \* threads, current number of uninterruptible-sleeping threads, total

+ \* number of context switches performed since bootup.

+ \*/

+unsigned long nr\_running(void)

+{

+ unsigned long i, sum = 0;

+

+ for\_each\_online\_cpu(i)

+ sum += cpu\_rq(i)->nr\_running;

+

+ return sum;

+}

+

+unsigned long nr\_uninterruptible(void)

+{

+ unsigned long i, sum = 0;

+

+ for\_each\_possible\_cpu(i)

+ sum += cpu\_rq(i)->nr\_uninterruptible;

+

+ /\*

+ \* Since we read the counters lockless, it might be slightly

+ \* inaccurate. Do not allow it to go below zero though:

+ \*/

+ if (unlikely((long)sum < 0))

+ sum = 0;

+

+ return sum;

+}

+

+unsigned long long nr\_context\_switches(void)

+{

+ int i;

+ unsigned long long sum = 0;

+

+ for\_each\_possible\_cpu(i)

+ sum += cpu\_rq(i)->nr\_switches;

+

+ return sum;

+}

+

+unsigned long nr\_iowait(void)

+{

+ unsigned long i, sum = 0;

+

+ for\_each\_possible\_cpu(i)

+ sum += atomic\_read(&cpu\_rq(i)->nr\_iowait);

+

+ return sum;

+}

+

+unsigned long nr\_active(void)

+{

+ unsigned long i, running = 0, uninterruptible = 0;

+

+ for\_each\_online\_cpu(i) {

+ running += cpu\_rq(i)->nr\_running;

+ uninterruptible += cpu\_rq(i)->nr\_uninterruptible;

+ }

+

+ if (unlikely((long)uninterruptible < 0))

+ uninterruptible = 0;

+

+ return running + uninterruptible;

+}

+

+/\*

+ \* Update rq->cpu\_load[] statistics. This function is usually called every

+ \* scheduler tick (TICK\_NSEC).

+ \*/

+static void update\_cpu\_load(struct rq \*this\_rq)

+{

+ unsigned long this\_load = this\_rq->load.weight;

+ int i, scale;

+

+ this\_rq->nr\_load\_updates++;

+

+ /\* Update our load: \*/

+ for (i = 0, scale = 1; i < CPU\_LOAD\_IDX\_MAX; i++, scale += scale) {

+ unsigned long old\_load, new\_load;

+

+ /\* scale is effectively 1 << i now, and >> i divides by scale \*/

+

+ old\_load = this\_rq->cpu\_load[i];

+ new\_load = this\_load;

+ /\*

+ \* Round up the averaging division if load is increasing. This

+ \* prevents us from getting stuck on 9 if the load is 10, for

+ \* example.

+ \*/

+ if (new\_load > old\_load)

+ new\_load += scale-1;

+ this\_rq->cpu\_load[i] = (old\_load\*(scale-1) + new\_load) >> i;

+ }

+}

+

+#ifdef CONFIG\_SMP

+

+/\*

+ \* double\_rq\_lock - safely lock two runqueues

+ \*

+ \* Note this does not disable interrupts like task\_rq\_lock,

+ \* you need to do so manually before calling.

+ \*/

+static void double\_rq\_lock(struct rq \*rq1, struct rq \*rq2)

+ \_\_acquires(rq1->lock)

+ \_\_acquires(rq2->lock)

+{

+ BUG\_ON(!irqs\_disabled());

+ if (rq1 == rq2) {

+ spin\_lock(&rq1->lock);

+ \_\_acquire(rq2->lock); /\* Fake it out ;) \*/

+ } else {

+ if (rq1 < rq2) {

+ spin\_lock(&rq1->lock);

+ spin\_lock(&rq2->lock);

+ } else {

+ spin\_lock(&rq2->lock);

+ spin\_lock(&rq1->lock);

+ }

+ }

+ update\_rq\_clock(rq1);

+ update\_rq\_clock(rq2);

+}

+

+/\*

+ \* double\_rq\_unlock - safely unlock two runqueues

+ \*

+ \* Note this does not restore interrupts like task\_rq\_unlock,

+ \* you need to do so manually after calling.

+ \*/

+static void double\_rq\_unlock(struct rq \*rq1, struct rq \*rq2)

+ \_\_releases(rq1->lock)

+ \_\_releases(rq2->lock)

+{

+ spin\_unlock(&rq1->lock);

+ if (rq1 != rq2)

+ spin\_unlock(&rq2->lock);

+ else

+ \_\_release(rq2->lock);

+}

+

+/\*

+ \* double\_lock\_balance - lock the busiest runqueue, this\_rq is locked already.

+ \*/

+static void double\_lock\_balance(struct rq \*this\_rq, struct rq \*busiest)

+ \_\_releases(this\_rq->lock)

+ \_\_acquires(busiest->lock)

+ \_\_acquires(this\_rq->lock)

+{

+ if (unlikely(!irqs\_disabled())) {

+ /\* printk() doesn't work good under rq->lock \*/

+ spin\_unlock(&this\_rq->lock);

+ BUG\_ON(1);

+ }

+ if (unlikely(!spin\_trylock(&busiest->lock))) {

+ if (busiest < this\_rq) {

+ spin\_unlock(&this\_rq->lock);

+ spin\_lock(&busiest->lock);

+ spin\_lock(&this\_rq->lock);

+ } else

+ spin\_lock(&busiest->lock);

+ }

+}

+

+/\*

+ \* If dest\_cpu is allowed for this process, migrate the task to it.

+ \* This is accomplished by forcing the cpu\_allowed mask to only

+ \* allow dest\_cpu, which will force the cpu onto dest\_cpu. Then

+ \* the cpu\_allowed mask is restored.

+ \*/

+static void sched\_migrate\_task(struct task\_struct \*p, int dest\_cpu)

+{

+ struct migration\_req req;

+ unsigned long flags;

+ struct rq \*rq;

+

+ rq = task\_rq\_lock(p, &flags);

+ if (!cpu\_isset(dest\_cpu, p->cpus\_allowed)

+ || unlikely(cpu\_is\_offline(dest\_cpu)))

+ goto out;

+

+ /\* force the process onto the specified CPU \*/

+ if (migrate\_task(p, dest\_cpu, &req)) {

+ /\* Need to wait for migration thread (might exit: take ref). \*/

+ struct task\_struct \*mt = rq->migration\_thread;

+

+ get\_task\_struct(mt);

+ task\_rq\_unlock(rq, &flags);

+ wake\_up\_process(mt);

+ put\_task\_struct(mt);

+ wait\_for\_completion(&req.done);

+

+ return;

+ }

+out:

+ task\_rq\_unlock(rq, &flags);

+}

+

+/\*

+ \* sched\_exec - execve() is a valuable balancing opportunity, because at

+ \* this point the task has the smallest effective memory and cache footprint.

+ \*/

+void sched\_exec(void)

+{

+ int new\_cpu, this\_cpu = get\_cpu();

+ new\_cpu = sched\_balance\_self(this\_cpu, SD\_BALANCE\_EXEC);

+ put\_cpu();

+ if (new\_cpu != this\_cpu)

+ sched\_migrate\_task(current, new\_cpu);

+}

+

+/\*

+ \* pull\_task - move a task from a remote runqueue to the local runqueue.

+ \* Both runqueues must be locked.

+ \*/

+static void pull\_task(struct rq \*src\_rq, struct task\_struct \*p,

+ struct rq \*this\_rq, int this\_cpu)

+{

+ deactivate\_task(src\_rq, p, 0);

+ set\_task\_cpu(p, this\_cpu);

+ activate\_task(this\_rq, p, 0);

+ /\*

+ \* Note that idle threads have a prio of MAX\_PRIO, for this test

+ \* to be always true for them.

+ \*/

+ check\_preempt\_curr(this\_rq, p);

+}

+

+/\*

+ \* can\_migrate\_task - may task p from runqueue rq be migrated to this\_cpu?

+ \*/

+static

+int can\_migrate\_task(struct task\_struct \*p, struct rq \*rq, int this\_cpu,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*all\_pinned)

+{

+ /\*

+ \* We do not migrate tasks that are:

+ \* 1) running (obviously), or

+ \* 2) cannot be migrated to this CPU due to cpus\_allowed, or

+ \* 3) are cache-hot on their current CPU.

+ \*/

+ if (!cpu\_isset(this\_cpu, p->cpus\_allowed)) {

+ schedstat\_inc(p, se.nr\_failed\_migrations\_affine);

+ return 0;

+ }

+ \*all\_pinned = 0;

+

+ if (task\_running(rq, p)) {

+ schedstat\_inc(p, se.nr\_failed\_migrations\_running);

+ return 0;

+ }

+

+ /\*

+ \* Aggressive migration if:

+ \* 1) task is cache cold, or

+ \* 2) too many balance attempts have failed.

+ \*/

+

+ if (!task\_hot(p, rq->clock, sd) ||

+ sd->nr\_balance\_failed > sd->cache\_nice\_tries) {

+#ifdef CONFIG\_SCHEDSTATS

+ if (task\_hot(p, rq->clock, sd)) {

+ schedstat\_inc(sd, lb\_hot\_gained[idle]);

+ schedstat\_inc(p, se.nr\_forced\_migrations);

+ }

+#endif

+ return 1;

+ }

+

+ if (task\_hot(p, rq->clock, sd)) {

+ schedstat\_inc(p, se.nr\_failed\_migrations\_hot);

+ return 0;

+ }

+ return 1;

+}

+

+static unsigned long

+balance\_tasks(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ unsigned long max\_load\_move, struct sched\_domain \*sd,

+ enum cpu\_idle\_type idle, int \*all\_pinned,

+ int \*this\_best\_prio, struct rq\_iterator \*iterator)

+{

+ int loops = 0, pulled = 0, pinned = 0, skip\_for\_load;

+ struct task\_struct \*p;

+ long rem\_load\_move = max\_load\_move;

+

+ if (max\_load\_move == 0)

+ goto out;

+

+ pinned = 1;

+

+ /\*

+ \* Start the load-balancing iterator:

+ \*/

+ p = iterator->start(iterator->arg);

+next:

+ if (!p || loops++ > sysctl\_sched\_nr\_migrate)

+ goto out;

+ /\*

+ \* To help distribute high priority tasks across CPUs we don't

+ \* skip a task if it will be the highest priority task (i.e. smallest

+ \* prio value) on its new queue regardless of its load weight

+ \*/

+ skip\_for\_load = (p->se.load.weight >> 1) > rem\_load\_move +

+ SCHED\_LOAD\_SCALE\_FUZZ;

+ if ((skip\_for\_load && p->prio >= \*this\_best\_prio) ||

+ !can\_migrate\_task(p, busiest, this\_cpu, sd, idle, &pinned)) {

+ p = iterator->next(iterator->arg);

+ goto next;

+ }

+

+ pull\_task(busiest, p, this\_rq, this\_cpu);

+ pulled++;

+ rem\_load\_move -= p->se.load.weight;

+

+ /\*

+ \* We only want to steal up to the prescribed amount of weighted load.

+ \*/

+ if (rem\_load\_move > 0) {

+ if (p->prio < \*this\_best\_prio)

+ \*this\_best\_prio = p->prio;

+ p = iterator->next(iterator->arg);

+ goto next;

+ }

+out:

+ /\*

+ \* Right now, this is one of only two places pull\_task() is called,

+ \* so we can safely collect pull\_task() stats here rather than

+ \* inside pull\_task().

+ \*/

+ schedstat\_add(sd, lb\_gained[idle], pulled);

+

+ if (all\_pinned)

+ \*all\_pinned = pinned;

+

+ return max\_load\_move - rem\_load\_move;

+}

+

+/\*

+ \* move\_tasks tries to move up to max\_load\_move weighted load from busiest to

+ \* this\_rq, as part of a balancing operation within domain "sd".

+ \* Returns 1 if successful and 0 otherwise.

+ \*

+ \* Called with both runqueues locked.

+ \*/

+static int move\_tasks(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ unsigned long max\_load\_move,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*all\_pinned)

+{

+ const struct sched\_class \*class = sched\_class\_highest;

+ unsigned long total\_load\_moved = 0;

+ int this\_best\_prio = this\_rq->curr->prio;

+

+ do {

+ total\_load\_moved +=

+ class->load\_balance(this\_rq, this\_cpu, busiest,

+ max\_load\_move - total\_load\_moved,

+ sd, idle, all\_pinned, &this\_best\_prio);

+ class = class->next;

+ } while (class && max\_load\_move > total\_load\_moved);

+

+ return total\_load\_moved > 0;

+}

+

+static int

+iter\_move\_one\_task(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ struct rq\_iterator \*iterator)

+{

+ struct task\_struct \*p = iterator->start(iterator->arg);

+ int pinned = 0;

+

+ while (p) {

+ if (can\_migrate\_task(p, busiest, this\_cpu, sd, idle, &pinned)) {

+ pull\_task(busiest, p, this\_rq, this\_cpu);

+ /\*

+ \* Right now, this is only the second place pull\_task()

+ \* is called, so we can safely collect pull\_task()

+ \* stats here rather than inside pull\_task().

+ \*/

+ schedstat\_inc(sd, lb\_gained[idle]);

+

+ return 1;

+ }

+ p = iterator->next(iterator->arg);

+ }

+

+ return 0;

+}

+

+/\*

+ \* move\_one\_task tries to move exactly one task from busiest to this\_rq, as

+ \* part of active balancing operations within "domain".

+ \* Returns 1 if successful and 0 otherwise.

+ \*

+ \* Called with both runqueues locked.

+ \*/

+static int move\_one\_task(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle)

+{

+ const struct sched\_class \*class;

+

+ for (class = sched\_class\_highest; class; class = class->next)

+ if (class->move\_one\_task(this\_rq, this\_cpu, busiest, sd, idle))

+ return 1;

+

+ return 0;

+}

+

+/\*

+ \* find\_busiest\_group finds and returns the busiest CPU group within the

+ \* domain. It calculates and returns the amount of weighted load which

+ \* should be moved to restore balance via the imbalance parameter.

+ \*/

+static struct sched\_group \*

+find\_busiest\_group(struct sched\_domain \*sd, int this\_cpu,

+ unsigned long \*imbalance, enum cpu\_idle\_type idle,

+ int \*sd\_idle, cpumask\_t \*cpus, int \*balance)

+{

+ struct sched\_group \*busiest = NULL, \*this = NULL, \*group = sd->groups;

+ unsigned long max\_load, avg\_load, total\_load, this\_load, total\_pwr;

+ unsigned long max\_pull;

+ unsigned long busiest\_load\_per\_task, busiest\_nr\_running;

+ unsigned long this\_load\_per\_task, this\_nr\_running;

+ int load\_idx, group\_imb = 0;

+#if defined(CONFIG\_SCHED\_MC) || defined(CONFIG\_SCHED\_SMT)

+ int power\_savings\_balance = 1;

+ unsigned long leader\_nr\_running = 0, min\_load\_per\_task = 0;

+ unsigned long min\_nr\_running = ULONG\_MAX;

+ struct sched\_group \*group\_min = NULL, \*group\_leader = NULL;

+#endif

+

+ max\_load = this\_load = total\_load = total\_pwr = 0;

+ busiest\_load\_per\_task = busiest\_nr\_running = 0;

+ this\_load\_per\_task = this\_nr\_running = 0;

+ if (idle == CPU\_NOT\_IDLE)

+ load\_idx = sd->busy\_idx;

+ else if (idle == CPU\_NEWLY\_IDLE)

+ load\_idx = sd->newidle\_idx;

+ else

+ load\_idx = sd->idle\_idx;

+

+ do {

+ unsigned long load, group\_capacity, max\_cpu\_load, min\_cpu\_load;

+ int local\_group;

+ int i;

+ int \_\_group\_imb = 0;

+ unsigned int balance\_cpu = -1, first\_idle\_cpu = 0;

+ unsigned long sum\_nr\_running, sum\_weighted\_load;

+

+ local\_group = cpu\_isset(this\_cpu, group->cpumask);

+

+ if (local\_group)

+ balance\_cpu = first\_cpu(group->cpumask);

+

+ /\* Tally up the load of all CPUs in the group \*/

+ sum\_weighted\_load = sum\_nr\_running = avg\_load = 0;

+ max\_cpu\_load = 0;

+ min\_cpu\_load = ~0UL;

+

+ for\_each\_cpu\_mask(i, group->cpumask) {

+ struct rq \*rq;

+

+ if (!cpu\_isset(i, \*cpus))

+ continue;

+

+ rq = cpu\_rq(i);

+

+ if (\*sd\_idle && rq->nr\_running)

+ \*sd\_idle = 0;

+

+ /\* Bias balancing toward cpus of our domain \*/

+ if (local\_group) {

+ if (idle\_cpu(i) && !first\_idle\_cpu) {

+ first\_idle\_cpu = 1;

+ balance\_cpu = i;

+ }

+

+ load = target\_load(i, load\_idx);

+ } else {

+ load = source\_load(i, load\_idx);

+ if (load > max\_cpu\_load)

+ max\_cpu\_load = load;

+ if (min\_cpu\_load > load)

+ min\_cpu\_load = load;

+ }

+

+ avg\_load += load;

+ sum\_nr\_running += rq->nr\_running;

+ sum\_weighted\_load += weighted\_cpuload(i);

+ }

+

+ /\*

+ \* First idle cpu or the first cpu(busiest) in this sched group

+ \* is eligible for doing load balancing at this and above

+ \* domains. In the newly idle case, we will allow all the cpu's

+ \* to do the newly idle load balance.

+ \*/

+ if (idle != CPU\_NEWLY\_IDLE && local\_group &&

+ balance\_cpu != this\_cpu && balance) {

+ \*balance = 0;

+ goto ret;

+ }

+

+ total\_load += avg\_load;

+ total\_pwr += group->\_\_cpu\_power;

+

+ /\* Adjust by relative CPU power of the group \*/

+ avg\_load = sg\_div\_cpu\_power(group,

+ avg\_load \* SCHED\_LOAD\_SCALE);

+

+ if ((max\_cpu\_load - min\_cpu\_load) > SCHED\_LOAD\_SCALE)

+ \_\_group\_imb = 1;

+

+ group\_capacity = group->\_\_cpu\_power / SCHED\_LOAD\_SCALE;

+

+ if (local\_group) {

+ this\_load = avg\_load;

+ this = group;

+ this\_nr\_running = sum\_nr\_running;

+ this\_load\_per\_task = sum\_weighted\_load;

+ } else if (avg\_load > max\_load &&

+ (sum\_nr\_running > group\_capacity || \_\_group\_imb)) {

+ max\_load = avg\_load;

+ busiest = group;

+ busiest\_nr\_running = sum\_nr\_running;

+ busiest\_load\_per\_task = sum\_weighted\_load;

+ group\_imb = \_\_group\_imb;

+ }

+

+#if defined(CONFIG\_SCHED\_MC) || defined(CONFIG\_SCHED\_SMT)

+ /\*

+ \* Busy processors will not participate in power savings

+ \* balance.

+ \*/

+ if (idle == CPU\_NOT\_IDLE ||

+ !(sd->flags & SD\_POWERSAVINGS\_BALANCE))

+ goto group\_next;

+

+ /\*

+ \* If the local group is idle or completely loaded

+ \* no need to do power savings balance at this domain

+ \*/

+ if (local\_group && (this\_nr\_running >= group\_capacity ||

+ !this\_nr\_running))

+ power\_savings\_balance = 0;

+

+ /\*

+ \* If a group is already running at full capacity or idle,

+ \* don't include that group in power savings calculations

+ \*/

+ if (!power\_savings\_balance || sum\_nr\_running >= group\_capacity

+ || !sum\_nr\_running)

+ goto group\_next;

+

+ /\*

+ \* Calculate the group which has the least non-idle load.

+ \* This is the group from where we need to pick up the load

+ \* for saving power

+ \*/

+ if ((sum\_nr\_running < min\_nr\_running) ||

+ (sum\_nr\_running == min\_nr\_running &&

+ first\_cpu(group->cpumask) <

+ first\_cpu(group\_min->cpumask))) {

+ group\_min = group;

+ min\_nr\_running = sum\_nr\_running;

+ min\_load\_per\_task = sum\_weighted\_load /

+ sum\_nr\_running;

+ }

+

+ /\*

+ \* Calculate the group which is almost near its

+ \* capacity but still has some space to pick up some load

+ \* from other group and save more power

+ \*/

+ if (sum\_nr\_running <= group\_capacity - 1) {

+ if (sum\_nr\_running > leader\_nr\_running ||

+ (sum\_nr\_running == leader\_nr\_running &&

+ first\_cpu(group->cpumask) >

+ first\_cpu(group\_leader->cpumask))) {

+ group\_leader = group;

+ leader\_nr\_running = sum\_nr\_running;

+ }

+ }

+group\_next:

+#endif

+ group = group->next;

+ } while (group != sd->groups);

+

+ if (!busiest || this\_load >= max\_load || busiest\_nr\_running == 0)

+ goto out\_balanced;

+

+ avg\_load = (SCHED\_LOAD\_SCALE \* total\_load) / total\_pwr;

+

+ if (this\_load >= avg\_load ||

+ 100\*max\_load <= sd->imbalance\_pct\*this\_load)

+ goto out\_balanced;

+

+ busiest\_load\_per\_task /= busiest\_nr\_running;

+ if (group\_imb)

+ busiest\_load\_per\_task = min(busiest\_load\_per\_task, avg\_load);

+

+ /\*

+ \* We're trying to get all the cpus to the average\_load, so we don't

+ \* want to push ourselves above the average load, nor do we wish to

+ \* reduce the max loaded cpu below the average load, as either of these

+ \* actions would just result in more rebalancing later, and ping-pong

+ \* tasks around. Thus we look for the minimum possible imbalance.

+ \* Negative imbalances (\*we\* are more loaded than anyone else) will

+ \* be counted as no imbalance for these purposes -- we can't fix that

+ \* by pulling tasks to us. Be careful of negative numbers as they'll

+ \* appear as very large values with unsigned longs.

+ \*/

+ if (max\_load <= busiest\_load\_per\_task)

+ goto out\_balanced;

+

+ /\*

+ \* In the presence of smp nice balancing, certain scenarios can have

+ \* max load less than avg load(as we skip the groups at or below

+ \* its cpu\_power, while calculating max\_load..)

+ \*/

+ if (max\_load < avg\_load) {

+ \*imbalance = 0;

+ goto small\_imbalance;

+ }

+

+ /\* Don't want to pull so many tasks that a group would go idle \*/

+ max\_pull = min(max\_load - avg\_load, max\_load - busiest\_load\_per\_task);

+

+ /\* How much load to actually move to equalise the imbalance \*/

+ \*imbalance = min(max\_pull \* busiest->\_\_cpu\_power,

+ (avg\_load - this\_load) \* this->\_\_cpu\_power)

+ / SCHED\_LOAD\_SCALE;

+

+ /\*

+ \* if \*imbalance is less than the average load per runnable task

+ \* there is no gaurantee that any tasks will be moved so we'll have

+ \* a think about bumping its value to force at least one task to be

+ \* moved

+ \*/

+ if (\*imbalance < busiest\_load\_per\_task) {

+ unsigned long tmp, pwr\_now, pwr\_move;

+ unsigned int imbn;

+

+small\_imbalance:

+ pwr\_move = pwr\_now = 0;

+ imbn = 2;

+ if (this\_nr\_running) {

+ this\_load\_per\_task /= this\_nr\_running;

+ if (busiest\_load\_per\_task > this\_load\_per\_task)

+ imbn = 1;

+ } else

+ this\_load\_per\_task = SCHED\_LOAD\_SCALE;

+

+ if (max\_load - this\_load + SCHED\_LOAD\_SCALE\_FUZZ >=

+ busiest\_load\_per\_task \* imbn) {

+ \*imbalance = busiest\_load\_per\_task;

+ return busiest;

+ }

+

+ /\*

+ \* OK, we don't have enough imbalance to justify moving tasks,

+ \* however we may be able to increase total CPU power used by

+ \* moving them.

+ \*/

+

+ pwr\_now += busiest->\_\_cpu\_power \*

+ min(busiest\_load\_per\_task, max\_load);

+ pwr\_now += this->\_\_cpu\_power \*

+ min(this\_load\_per\_task, this\_load);

+ pwr\_now /= SCHED\_LOAD\_SCALE;

+

+ /\* Amount of load we'd subtract \*/

+ tmp = sg\_div\_cpu\_power(busiest,

+ busiest\_load\_per\_task \* SCHED\_LOAD\_SCALE);

+ if (max\_load > tmp)

+ pwr\_move += busiest->\_\_cpu\_power \*

+ min(busiest\_load\_per\_task, max\_load - tmp);

+

+ /\* Amount of load we'd add \*/

+ if (max\_load \* busiest->\_\_cpu\_power <

+ busiest\_load\_per\_task \* SCHED\_LOAD\_SCALE)

+ tmp = sg\_div\_cpu\_power(this,

+ max\_load \* busiest->\_\_cpu\_power);

+ else

+ tmp = sg\_div\_cpu\_power(this,

+ busiest\_load\_per\_task \* SCHED\_LOAD\_SCALE);

+ pwr\_move += this->\_\_cpu\_power \*

+ min(this\_load\_per\_task, this\_load + tmp);

+ pwr\_move /= SCHED\_LOAD\_SCALE;

+

+ /\* Move if we gain throughput \*/

+ if (pwr\_move > pwr\_now)

+ \*imbalance = busiest\_load\_per\_task;

+ }

+

+ return busiest;

+

+out\_balanced:

+#if defined(CONFIG\_SCHED\_MC) || defined(CONFIG\_SCHED\_SMT)

+ if (idle == CPU\_NOT\_IDLE || !(sd->flags & SD\_POWERSAVINGS\_BALANCE))

+ goto ret;

+

+ if (this == group\_leader && group\_leader != group\_min) {

+ \*imbalance = min\_load\_per\_task;

+ return group\_min;

+ }

+#endif

+ret:

+ \*imbalance = 0;

+ return NULL;

+}

+

+/\*

+ \* find\_busiest\_queue - find the busiest runqueue among the cpus in group.

+ \*/

+static struct rq \*

+find\_busiest\_queue(struct sched\_group \*group, enum cpu\_idle\_type idle,

+ unsigned long imbalance, cpumask\_t \*cpus)

+{

+ struct rq \*busiest = NULL, \*rq;

+ unsigned long max\_load = 0;

+ int i;

+

+ for\_each\_cpu\_mask(i, group->cpumask) {

+ unsigned long wl;

+

+ if (!cpu\_isset(i, \*cpus))

+ continue;

+

+ rq = cpu\_rq(i);

+ wl = weighted\_cpuload(i);

+

+ if (rq->nr\_running == 1 && wl > imbalance)

+ continue;

+

+ if (wl > max\_load) {

+ max\_load = wl;

+ busiest = rq;

+ }

+ }

+

+ return busiest;

+}

+

+/\*

+ \* Max backoff if we encounter pinned tasks. Pretty arbitrary value, but

+ \* so long as it is large enough.

+ \*/

+#define MAX\_PINNED\_INTERVAL 512

+

+/\*

+ \* Check this\_cpu to ensure it is balanced within domain. Attempt to move

+ \* tasks if there is an imbalance.

+ \*/

+static int load\_balance(int this\_cpu, struct rq \*this\_rq,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*balance)

+{

+ int ld\_moved, all\_pinned = 0, active\_balance = 0, sd\_idle = 0;

+ struct sched\_group \*group;

+ unsigned long imbalance;

+ struct rq \*busiest;

+ cpumask\_t cpus = CPU\_MASK\_ALL;

+ unsigned long flags;

+

+ /\*

+ \* When power savings policy is enabled for the parent domain, idle

+ \* sibling can pick up load irrespective of busy siblings. In this case,

+ \* let the state of idle sibling percolate up as CPU\_IDLE, instead of

+ \* portraying it as CPU\_NOT\_IDLE.

+ \*/

+ if (idle != CPU\_NOT\_IDLE && sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ sd\_idle = 1;

+

+ schedstat\_inc(sd, lb\_count[idle]);

+

+redo:

+ group = find\_busiest\_group(sd, this\_cpu, &imbalance, idle, &sd\_idle,

+ &cpus, balance);

+

+ if (\*balance == 0)

+ goto out\_balanced;

+

+ if (!group) {

+ schedstat\_inc(sd, lb\_nobusyg[idle]);

+ goto out\_balanced;

+ }

+

+ busiest = find\_busiest\_queue(group, idle, imbalance, &cpus);

+ if (!busiest) {

+ schedstat\_inc(sd, lb\_nobusyq[idle]);

+ goto out\_balanced;

+ }

+

+ BUG\_ON(busiest == this\_rq);

+

+ schedstat\_add(sd, lb\_imbalance[idle], imbalance);

+

+ ld\_moved = 0;

+ if (busiest->nr\_running > 1) {

+ /\*

+ \* Attempt to move tasks. If find\_busiest\_group has found

+ \* an imbalance but busiest->nr\_running <= 1, the group is

+ \* still unbalanced. ld\_moved simply stays zero, so it is

+ \* correctly treated as an imbalance.

+ \*/

+ local\_irq\_save(flags);

+ double\_rq\_lock(this\_rq, busiest);

+ ld\_moved = move\_tasks(this\_rq, this\_cpu, busiest,

+ imbalance, sd, idle, &all\_pinned);

+ double\_rq\_unlock(this\_rq, busiest);

+ local\_irq\_restore(flags);

+

+ /\*

+ \* some other cpu did the load balance for us.

+ \*/

+ if (ld\_moved && this\_cpu != smp\_processor\_id())

+ resched\_cpu(this\_cpu);

+

+ /\* All tasks on this runqueue were pinned by CPU affinity \*/

+ if (unlikely(all\_pinned)) {

+ cpu\_clear(cpu\_of(busiest), cpus);

+ if (!cpus\_empty(cpus))

+ goto redo;

+ goto out\_balanced;

+ }

+ }

+

+ if (!ld\_moved) {

+ schedstat\_inc(sd, lb\_failed[idle]);

+ sd->nr\_balance\_failed++;

+

+ if (unlikely(sd->nr\_balance\_failed > sd->cache\_nice\_tries+2)) {

+

+ spin\_lock\_irqsave(&busiest->lock, flags);

+

+ /\* don't kick the migration\_thread, if the curr

+ \* task on busiest cpu can't be moved to this\_cpu

+ \*/

+ if (!cpu\_isset(this\_cpu, busiest->curr->cpus\_allowed)) {

+ spin\_unlock\_irqrestore(&busiest->lock, flags);

+ all\_pinned = 1;

+ goto out\_one\_pinned;

+ }

+

+ if (!busiest->active\_balance) {

+ busiest->active\_balance = 1;

+ busiest->push\_cpu = this\_cpu;

+ active\_balance = 1;

+ }

+ spin\_unlock\_irqrestore(&busiest->lock, flags);

+ if (active\_balance)

+ wake\_up\_process(busiest->migration\_thread);

+

+ /\*

+ \* We've kicked active balancing, reset the failure

+ \* counter.

+ \*/

+ sd->nr\_balance\_failed = sd->cache\_nice\_tries+1;

+ }

+ } else

+ sd->nr\_balance\_failed = 0;

+

+ if (likely(!active\_balance)) {

+ /\* We were unbalanced, so reset the balancing interval \*/

+ sd->balance\_interval = sd->min\_interval;

+ } else {

+ /\*

+ \* If we've begun active balancing, start to back off. This

+ \* case may not be covered by the all\_pinned logic if there

+ \* is only 1 task on the busy runqueue (because we don't call

+ \* move\_tasks).

+ \*/

+ if (sd->balance\_interval < sd->max\_interval)

+ sd->balance\_interval \*= 2;

+ }

+

+ if (!ld\_moved && !sd\_idle && sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ return -1;

+ return ld\_moved;

+

+out\_balanced:

+ schedstat\_inc(sd, lb\_balanced[idle]);

+

+ sd->nr\_balance\_failed = 0;

+

+out\_one\_pinned:

+ /\* tune up the balancing interval \*/

+ if ((all\_pinned && sd->balance\_interval < MAX\_PINNED\_INTERVAL) ||

+ (sd->balance\_interval < sd->max\_interval))

+ sd->balance\_interval \*= 2;

+

+ if (!sd\_idle && sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ return -1;

+ return 0;

+}

+

+/\*

+ \* Check this\_cpu to ensure it is balanced within domain. Attempt to move

+ \* tasks if there is an imbalance.

+ \*

+ \* Called from schedule when this\_rq is about to become idle (CPU\_NEWLY\_IDLE).

+ \* this\_rq is locked.

+ \*/

+static int

+load\_balance\_newidle(int this\_cpu, struct rq \*this\_rq, struct sched\_domain \*sd)

+{

+ struct sched\_group \*group;

+ struct rq \*busiest = NULL;

+ unsigned long imbalance;

+ int ld\_moved = 0;

+ int sd\_idle = 0;

+ int all\_pinned = 0;

+ cpumask\_t cpus = CPU\_MASK\_ALL;

+

+ /\*

+ \* When power savings policy is enabled for the parent domain, idle

+ \* sibling can pick up load irrespective of busy siblings. In this case,

+ \* let the state of idle sibling percolate up as IDLE, instead of

+ \* portraying it as CPU\_NOT\_IDLE.

+ \*/

+ if (sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ sd\_idle = 1;

+

+ schedstat\_inc(sd, lb\_count[CPU\_NEWLY\_IDLE]);

+redo:

+ group = find\_busiest\_group(sd, this\_cpu, &imbalance, CPU\_NEWLY\_IDLE,

+ &sd\_idle, &cpus, NULL);

+ if (!group) {

+ schedstat\_inc(sd, lb\_nobusyg[CPU\_NEWLY\_IDLE]);

+ goto out\_balanced;

+ }

+

+ busiest = find\_busiest\_queue(group, CPU\_NEWLY\_IDLE, imbalance,

+ &cpus);

+ if (!busiest) {

+ schedstat\_inc(sd, lb\_nobusyq[CPU\_NEWLY\_IDLE]);

+ goto out\_balanced;

+ }

+

+ BUG\_ON(busiest == this\_rq);

+

+ schedstat\_add(sd, lb\_imbalance[CPU\_NEWLY\_IDLE], imbalance);

+

+ ld\_moved = 0;

+ if (busiest->nr\_running > 1) {

+ /\* Attempt to move tasks \*/

+ double\_lock\_balance(this\_rq, busiest);

+ /\* this\_rq->clock is already updated \*/

+ update\_rq\_clock(busiest);

+ ld\_moved = move\_tasks(this\_rq, this\_cpu, busiest,

+ imbalance, sd, CPU\_NEWLY\_IDLE,

+ &all\_pinned);

+ spin\_unlock(&busiest->lock);

+

+ if (unlikely(all\_pinned)) {

+ cpu\_clear(cpu\_of(busiest), cpus);

+ if (!cpus\_empty(cpus))

+ goto redo;

+ }

+ }

+

+ if (!ld\_moved) {

+ schedstat\_inc(sd, lb\_failed[CPU\_NEWLY\_IDLE]);

+ if (!sd\_idle && sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ return -1;

+ } else

+ sd->nr\_balance\_failed = 0;

+

+ return ld\_moved;

+

+out\_balanced:

+ schedstat\_inc(sd, lb\_balanced[CPU\_NEWLY\_IDLE]);

+ if (!sd\_idle && sd->flags & SD\_SHARE\_CPUPOWER &&

+ !test\_sd\_parent(sd, SD\_POWERSAVINGS\_BALANCE))

+ return -1;

+ sd->nr\_balance\_failed = 0;

+

+ return 0;

+}

+

+/\*

+ \* idle\_balance is called by schedule() if this\_cpu is about to become

+ \* idle. Attempts to pull tasks from other CPUs.

+ \*/

+static void idle\_balance(int this\_cpu, struct rq \*this\_rq)

+{

+ struct sched\_domain \*sd;

+ int pulled\_task = -1;

+ unsigned long next\_balance = jiffies + HZ;

+

+ for\_each\_domain(this\_cpu, sd) {

+ unsigned long interval;

+

+ if (!(sd->flags & SD\_LOAD\_BALANCE))

+ continue;

+

+ if (sd->flags & SD\_BALANCE\_NEWIDLE)

+ /\* If we've pulled tasks over stop searching: \*/

+ pulled\_task = load\_balance\_newidle(this\_cpu,

+ this\_rq, sd);

+

+ interval = msecs\_to\_jiffies(sd->balance\_interval);

+ if (time\_after(next\_balance, sd->last\_balance + interval))

+ next\_balance = sd->last\_balance + interval;

+ if (pulled\_task)

+ break;

+ }

+ if (pulled\_task || time\_after(jiffies, this\_rq->next\_balance)) {

+ /\*

+ \* We are going idle. next\_balance may be set based on

+ \* a busy processor. So reset next\_balance.

+ \*/

+ this\_rq->next\_balance = next\_balance;

+ }

+}

+

+/\*

+ \* active\_load\_balance is run by migration threads. It pushes running tasks

+ \* off the busiest CPU onto idle CPUs. It requires at least 1 task to be

+ \* running on each physical CPU where possible, and avoids physical /

+ \* logical imbalances.

+ \*

+ \* Called with busiest\_rq locked.

+ \*/

+static void active\_load\_balance(struct rq \*busiest\_rq, int busiest\_cpu)

+{

+ int target\_cpu = busiest\_rq->push\_cpu;

+ struct sched\_domain \*sd;

+ struct rq \*target\_rq;

+

+ /\* Is there any task to move? \*/

+ if (busiest\_rq->nr\_running <= 1)

+ return;

+

+ target\_rq = cpu\_rq(target\_cpu);

+

+ /\*

+ \* This condition is "impossible", if it occurs

+ \* we need to fix it. Originally reported by

+ \* Bjorn Helgaas on a 128-cpu setup.

+ \*/

+ BUG\_ON(busiest\_rq == target\_rq);

+

+ /\* move a task from busiest\_rq to target\_rq \*/

+ double\_lock\_balance(busiest\_rq, target\_rq);

+ update\_rq\_clock(busiest\_rq);

+ update\_rq\_clock(target\_rq);

+

+ /\* Search for an sd spanning us and the target CPU. \*/

+ for\_each\_domain(target\_cpu, sd) {

+ if ((sd->flags & SD\_LOAD\_BALANCE) &&

+ cpu\_isset(busiest\_cpu, sd->span))

+ break;

+ }

+

+ if (likely(sd)) {

+ schedstat\_inc(sd, alb\_count);

+

+ if (move\_one\_task(target\_rq, target\_cpu, busiest\_rq,

+ sd, CPU\_IDLE))

+ schedstat\_inc(sd, alb\_pushed);

+ else

+ schedstat\_inc(sd, alb\_failed);

+ }

+ spin\_unlock(&target\_rq->lock);

+}

+

+#ifdef CONFIG\_NO\_HZ

+static struct {

+ atomic\_t load\_balancer;

+ cpumask\_t cpu\_mask;

+} nohz \_\_\_\_cacheline\_aligned = {

+ .load\_balancer = ATOMIC\_INIT(-1),

+ .cpu\_mask = CPU\_MASK\_NONE,

+};

+

+/\*

+ \* This routine will try to nominate the ilb (idle load balancing)

+ \* owner among the cpus whose ticks are stopped. ilb owner will do the idle

+ \* load balancing on behalf of all those cpus. If all the cpus in the system

+ \* go into this tickless mode, then there will be no ilb owner (as there is

+ \* no need for one) and all the cpus will sleep till the next wakeup event

+ \* arrives...

+ \*

+ \* For the ilb owner, tick is not stopped. And this tick will be used

+ \* for idle load balancing. ilb owner will still be part of

+ \* nohz.cpu\_mask..

+ \*

+ \* While stopping the tick, this cpu will become the ilb owner if there

+ \* is no other owner. And will be the owner till that cpu becomes busy

+ \* or if all cpus in the system stop their ticks at which point

+ \* there is no need for ilb owner.

+ \*

+ \* When the ilb owner becomes busy, it nominates another owner, during the

+ \* next busy scheduler\_tick()

+ \*/

+int select\_nohz\_load\_balancer(int stop\_tick)

+{

+ int cpu = smp\_processor\_id();

+

+ if (stop\_tick) {

+ cpu\_set(cpu, nohz.cpu\_mask);

+ cpu\_rq(cpu)->in\_nohz\_recently = 1;

+

+ /\*

+ \* If we are going offline and still the leader, give up!

+ \*/

+ if (cpu\_is\_offline(cpu) &&

+ atomic\_read(&nohz.load\_balancer) == cpu) {

+ if (atomic\_cmpxchg(&nohz.load\_balancer, cpu, -1) != cpu)

+ BUG();

+ return 0;

+ }

+

+ /\* time for ilb owner also to sleep \*/

+ if (cpus\_weight(nohz.cpu\_mask) == num\_online\_cpus()) {

+ if (atomic\_read(&nohz.load\_balancer) == cpu)

+ atomic\_set(&nohz.load\_balancer, -1);

+ return 0;

+ }

+

+ if (atomic\_read(&nohz.load\_balancer) == -1) {

+ /\* make me the ilb owner \*/

+ if (atomic\_cmpxchg(&nohz.load\_balancer, -1, cpu) == -1)

+ return 1;

+ } else if (atomic\_read(&nohz.load\_balancer) == cpu)

+ return 1;

+ } else {

+ if (!cpu\_isset(cpu, nohz.cpu\_mask))

+ return 0;

+

+ cpu\_clear(cpu, nohz.cpu\_mask);

+

+ if (atomic\_read(&nohz.load\_balancer) == cpu)

+ if (atomic\_cmpxchg(&nohz.load\_balancer, cpu, -1) != cpu)

+ BUG();

+ }

+ return 0;

+}

+#endif

+

+static DEFINE\_SPINLOCK(balancing);

+

+/\*

+ \* It checks each scheduling domain to see if it is due to be balanced,

+ \* and initiates a balancing operation if so.

+ \*

+ \* Balancing parameters are set up in arch\_init\_sched\_domains.

+ \*/

+static void rebalance\_domains(int cpu, enum cpu\_idle\_type idle)

+{

+ int balance = 1;

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long interval;

+ struct sched\_domain \*sd;

+ /\* Earliest time when we have to do rebalance again \*/

+ unsigned long next\_balance = jiffies + 60\*HZ;

+ int update\_next\_balance = 0;

+

+ for\_each\_domain(cpu, sd) {

+ if (!(sd->flags & SD\_LOAD\_BALANCE))

+ continue;

+

+ interval = sd->balance\_interval;

+ if (idle != CPU\_IDLE)

+ interval \*= sd->busy\_factor;

+

+ /\* scale ms to jiffies \*/

+ interval = msecs\_to\_jiffies(interval);

+ if (unlikely(!interval))

+ interval = 1;

+ if (interval > HZ\*NR\_CPUS/10)

+ interval = HZ\*NR\_CPUS/10;

+

+

+ if (sd->flags & SD\_SERIALIZE) {

+ if (!spin\_trylock(&balancing))

+ goto out;

+ }

+

+ if (time\_after\_eq(jiffies, sd->last\_balance + interval)) {

+ if (load\_balance(cpu, rq, sd, idle, &balance)) {

+ /\*

+ \* We've pulled tasks over so either we're no

+ \* longer idle, or one of our SMT siblings is

+ \* not idle.

+ \*/

+ idle = CPU\_NOT\_IDLE;

+ }

+ sd->last\_balance = jiffies;

+ }

+ if (sd->flags & SD\_SERIALIZE)

+ spin\_unlock(&balancing);

+out:

+ if (time\_after(next\_balance, sd->last\_balance + interval)) {

+ next\_balance = sd->last\_balance + interval;

+ update\_next\_balance = 1;

+ }

+

+ /\*

+ \* Stop the load balance at this level. There is another

+ \* CPU in our sched group which is doing load balancing more

+ \* actively.

+ \*/

+ if (!balance)

+ break;

+ }

+

+ /\*

+ \* next\_balance will be updated only when there is a need.

+ \* When the cpu is attached to null domain for ex, it will not be

+ \* updated.

+ \*/

+ if (likely(update\_next\_balance))

+ rq->next\_balance = next\_balance;

+}

+

+/\*

+ \* run\_rebalance\_domains is triggered when needed from the scheduler tick.

+ \* In CONFIG\_NO\_HZ case, the idle load balance owner will do the

+ \* rebalancing for all the cpus for whom scheduler ticks are stopped.

+ \*/

+static void run\_rebalance\_domains(struct softirq\_action \*h)

+{

+ int this\_cpu = smp\_processor\_id();

+ struct rq \*this\_rq = cpu\_rq(this\_cpu);

+ enum cpu\_idle\_type idle = this\_rq->idle\_at\_tick ?

+ CPU\_IDLE : CPU\_NOT\_IDLE;

+

+ rebalance\_domains(this\_cpu, idle);

+

+#ifdef CONFIG\_NO\_HZ

+ /\*

+ \* If this cpu is the owner for idle load balancing, then do the

+ \* balancing on behalf of the other idle cpus whose ticks are

+ \* stopped.

+ \*/

+ if (this\_rq->idle\_at\_tick &&

+ atomic\_read(&nohz.load\_balancer) == this\_cpu) {

+ cpumask\_t cpus = nohz.cpu\_mask;

+ struct rq \*rq;

+ int balance\_cpu;

+

+ cpu\_clear(this\_cpu, cpus);

+ for\_each\_cpu\_mask(balance\_cpu, cpus) {

+ /\*

+ \* If this cpu gets work to do, stop the load balancing

+ \* work being done for other cpus. Next load

+ \* balancing owner will pick it up.

+ \*/

+ if (need\_resched())

+ break;

+

+ rebalance\_domains(balance\_cpu, CPU\_IDLE);

+

+ rq = cpu\_rq(balance\_cpu);

+ if (time\_after(this\_rq->next\_balance, rq->next\_balance))

+ this\_rq->next\_balance = rq->next\_balance;

+ }

+ }

+#endif

+}

+

+/\*

+ \* Trigger the SCHED\_SOFTIRQ if it is time to do periodic load balancing.

+ \*

+ \* In case of CONFIG\_NO\_HZ, this is the place where we nominate a new

+ \* idle load balancing owner or decide to stop the periodic load balancing,

+ \* if the whole system is idle.

+ \*/

+static inline void trigger\_load\_balance(struct rq \*rq, int cpu)

+{

+#ifdef CONFIG\_NO\_HZ

+ /\*

+ \* If we were in the nohz mode recently and busy at the current

+ \* scheduler tick, then check if we need to nominate new idle

+ \* load balancer.

+ \*/

+ if (rq->in\_nohz\_recently && !rq->idle\_at\_tick) {

+ rq->in\_nohz\_recently = 0;

+

+ if (atomic\_read(&nohz.load\_balancer) == cpu) {

+ cpu\_clear(cpu, nohz.cpu\_mask);

+ atomic\_set(&nohz.load\_balancer, -1);

+ }

+

+ if (atomic\_read(&nohz.load\_balancer) == -1) {

+ /\*

+ \* simple selection for now: Nominate the

+ \* first cpu in the nohz list to be the next

+ \* ilb owner.

+ \*

+ \* TBD: Traverse the sched domains and nominate

+ \* the nearest cpu in the nohz.cpu\_mask.

+ \*/

+ int ilb = first\_cpu(nohz.cpu\_mask);

+

+ if (ilb != NR\_CPUS)

+ resched\_cpu(ilb);

+ }

+ }

+

+ /\*

+ \* If this cpu is idle and doing idle load balancing for all the

+ \* cpus with ticks stopped, is it time for that to stop?

+ \*/

+ if (rq->idle\_at\_tick && atomic\_read(&nohz.load\_balancer) == cpu &&

+ cpus\_weight(nohz.cpu\_mask) == num\_online\_cpus()) {

+ resched\_cpu(cpu);

+ return;

+ }

+

+ /\*

+ \* If this cpu is idle and the idle load balancing is done by

+ \* someone else, then no need raise the SCHED\_SOFTIRQ

+ \*/

+ if (rq->idle\_at\_tick && atomic\_read(&nohz.load\_balancer) != cpu &&

+ cpu\_isset(cpu, nohz.cpu\_mask))

+ return;

+#endif

+ if (time\_after\_eq(jiffies, rq->next\_balance))

+ raise\_softirq(SCHED\_SOFTIRQ);

+}

+

+#else /\* CONFIG\_SMP \*/

+

+/\*

+ \* on UP we do not need to balance between CPUs:

+ \*/

+static inline void idle\_balance(int cpu, struct rq \*rq)

+{

+}

+

+#endif

+

+DEFINE\_PER\_CPU(struct kernel\_stat, kstat);

+

+EXPORT\_PER\_CPU\_SYMBOL(kstat);

+

+/\*

+ \* Return p->sum\_exec\_runtime plus any more ns on the sched\_clock

+ \* that have not yet been banked in case the task is currently running.

+ \*/

+unsigned long long task\_sched\_runtime(struct task\_struct \*p)

+{

+ unsigned long flags;

+ u64 ns, delta\_exec;

+ struct rq \*rq;

+

+ rq = task\_rq\_lock(p, &flags);

+ ns = p->se.sum\_exec\_runtime;

+ if (task\_current(rq, p)) {

+ update\_rq\_clock(rq);

+ delta\_exec = rq->clock - p->se.exec\_start;

+ if ((s64)delta\_exec > 0)

+ ns += delta\_exec;

+ }

+ task\_rq\_unlock(rq, &flags);

+

+ return ns;

+}

+

+/\*

+ \* Account user cpu time to a process.

+ \* @p: the process that the cpu time gets accounted to

+ \* @cputime: the cpu time spent in user space since the last update

+ \*/

+void account\_user\_time(struct task\_struct \*p, cputime\_t cputime)

+{

+ struct cpu\_usage\_stat \*cpustat = &kstat\_this\_cpu.cpustat;

+ cputime64\_t tmp;

+

+ p->utime = cputime\_add(p->utime, cputime);

+

+ /\* Add user time to cpustat. \*/

+ tmp = cputime\_to\_cputime64(cputime);

+ if (TASK\_NICE(p) > 0)

+ cpustat->nice = cputime64\_add(cpustat->nice, tmp);

+ else

+ cpustat->user = cputime64\_add(cpustat->user, tmp);

+}

+

+/\*

+ \* Account guest cpu time to a process.

+ \* @p: the process that the cpu time gets accounted to

+ \* @cputime: the cpu time spent in virtual machine since the last update

+ \*/

+static void account\_guest\_time(struct task\_struct \*p, cputime\_t cputime)

+{

+ cputime64\_t tmp;

+ struct cpu\_usage\_stat \*cpustat = &kstat\_this\_cpu.cpustat;

+

+ tmp = cputime\_to\_cputime64(cputime);

+

+ p->utime = cputime\_add(p->utime, cputime);

+ p->gtime = cputime\_add(p->gtime, cputime);

+

+ cpustat->user = cputime64\_add(cpustat->user, tmp);

+ cpustat->guest = cputime64\_add(cpustat->guest, tmp);

+}

+

+/\*

+ \* Account scaled user cpu time to a process.

+ \* @p: the process that the cpu time gets accounted to

+ \* @cputime: the cpu time spent in user space since the last update

+ \*/

+void account\_user\_time\_scaled(struct task\_struct \*p, cputime\_t cputime)

+{

+ p->utimescaled = cputime\_add(p->utimescaled, cputime);

+}

+

+/\*

+ \* Account system cpu time to a process.

+ \* @p: the process that the cpu time gets accounted to

+ \* @hardirq\_offset: the offset to subtract from hardirq\_count()

+ \* @cputime: the cpu time spent in kernel space since the last update

+ \*/

+void account\_system\_time(struct task\_struct \*p, int hardirq\_offset,

+ cputime\_t cputime)

+{

+ struct cpu\_usage\_stat \*cpustat = &kstat\_this\_cpu.cpustat;

+ struct rq \*rq = this\_rq();

+ cputime64\_t tmp;

+

+ if ((p->flags & PF\_VCPU) && (irq\_count() - hardirq\_offset == 0))

+ return account\_guest\_time(p, cputime);

+

+ p->stime = cputime\_add(p->stime, cputime);

+

+ /\* Add system time to cpustat. \*/

+ tmp = cputime\_to\_cputime64(cputime);

+ if (hardirq\_count() - hardirq\_offset)

+ cpustat->irq = cputime64\_add(cpustat->irq, tmp);

+ else if (softirq\_count())

+ cpustat->softirq = cputime64\_add(cpustat->softirq, tmp);

+ else if (p != rq->idle)

+ cpustat->system = cputime64\_add(cpustat->system, tmp);

+ else if (atomic\_read(&rq->nr\_iowait) > 0)

+ cpustat->iowait = cputime64\_add(cpustat->iowait, tmp);

+ else

+ cpustat->idle = cputime64\_add(cpustat->idle, tmp);

+ /\* Account for system time used \*/

+ acct\_update\_integrals(p);

+}

+

+/\*

+ \* Account scaled system cpu time to a process.

+ \* @p: the process that the cpu time gets accounted to

+ \* @hardirq\_offset: the offset to subtract from hardirq\_count()

+ \* @cputime: the cpu time spent in kernel space since the last update

+ \*/

+void account\_system\_time\_scaled(struct task\_struct \*p, cputime\_t cputime)

+{

+ p->stimescaled = cputime\_add(p->stimescaled, cputime);

+}

+

+/\*

+ \* Account for involuntary wait time.

+ \* @p: the process from which the cpu time has been stolen

+ \* @steal: the cpu time spent in involuntary wait

+ \*/

+void account\_steal\_time(struct task\_struct \*p, cputime\_t steal)

+{

+ struct cpu\_usage\_stat \*cpustat = &kstat\_this\_cpu.cpustat;

+ cputime64\_t tmp = cputime\_to\_cputime64(steal);

+ struct rq \*rq = this\_rq();

+

+ if (p == rq->idle) {

+ p->stime = cputime\_add(p->stime, steal);

+ if (atomic\_read(&rq->nr\_iowait) > 0)

+ cpustat->iowait = cputime64\_add(cpustat->iowait, tmp);

+ else

+ cpustat->idle = cputime64\_add(cpustat->idle, tmp);

+ } else

+ cpustat->steal = cputime64\_add(cpustat->steal, tmp);

+}

+

+/\*

+ \* This function gets called by the timer code, with HZ frequency.

+ \* We call it with interrupts disabled.

+ \*

+ \* It also gets called by the fork code, when changing the parent's

+ \* timeslices.

+ \*/

+void scheduler\_tick(void)

+{

+ int cpu = smp\_processor\_id();

+ struct rq \*rq = cpu\_rq(cpu);

+ struct task\_struct \*curr = rq->curr;

+ u64 next\_tick = rq->tick\_timestamp + TICK\_NSEC;

+

+ spin\_lock(&rq->lock);

+ \_\_update\_rq\_clock(rq);

+ /\*

+ \* Let rq->clock advance by at least TICK\_NSEC:

+ \*/

+ if (unlikely(rq->clock < next\_tick))

+ rq->clock = next\_tick;

+ rq->tick\_timestamp = rq->clock;

+ update\_cpu\_load(rq);

+ if (curr != rq->idle) /\* FIXME: needed? \*/

+ curr->sched\_class->task\_tick(rq, curr);

+ spin\_unlock(&rq->lock);

+

+#ifdef CONFIG\_SMP

+ rq->idle\_at\_tick = idle\_cpu(cpu);

+ trigger\_load\_balance(rq, cpu);

+#endif

+}

+

+#if defined(CONFIG\_PREEMPT) && defined(CONFIG\_DEBUG\_PREEMPT)

+

+void fastcall add\_preempt\_count(int val)

+{

+ /\*

+ \* Underflow?

+ \*/

+ if (DEBUG\_LOCKS\_WARN\_ON((preempt\_count() < 0)))

+ return;

+ preempt\_count() += val;

+ /\*

+ \* Spinlock count overflowing soon?

+ \*/

+ DEBUG\_LOCKS\_WARN\_ON((preempt\_count() & PREEMPT\_MASK) >=

+ PREEMPT\_MASK - 10);

+}

+EXPORT\_SYMBOL(add\_preempt\_count);

+

+void fastcall sub\_preempt\_count(int val)

+{

+ /\*

+ \* Underflow?

+ \*/

+ if (DEBUG\_LOCKS\_WARN\_ON(val > preempt\_count()))

+ return;

+ /\*

+ \* Is the spinlock portion underflowing?

+ \*/

+ if (DEBUG\_LOCKS\_WARN\_ON((val < PREEMPT\_MASK) &&

+ !(preempt\_count() & PREEMPT\_MASK)))

+ return;

+

+ preempt\_count() -= val;

+}

+EXPORT\_SYMBOL(sub\_preempt\_count);

+

+#endif

+

+/\*

+ \* Print scheduling while atomic bug:

+ \*/

+static noinline void \_\_schedule\_bug(struct task\_struct \*prev)

+{

+ struct pt\_regs \*regs = get\_irq\_regs();

+

+ printk(KERN\_ERR "BUG: scheduling while atomic: %s/%d/0x%08x\n",

+ prev->comm, prev->pid, preempt\_count());

+

+ debug\_show\_held\_locks(prev);

+ if (irqs\_disabled())

+ print\_irqtrace\_events(prev);

+

+ if (regs)

+ show\_regs(regs);

+ else

+ dump\_stack();

+}

+

+/\*

+ \* Various schedule()-time debugging checks and statistics:

+ \*/

+static inline void schedule\_debug(struct task\_struct \*prev)

+{

+ /\*

+ \* Test if we are atomic. Since do\_exit() needs to call into

+ \* schedule() atomically, we ignore that path for now.

+ \* Otherwise, whine if we are scheduling when we should not be.

+ \*/

+ if (unlikely(in\_atomic\_preempt\_off()) && unlikely(!prev->exit\_state))

+ \_\_schedule\_bug(prev);

+

+ profile\_hit(SCHED\_PROFILING, \_\_builtin\_return\_address(0));

+

+ schedstat\_inc(this\_rq(), sched\_count);

+#ifdef CONFIG\_SCHEDSTATS

+ if (unlikely(prev->lock\_depth >= 0)) {

+ schedstat\_inc(this\_rq(), bkl\_count);

+ schedstat\_inc(prev, sched\_info.bkl\_count);

+ }

+#endif

+}

+

+/\*

+ \* Pick up the highest-prio task:

+ \*/

+static inline struct task\_struct \*

+pick\_next\_task(struct rq \*rq, struct task\_struct \*prev)

+{

+ const struct sched\_class \*class;

+ struct task\_struct \*p;

+

+ /\*

+ \* Optimization: we know that if all tasks are in

+ \* the fair class we can call that function directly:

+ \*/

+/\*

+ if (likely(rq->nr\_running == rq->cfs.nr\_running)) {

+ p = fair\_sched\_class.pick\_next\_task(rq);

+ if (likely(p))

+ return p;

+ }

+\*/

+ class = sched\_class\_highest;

+ for ( ; ; ) {

+ p = class->pick\_next\_task(rq);

+ if (p)

+ return p;

+ /\*

+ \* Will never be NULL as the idle class always

+ \* returns a non-NULL p:

+ \*/

+ class = class->next;

+ }

+}

+

+/\*

+ \* schedule() is the main scheduler function.

+ \*/

+asmlinkage void \_\_sched schedule(void)

+{

+ struct task\_struct \*prev, \*next;

+ long \*switch\_count;

+ struct rq \*rq;

+ int cpu;

+

+need\_resched:

+ preempt\_disable();

+ cpu = smp\_processor\_id();

+ rq = cpu\_rq(cpu);

+ rcu\_qsctr\_inc(cpu);

+ prev = rq->curr;

+ switch\_count = &prev->nivcsw;

+

+ release\_kernel\_lock(prev);

+need\_resched\_nonpreemptible:

+

+ schedule\_debug(prev);

+

+ /\*

+ \* Do the rq-clock update outside the rq lock:

+ \*/

+ local\_irq\_disable();

+ \_\_update\_rq\_clock(rq);

+ spin\_lock(&rq->lock);

+ clear\_tsk\_need\_resched(prev);

+

+ if (prev->state && !(preempt\_count() & PREEMPT\_ACTIVE)) {

+ if (unlikely((prev->state & TASK\_INTERRUPTIBLE) &&

+ unlikely(signal\_pending(prev)))) {

+ prev->state = TASK\_RUNNING;

+ } else {

+ deactivate\_task(rq, prev, 1);

+ }

+ switch\_count = &prev->nvcsw;

+ }

+

+ if (unlikely(!rq->nr\_running))

+ idle\_balance(cpu, rq);

+

+ prev->sched\_class->put\_prev\_task(rq, prev);

+ next = pick\_next\_task(rq, prev);

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ char msg[CASIO\_MSG\_SIZE];

+ if(prev->policy==SCHED\_CASIO || next->policy==SCHED\_CASIO){

+ if(prev->policy==SCHED\_CASIO && next->policy==SCHED\_CASIO){

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(%d:%d),next->(%d:%d)",prev->casio\_id,prev->pid,next->casio\_id,next->pid);

+ }

+ else{

+ if(prev->policy==SCHED\_CASIO){

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(%d:%d),next->(-1:%d)",prev->casio\_id,prev->pid,next->pid);

+ }else{

+ snprintf(msg,CASIO\_MSG\_SIZE,"prev->(-1:%d),next->(%d:%d)",prev->pid,next->casio\_id,next->pid);

+ }

+ }

+ register\_casio\_event(sched\_clock(), msg, CASIO\_CONTEXT\_SWITCH);

+

+

+ }

+#endif

+

+ sched\_info\_switch(prev, next);

+

+ if (likely(prev != next)) {

+ rq->nr\_switches++;

+ rq->curr = next;

+ ++\*switch\_count;

+

+ context\_switch(rq, prev, next); /\* unlocks the rq \*/

+ } else

+ spin\_unlock\_irq(&rq->lock);

+

+ if (unlikely(reacquire\_kernel\_lock(current) < 0)) {

+ cpu = smp\_processor\_id();

+ rq = cpu\_rq(cpu);

+ goto need\_resched\_nonpreemptible;

+ }

+ preempt\_enable\_no\_resched();

+ if (unlikely(test\_thread\_flag(TIF\_NEED\_RESCHED)))

+ goto need\_resched;

+}

+EXPORT\_SYMBOL(schedule);

+

+#ifdef CONFIG\_PREEMPT

+/\*

+ \* this is the entry point to schedule() from in-kernel preemption

+ \* off of preempt\_enable. Kernel preemptions off return from interrupt

+ \* occur there and call schedule directly.

+ \*/

+asmlinkage void \_\_sched preempt\_schedule(void)

+{

+ struct thread\_info \*ti = current\_thread\_info();

+#ifdef CONFIG\_PREEMPT\_BKL

+ struct task\_struct \*task = current;

+ int saved\_lock\_depth;

+#endif

+ /\*

+ \* If there is a non-zero preempt\_count or interrupts are disabled,

+ \* we do not want to preempt the current task. Just return..

+ \*/

+ if (likely(ti->preempt\_count || irqs\_disabled()))

+ return;

+

+ do {

+ add\_preempt\_count(PREEMPT\_ACTIVE);

+

+ /\*

+ \* We keep the big kernel semaphore locked, but we

+ \* clear ->lock\_depth so that schedule() doesnt

+ \* auto-release the semaphore:

+ \*/

+#ifdef CONFIG\_PREEMPT\_BKL

+ saved\_lock\_depth = task->lock\_depth;

+ task->lock\_depth = -1;

+#endif

+ schedule();

+#ifdef CONFIG\_PREEMPT\_BKL

+ task->lock\_depth = saved\_lock\_depth;

+#endif

+ sub\_preempt\_count(PREEMPT\_ACTIVE);

+

+ /\*

+ \* Check again in case we missed a preemption opportunity

+ \* between schedule and now.

+ \*/

+ barrier();

+ } while (unlikely(test\_thread\_flag(TIF\_NEED\_RESCHED)));

+}

+EXPORT\_SYMBOL(preempt\_schedule);

+

+/\*

+ \* this is the entry point to schedule() from kernel preemption

+ \* off of irq context.

+ \* Note, that this is called and return with irqs disabled. This will

+ \* protect us against recursive calling from irq.

+ \*/

+asmlinkage void \_\_sched preempt\_schedule\_irq(void)

+{

+ struct thread\_info \*ti = current\_thread\_info();

+#ifdef CONFIG\_PREEMPT\_BKL

+ struct task\_struct \*task = current;

+ int saved\_lock\_depth;

+#endif

+ /\* Catch callers which need to be fixed \*/

+ BUG\_ON(ti->preempt\_count || !irqs\_disabled());

+

+ do {

+ add\_preempt\_count(PREEMPT\_ACTIVE);

+

+ /\*

+ \* We keep the big kernel semaphore locked, but we

+ \* clear ->lock\_depth so that schedule() doesnt

+ \* auto-release the semaphore:

+ \*/

+#ifdef CONFIG\_PREEMPT\_BKL

+ saved\_lock\_depth = task->lock\_depth;

+ task->lock\_depth = -1;

+#endif

+ local\_irq\_enable();

+ schedule();

+ local\_irq\_disable();

+#ifdef CONFIG\_PREEMPT\_BKL

+ task->lock\_depth = saved\_lock\_depth;

+#endif

+ sub\_preempt\_count(PREEMPT\_ACTIVE);

+

+ /\*

+ \* Check again in case we missed a preemption opportunity

+ \* between schedule and now.

+ \*/

+ barrier();

+ } while (unlikely(test\_thread\_flag(TIF\_NEED\_RESCHED)));

+}

+

+#endif /\* CONFIG\_PREEMPT \*/

+

+int default\_wake\_function(wait\_queue\_t \*curr, unsigned mode, int sync,

+ void \*key)

+{

+ return try\_to\_wake\_up(curr->private, mode, sync);

+}

+EXPORT\_SYMBOL(default\_wake\_function);

+

+/\*

+ \* The core wakeup function. Non-exclusive wakeups (nr\_exclusive == 0) just

+ \* wake everything up. If it's an exclusive wakeup (nr\_exclusive == small +ve

+ \* number) then we wake all the non-exclusive tasks and one exclusive task.

+ \*

+ \* There are circumstances in which we can try to wake a task which has already

+ \* started to run but is not in state TASK\_RUNNING. try\_to\_wake\_up() returns

+ \* zero in this (rare) case, and we handle it by continuing to scan the queue.

+ \*/

+static void \_\_wake\_up\_common(wait\_queue\_head\_t \*q, unsigned int mode,

+ int nr\_exclusive, int sync, void \*key)

+{

+ wait\_queue\_t \*curr, \*next;

+

+ list\_for\_each\_entry\_safe(curr, next, &q->task\_list, task\_list) {

+ unsigned flags = curr->flags;

+

+ if (curr->func(curr, mode, sync, key) &&

+ (flags & WQ\_FLAG\_EXCLUSIVE) && !--nr\_exclusive)

+ break;

+ }

+}

+

+/\*\*

+ \* \_\_wake\_up - wake up threads blocked on a waitqueue.

+ \* @q: the waitqueue

+ \* @mode: which threads

+ \* @nr\_exclusive: how many wake-one or wake-many threads to wake up

+ \* @key: is directly passed to the wakeup function

+ \*/

+void fastcall \_\_wake\_up(wait\_queue\_head\_t \*q, unsigned int mode,

+ int nr\_exclusive, void \*key)

+{

+ unsigned long flags;

+

+ spin\_lock\_irqsave(&q->lock, flags);

+ \_\_wake\_up\_common(q, mode, nr\_exclusive, 0, key);

+ spin\_unlock\_irqrestore(&q->lock, flags);

+}

+EXPORT\_SYMBOL(\_\_wake\_up);

+

+/\*

+ \* Same as \_\_wake\_up but called with the spinlock in wait\_queue\_head\_t held.

+ \*/

+void fastcall \_\_wake\_up\_locked(wait\_queue\_head\_t \*q, unsigned int mode)

+{

+ \_\_wake\_up\_common(q, mode, 1, 0, NULL);

+}

+

+/\*\*

+ \* \_\_wake\_up\_sync - wake up threads blocked on a waitqueue.

+ \* @q: the waitqueue

+ \* @mode: which threads

+ \* @nr\_exclusive: how many wake-one or wake-many threads to wake up

+ \*

+ \* The sync wakeup differs that the waker knows that it will schedule

+ \* away soon, so while the target thread will be woken up, it will not

+ \* be migrated to another CPU - ie. the two threads are 'synchronized'

+ \* with each other. This can prevent needless bouncing between CPUs.

+ \*

+ \* On UP it can prevent extra preemption.

+ \*/

+void fastcall

+\_\_wake\_up\_sync(wait\_queue\_head\_t \*q, unsigned int mode, int nr\_exclusive)

+{

+ unsigned long flags;

+ int sync = 1;

+

+ if (unlikely(!q))

+ return;

+

+ if (unlikely(!nr\_exclusive))

+ sync = 0;

+

+ spin\_lock\_irqsave(&q->lock, flags);

+ \_\_wake\_up\_common(q, mode, nr\_exclusive, sync, NULL);

+ spin\_unlock\_irqrestore(&q->lock, flags);

+}

+EXPORT\_SYMBOL\_GPL(\_\_wake\_up\_sync); /\* For internal use only \*/

+

+void complete(struct completion \*x)

+{

+ unsigned long flags;

+

+ spin\_lock\_irqsave(&x->wait.lock, flags);

+ x->done++;

+ \_\_wake\_up\_common(&x->wait, TASK\_UNINTERRUPTIBLE | TASK\_INTERRUPTIBLE,

+ 1, 0, NULL);

+ spin\_unlock\_irqrestore(&x->wait.lock, flags);

+}

+EXPORT\_SYMBOL(complete);

+

+void complete\_all(struct completion \*x)

+{

+ unsigned long flags;

+

+ spin\_lock\_irqsave(&x->wait.lock, flags);

+ x->done += UINT\_MAX/2;

+ \_\_wake\_up\_common(&x->wait, TASK\_UNINTERRUPTIBLE | TASK\_INTERRUPTIBLE,

+ 0, 0, NULL);

+ spin\_unlock\_irqrestore(&x->wait.lock, flags);

+}

+EXPORT\_SYMBOL(complete\_all);

+

+static inline long \_\_sched

+do\_wait\_for\_common(struct completion \*x, long timeout, int state)

+{

+ if (!x->done) {

+ DECLARE\_WAITQUEUE(wait, current);

+

+ wait.flags |= WQ\_FLAG\_EXCLUSIVE;

+ \_\_add\_wait\_queue\_tail(&x->wait, &wait);

+ do {

+ if (state == TASK\_INTERRUPTIBLE &&

+ signal\_pending(current)) {

+ \_\_remove\_wait\_queue(&x->wait, &wait);

+ return -ERESTARTSYS;

+ }

+ \_\_set\_current\_state(state);

+ spin\_unlock\_irq(&x->wait.lock);

+ timeout = schedule\_timeout(timeout);

+ spin\_lock\_irq(&x->wait.lock);

+ if (!timeout) {

+ \_\_remove\_wait\_queue(&x->wait, &wait);

+ return timeout;

+ }

+ } while (!x->done);

+ \_\_remove\_wait\_queue(&x->wait, &wait);

+ }

+ x->done--;

+ return timeout;

+}

+

+static long \_\_sched

+wait\_for\_common(struct completion \*x, long timeout, int state)

+{

+ might\_sleep();

+

+ spin\_lock\_irq(&x->wait.lock);

+ timeout = do\_wait\_for\_common(x, timeout, state);

+ spin\_unlock\_irq(&x->wait.lock);

+ return timeout;

+}

+

+void \_\_sched wait\_for\_completion(struct completion \*x)

+{

+ wait\_for\_common(x, MAX\_SCHEDULE\_TIMEOUT, TASK\_UNINTERRUPTIBLE);

+}

+EXPORT\_SYMBOL(wait\_for\_completion);

+

+unsigned long \_\_sched

+wait\_for\_completion\_timeout(struct completion \*x, unsigned long timeout)

+{

+ return wait\_for\_common(x, timeout, TASK\_UNINTERRUPTIBLE);

+}

+EXPORT\_SYMBOL(wait\_for\_completion\_timeout);

+

+int \_\_sched wait\_for\_completion\_interruptible(struct completion \*x)

+{

+ long t = wait\_for\_common(x, MAX\_SCHEDULE\_TIMEOUT, TASK\_INTERRUPTIBLE);

+ if (t == -ERESTARTSYS)

+ return t;

+ return 0;

+}

+EXPORT\_SYMBOL(wait\_for\_completion\_interruptible);

+

+unsigned long \_\_sched

+wait\_for\_completion\_interruptible\_timeout(struct completion \*x,

+ unsigned long timeout)

+{

+ return wait\_for\_common(x, timeout, TASK\_INTERRUPTIBLE);

+}

+EXPORT\_SYMBOL(wait\_for\_completion\_interruptible\_timeout);

+

+static long \_\_sched

+sleep\_on\_common(wait\_queue\_head\_t \*q, int state, long timeout)

+{

+ unsigned long flags;

+ wait\_queue\_t wait;

+

+ init\_waitqueue\_entry(&wait, current);

+

+ \_\_set\_current\_state(state);

+

+ spin\_lock\_irqsave(&q->lock, flags);

+ \_\_add\_wait\_queue(q, &wait);

+ spin\_unlock(&q->lock);

+ timeout = schedule\_timeout(timeout);

+ spin\_lock\_irq(&q->lock);

+ \_\_remove\_wait\_queue(q, &wait);

+ spin\_unlock\_irqrestore(&q->lock, flags);

+

+ return timeout;

+}

+

+void \_\_sched interruptible\_sleep\_on(wait\_queue\_head\_t \*q)

+{

+ sleep\_on\_common(q, TASK\_INTERRUPTIBLE, MAX\_SCHEDULE\_TIMEOUT);

+}

+EXPORT\_SYMBOL(interruptible\_sleep\_on);

+

+long \_\_sched

+interruptible\_sleep\_on\_timeout(wait\_queue\_head\_t \*q, long timeout)

+{

+ return sleep\_on\_common(q, TASK\_INTERRUPTIBLE, timeout);

+}

+EXPORT\_SYMBOL(interruptible\_sleep\_on\_timeout);

+

+void \_\_sched sleep\_on(wait\_queue\_head\_t \*q)

+{

+ sleep\_on\_common(q, TASK\_UNINTERRUPTIBLE, MAX\_SCHEDULE\_TIMEOUT);

+}

+EXPORT\_SYMBOL(sleep\_on);

+

+long \_\_sched sleep\_on\_timeout(wait\_queue\_head\_t \*q, long timeout)

+{

+ return sleep\_on\_common(q, TASK\_UNINTERRUPTIBLE, timeout);

+}

+EXPORT\_SYMBOL(sleep\_on\_timeout);

+

+#ifdef CONFIG\_RT\_MUTEXES

+

+/\*

+ \* rt\_mutex\_setprio - set the current priority of a task

+ \* @p: task

+ \* @prio: prio value (kernel-internal form)

+ \*

+ \* This function changes the 'effective' priority of a task. It does

+ \* not touch ->normal\_prio like \_\_setscheduler().

+ \*

+ \* Used by the rt\_mutex code to implement priority inheritance logic.

+ \*/

+void rt\_mutex\_setprio(struct task\_struct \*p, int prio)

+{

+ unsigned long flags;

+ int oldprio, on\_rq, running;

+ struct rq \*rq;

+

+ BUG\_ON(prio < 0 || prio > MAX\_PRIO);

+

+ rq = task\_rq\_lock(p, &flags);

+ update\_rq\_clock(rq);

+

+ oldprio = p->prio;

+ on\_rq = p->se.on\_rq;

+ running = task\_current(rq, p);

+ if (on\_rq) {

+ dequeue\_task(rq, p, 0);

+ if (running)

+ p->sched\_class->put\_prev\_task(rq, p);

+ }

+

+ if (rt\_prio(prio))

+ p->sched\_class = &rt\_sched\_class;

+ else

+ p->sched\_class = &fair\_sched\_class;

+

+ p->prio = prio;

+

+ if (on\_rq) {

+ if (running)

+ p->sched\_class->set\_curr\_task(rq);

+ enqueue\_task(rq, p, 0);

+ /\*

+ \* Reschedule if we are currently running on this runqueue and

+ \* our priority decreased, or if we are not currently running on

+ \* this runqueue and our priority is higher than the current's

+ \*/

+ if (running) {

+ if (p->prio > oldprio)

+ resched\_task(rq->curr);

+ } else {

+ check\_preempt\_curr(rq, p);

+ }

+ }

+ task\_rq\_unlock(rq, &flags);

+}

+

+#endif

+

+void set\_user\_nice(struct task\_struct \*p, long nice)

+{

+ int old\_prio, delta, on\_rq;

+ unsigned long flags;

+ struct rq \*rq;

+

+ if (TASK\_NICE(p) == nice || nice < -20 || nice > 19)

+ return;

+ /\*

+ \* We have to be careful, if called from sys\_setpriority(),

+ \* the task might be in the middle of scheduling on another CPU.

+ \*/

+ rq = task\_rq\_lock(p, &flags);

+ update\_rq\_clock(rq);

+ /\*

+ \* The RT priorities are set via sched\_setscheduler(), but we still

+ \* allow the 'normal' nice value to be set - but as expected

+ \* it wont have any effect on scheduling until the task is

+ \* SCHED\_FIFO/SCHED\_RR:

+ \*/

+ if (task\_has\_rt\_policy(p)) {

+ p->static\_prio = NICE\_TO\_PRIO(nice);

+ goto out\_unlock;

+ }

+ on\_rq = p->se.on\_rq;

+ if (on\_rq) {

+ dequeue\_task(rq, p, 0);

+ dec\_load(rq, p);

+ }

+

+ p->static\_prio = NICE\_TO\_PRIO(nice);

+ set\_load\_weight(p);

+ old\_prio = p->prio;

+ p->prio = effective\_prio(p);

+ delta = p->prio - old\_prio;

+

+ if (on\_rq) {

+ enqueue\_task(rq, p, 0);

+ inc\_load(rq, p);

+ /\*

+ \* If the task increased its priority or is running and

+ \* lowered its priority, then reschedule its CPU:

+ \*/

+ if (delta < 0 || (delta > 0 && task\_running(rq, p)))

+ resched\_task(rq->curr);

+ }

+out\_unlock:

+ task\_rq\_unlock(rq, &flags);

+}

+EXPORT\_SYMBOL(set\_user\_nice);

+

+/\*

+ \* can\_nice - check if a task can reduce its nice value

+ \* @p: task

+ \* @nice: nice value

+ \*/

+int can\_nice(const struct task\_struct \*p, const int nice)

+{

+ /\* convert nice value [19,-20] to rlimit style value [1,40] \*/

+ int nice\_rlim = 20 - nice;

+

+ return (nice\_rlim <= p->signal->rlim[RLIMIT\_NICE].rlim\_cur ||

+ capable(CAP\_SYS\_NICE));

+}

+

+#ifdef \_\_ARCH\_WANT\_SYS\_NICE

+

+/\*

+ \* sys\_nice - change the priority of the current process.

+ \* @increment: priority increment

+ \*

+ \* sys\_setpriority is a more generic, but much slower function that

+ \* does similar things.

+ \*/

+asmlinkage long sys\_nice(int increment)

+{

+ long nice, retval;

+

+ /\*

+ \* Setpriority might change our priority at the same moment.

+ \* We don't have to worry. Conceptually one call occurs first

+ \* and we have a single winner.

+ \*/

+ if (increment < -40)

+ increment = -40;

+ if (increment > 40)

+ increment = 40;

+

+ nice = PRIO\_TO\_NICE(current->static\_prio) + increment;

+ if (nice < -20)

+ nice = -20;

+ if (nice > 19)

+ nice = 19;

+

+ if (increment < 0 && !can\_nice(current, nice))

+ return -EPERM;

+

+ retval = security\_task\_setnice(current, nice);

+ if (retval)

+ return retval;

+

+ set\_user\_nice(current, nice);

+ return 0;

+}

+

+#endif

+

+/\*\*

+ \* task\_prio - return the priority value of a given task.

+ \* @p: the task in question.

+ \*

+ \* This is the priority value as seen by users in /proc.

+ \* RT tasks are offset by -200. Normal tasks are centered

+ \* around 0, value goes from -16 to +15.

+ \*/

+int task\_prio(const struct task\_struct \*p)

+{

+ return p->prio - MAX\_RT\_PRIO;

+}

+

+/\*\*

+ \* task\_nice - return the nice value of a given task.

+ \* @p: the task in question.

+ \*/

+int task\_nice(const struct task\_struct \*p)

+{

+ return TASK\_NICE(p);

+}

+EXPORT\_SYMBOL\_GPL(task\_nice);

+

+/\*\*

+ \* idle\_cpu - is a given cpu idle currently?

+ \* @cpu: the processor in question.

+ \*/

+int idle\_cpu(int cpu)

+{

+ return cpu\_curr(cpu) == cpu\_rq(cpu)->idle;

+}

+

+/\*\*

+ \* idle\_task - return the idle task for a given cpu.

+ \* @cpu: the processor in question.

+ \*/

+struct task\_struct \*idle\_task(int cpu)

+{

+ return cpu\_rq(cpu)->idle;

+}

+

+/\*\*

+ \* find\_process\_by\_pid - find a process with a matching PID value.

+ \* @pid: the pid in question.

+ \*/

+static struct task\_struct \*find\_process\_by\_pid(pid\_t pid)

+{

+ return pid ? find\_task\_by\_vpid(pid) : current;

+}

+

+/\* Actually do priority change: must hold rq lock. \*/

+static void

+\_\_setscheduler(struct rq \*rq, struct task\_struct \*p, int policy, int prio)

+{

+ BUG\_ON(p->se.on\_rq);

+

+ p->policy = policy;

+ switch (p->policy) {

+ case SCHED\_NORMAL:

+ case SCHED\_BATCH:

+ case SCHED\_IDLE:

+ p->sched\_class = &fair\_sched\_class;

+ break;

+ case SCHED\_FIFO:

+ case SCHED\_RR:

+ p->sched\_class = &rt\_sched\_class;

+ break;

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ case SCHED\_CASIO:

+ p->sched\_class = &casio\_sched\_class;

+ break;

+#endif

+ }

+

+ p->rt\_priority = prio;

+ p->normal\_prio = normal\_prio(p);

+ /\* we are holding p->pi\_lock already \*/

+ p->prio = rt\_mutex\_getprio(p);

+ set\_load\_weight(p);

+}

+

+/\*\*

+ \* sched\_setscheduler - change the scheduling policy and/or RT priority of a thread.

+ \* @p: the task in question.

+ \* @policy: new policy.

+ \* @param: structure containing the new RT priority.

+ \*

+ \* NOTE that the task may be already dead.

+ \*/

+int sched\_setscheduler(struct task\_struct \*p, int policy,

+ struct sched\_param \*param)

+{

+ int retval, oldprio, oldpolicy = -1, on\_rq, running;

+ unsigned long flags;

+ struct rq \*rq;

+ /\* may grab non-irq protected spin\_locks \*/

+ BUG\_ON(in\_interrupt());

+recheck:

+ /\* double check policy once rq lock held \*/

+ if (policy < 0)

+ policy = oldpolicy = p->policy;

+ else if (policy != SCHED\_FIFO && policy != SCHED\_RR &&

+ policy != SCHED\_NORMAL && policy != SCHED\_BATCH &&

+ policy != SCHED\_IDLE

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ && policy!=SCHED\_CASIO

+#endif

+ )

+ return -EINVAL;

+ /\*

+ \* Valid priorities for SCHED\_FIFO and SCHED\_RR are

+ \* 1..MAX\_USER\_RT\_PRIO-1, valid priority for SCHED\_NORMAL,

+ \* SCHED\_BATCH and SCHED\_IDLE is 0.

+ \*/

+ if (param->sched\_priority < 0 ||

+ (p->mm && param->sched\_priority > MAX\_USER\_RT\_PRIO-1) ||

+ (!p->mm && param->sched\_priority > MAX\_RT\_PRIO-1))

+ return -EINVAL;

+ if (rt\_policy(policy) != (param->sched\_priority != 0))

+ return -EINVAL;

+

+ /\*

+ \* Allow unprivileged RT tasks to decrease priority:

+ \*/

+ if (!capable(CAP\_SYS\_NICE)) {

+ if (rt\_policy(policy)) {

+ unsigned long rlim\_rtprio;

+

+ if (!lock\_task\_sighand(p, &flags))

+ return -ESRCH;

+ rlim\_rtprio = p->signal->rlim[RLIMIT\_RTPRIO].rlim\_cur;

+ unlock\_task\_sighand(p, &flags);

+

+ /\* can't set/change the rt policy \*/

+ if (policy != p->policy && !rlim\_rtprio)

+ return -EPERM;

+

+ /\* can't increase priority \*/

+ if (param->sched\_priority > p->rt\_priority &&

+ param->sched\_priority > rlim\_rtprio)

+ return -EPERM;

+ }

+ /\*

+ \* Like positive nice levels, dont allow tasks to

+ \* move out of SCHED\_IDLE either:

+ \*/

+ if (p->policy == SCHED\_IDLE && policy != SCHED\_IDLE)

+ return -EPERM;

+

+ /\* can't change other user's priorities \*/

+ if ((current->euid != p->euid) &&

+ (current->euid != p->uid))

+ return -EPERM;

+ }

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ if(policy==SCHED\_CASIO){

+ p->deadline=param->deadline;

+ p->casio\_id=param->casio\_id;

+ }

+#endif

+

+ retval = security\_task\_setscheduler(p, policy, param);

+ if (retval)

+ return retval;

+ /\*

+ \* make sure no PI-waiters arrive (or leave) while we are

+ \* changing the priority of the task:

+ \*/

+ spin\_lock\_irqsave(&p->pi\_lock, flags);

+ /\*

+ \* To be able to change p->policy safely, the apropriate

+ \* runqueue lock must be held.

+ \*/

+ rq = \_\_task\_rq\_lock(p);

+ /\* recheck policy now with rq lock held \*/

+ if (unlikely(oldpolicy != -1 && oldpolicy != p->policy)) {

+ policy = oldpolicy = -1;

+ \_\_task\_rq\_unlock(rq);

+ spin\_unlock\_irqrestore(&p->pi\_lock, flags);

+ goto recheck;

+ }

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ if(policy==SCHED\_CASIO){

+ add\_casio\_task\_2\_list(&rq->casio\_rq, p);

+ }

+#endif

+ update\_rq\_clock(rq);

+ on\_rq = p->se.on\_rq;

+ running = task\_current(rq, p);

+ if (on\_rq) {

+ deactivate\_task(rq, p, 0);

+ if (running)

+ p->sched\_class->put\_prev\_task(rq, p);

+ }

+

+ oldprio = p->prio;

+ \_\_setscheduler(rq, p, policy, param->sched\_priority);

+

+ if (on\_rq) {

+ if (running)

+ p->sched\_class->set\_curr\_task(rq);

+ activate\_task(rq, p, 0);

+ /\*

+ \* Reschedule if we are currently running on this runqueue and

+ \* our priority decreased, or if we are not currently running on

+ \* this runqueue and our priority is higher than the current's

+ \*/

+ if (running) {

+ if (p->prio > oldprio)

+ resched\_task(rq->curr);

+ } else {

+ check\_preempt\_curr(rq, p);

+ }

+ }

+ \_\_task\_rq\_unlock(rq);

+ spin\_unlock\_irqrestore(&p->pi\_lock, flags);

+

+ rt\_mutex\_adjust\_pi(p);

+

+ return 0;

+}

+EXPORT\_SYMBOL\_GPL(sched\_setscheduler);

+

+static int

+do\_sched\_setscheduler(pid\_t pid, int policy, struct sched\_param \_\_user \*param)

+{

+ struct sched\_param lparam;

+ struct task\_struct \*p;

+ int retval;

+

+ if (!param || pid < 0)

+ return -EINVAL;

+ if (copy\_from\_user(&lparam, param, sizeof(struct sched\_param)))

+ return -EFAULT;

+

+ rcu\_read\_lock();

+ retval = -ESRCH;

+ p = find\_process\_by\_pid(pid);

+ if (p != NULL)

+ retval = sched\_setscheduler(p, policy, &lparam);

+ rcu\_read\_unlock();

+

+ return retval;

+}

+

+/\*\*

+ \* sys\_sched\_setscheduler - set/change the scheduler policy and RT priority

+ \* @pid: the pid in question.

+ \* @policy: new policy.

+ \* @param: structure containing the new RT priority.

+ \*/

+asmlinkage long

+sys\_sched\_setscheduler(pid\_t pid, int policy, struct sched\_param \_\_user \*param)

+{

+ /\* negative values for policy are not valid \*/

+ if (policy < 0)

+ return -EINVAL;

+

+ return do\_sched\_setscheduler(pid, policy, param);

+}

+

+/\*\*

+ \* sys\_sched\_setparam - set/change the RT priority of a thread

+ \* @pid: the pid in question.

+ \* @param: structure containing the new RT priority.

+ \*/

+asmlinkage long sys\_sched\_setparam(pid\_t pid, struct sched\_param \_\_user \*param)

+{

+ return do\_sched\_setscheduler(pid, -1, param);

+}

+

+/\*\*

+ \* sys\_sched\_getscheduler - get the policy (scheduling class) of a thread

+ \* @pid: the pid in question.

+ \*/

+asmlinkage long sys\_sched\_getscheduler(pid\_t pid)

+{

+ struct task\_struct \*p;

+ int retval;

+

+ if (pid < 0)

+ return -EINVAL;

+

+ retval = -ESRCH;

+ read\_lock(&tasklist\_lock);

+ p = find\_process\_by\_pid(pid);

+ if (p) {

+ retval = security\_task\_getscheduler(p);

+ if (!retval)

+ retval = p->policy;

+ }

+ read\_unlock(&tasklist\_lock);

+ return retval;

+}

+

+/\*\*

+ \* sys\_sched\_getscheduler - get the RT priority of a thread

+ \* @pid: the pid in question.

+ \* @param: structure containing the RT priority.

+ \*/

+asmlinkage long sys\_sched\_getparam(pid\_t pid, struct sched\_param \_\_user \*param)

+{

+ struct sched\_param lp;

+ struct task\_struct \*p;

+ int retval;

+

+ if (!param || pid < 0)

+ return -EINVAL;

+

+ read\_lock(&tasklist\_lock);

+ p = find\_process\_by\_pid(pid);

+ retval = -ESRCH;

+ if (!p)

+ goto out\_unlock;

+

+ retval = security\_task\_getscheduler(p);

+ if (retval)

+ goto out\_unlock;

+

+ lp.sched\_priority = p->rt\_priority;

+ read\_unlock(&tasklist\_lock);

+

+ /\*

+ \* This one might sleep, we cannot do it with a spinlock held ...

+ \*/

+ retval = copy\_to\_user(param, &lp, sizeof(\*param)) ? -EFAULT : 0;

+

+ return retval;

+

+out\_unlock:

+ read\_unlock(&tasklist\_lock);

+ return retval;

+}

+

+long sched\_setaffinity(pid\_t pid, cpumask\_t new\_mask)

+{

+ cpumask\_t cpus\_allowed;

+ struct task\_struct \*p;

+ int retval;

+

+ mutex\_lock(&sched\_hotcpu\_mutex);

+ read\_lock(&tasklist\_lock);

+

+ p = find\_process\_by\_pid(pid);

+ if (!p) {

+ read\_unlock(&tasklist\_lock);

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+ return -ESRCH;

+ }

+

+ /\*

+ \* It is not safe to call set\_cpus\_allowed with the

+ \* tasklist\_lock held. We will bump the task\_struct's

+ \* usage count and then drop tasklist\_lock.

+ \*/

+ get\_task\_struct(p);

+ read\_unlock(&tasklist\_lock);

+

+ retval = -EPERM;

+ if ((current->euid != p->euid) && (current->euid != p->uid) &&

+ !capable(CAP\_SYS\_NICE))

+ goto out\_unlock;

+

+ retval = security\_task\_setscheduler(p, 0, NULL);

+ if (retval)

+ goto out\_unlock;

+

+ cpus\_allowed = cpuset\_cpus\_allowed(p);

+ cpus\_and(new\_mask, new\_mask, cpus\_allowed);

+ again:

+ retval = set\_cpus\_allowed(p, new\_mask);

+

+ if (!retval) {

+ cpus\_allowed = cpuset\_cpus\_allowed(p);

+ if (!cpus\_subset(new\_mask, cpus\_allowed)) {

+ /\*

+ \* We must have raced with a concurrent cpuset

+ \* update. Just reset the cpus\_allowed to the

+ \* cpuset's cpus\_allowed

+ \*/

+ new\_mask = cpus\_allowed;

+ goto again;

+ }

+ }

+out\_unlock:

+ put\_task\_struct(p);

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+ return retval;

+}

+

+static int get\_user\_cpu\_mask(unsigned long \_\_user \*user\_mask\_ptr, unsigned len,

+ cpumask\_t \*new\_mask)

+{

+ if (len < sizeof(cpumask\_t)) {

+ memset(new\_mask, 0, sizeof(cpumask\_t));

+ } else if (len > sizeof(cpumask\_t)) {

+ len = sizeof(cpumask\_t);

+ }

+ return copy\_from\_user(new\_mask, user\_mask\_ptr, len) ? -EFAULT : 0;

+}

+

+/\*\*

+ \* sys\_sched\_setaffinity - set the cpu affinity of a process

+ \* @pid: pid of the process

+ \* @len: length in bytes of the bitmask pointed to by user\_mask\_ptr

+ \* @user\_mask\_ptr: user-space pointer to the new cpu mask

+ \*/

+asmlinkage long sys\_sched\_setaffinity(pid\_t pid, unsigned int len,

+ unsigned long \_\_user \*user\_mask\_ptr)

+{

+ cpumask\_t new\_mask;

+ int retval;

+

+ retval = get\_user\_cpu\_mask(user\_mask\_ptr, len, &new\_mask);

+ if (retval)

+ return retval;

+

+ return sched\_setaffinity(pid, new\_mask);

+}

+

+/\*

+ \* Represents all cpu's present in the system

+ \* In systems capable of hotplug, this map could dynamically grow

+ \* as new cpu's are detected in the system via any platform specific

+ \* method, such as ACPI for e.g.

+ \*/

+

+cpumask\_t cpu\_present\_map \_\_read\_mostly;

+EXPORT\_SYMBOL(cpu\_present\_map);

+

+#ifndef CONFIG\_SMP

+cpumask\_t cpu\_online\_map \_\_read\_mostly = CPU\_MASK\_ALL;

+EXPORT\_SYMBOL(cpu\_online\_map);

+

+cpumask\_t cpu\_possible\_map \_\_read\_mostly = CPU\_MASK\_ALL;

+EXPORT\_SYMBOL(cpu\_possible\_map);

+#endif

+

+long sched\_getaffinity(pid\_t pid, cpumask\_t \*mask)

+{

+ struct task\_struct \*p;

+ int retval;

+

+ mutex\_lock(&sched\_hotcpu\_mutex);

+ read\_lock(&tasklist\_lock);

+

+ retval = -ESRCH;

+ p = find\_process\_by\_pid(pid);

+ if (!p)

+ goto out\_unlock;

+

+ retval = security\_task\_getscheduler(p);

+ if (retval)

+ goto out\_unlock;

+

+ cpus\_and(\*mask, p->cpus\_allowed, cpu\_online\_map);

+

+out\_unlock:

+ read\_unlock(&tasklist\_lock);

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+

+ return retval;

+}

+

+/\*\*

+ \* sys\_sched\_getaffinity - get the cpu affinity of a process

+ \* @pid: pid of the process

+ \* @len: length in bytes of the bitmask pointed to by user\_mask\_ptr

+ \* @user\_mask\_ptr: user-space pointer to hold the current cpu mask

+ \*/

+asmlinkage long sys\_sched\_getaffinity(pid\_t pid, unsigned int len,

+ unsigned long \_\_user \*user\_mask\_ptr)

+{

+ int ret;

+ cpumask\_t mask;

+

+ if (len < sizeof(cpumask\_t))

+ return -EINVAL;

+

+ ret = sched\_getaffinity(pid, &mask);

+ if (ret < 0)

+ return ret;

+

+ if (copy\_to\_user(user\_mask\_ptr, &mask, sizeof(cpumask\_t)))

+ return -EFAULT;

+

+ return sizeof(cpumask\_t);

+}

+

+/\*\*

+ \* sys\_sched\_yield - yield the current processor to other threads.

+ \*

+ \* This function yields the current CPU to other tasks. If there are no

+ \* other threads running on this CPU then this function will return.

+ \*/

+asmlinkage long sys\_sched\_yield(void)

+{

+ struct rq \*rq = this\_rq\_lock();

+

+ schedstat\_inc(rq, yld\_count);

+ current->sched\_class->yield\_task(rq);

+

+ /\*

+ \* Since we are going to call schedule() anyway, there's

+ \* no need to preempt or enable interrupts:

+ \*/

+ \_\_release(rq->lock);

+ spin\_release(&rq->lock.dep\_map, 1, \_THIS\_IP\_);

+ \_raw\_spin\_unlock(&rq->lock);

+ preempt\_enable\_no\_resched();

+

+ schedule();

+

+ return 0;

+}

+

+static void \_\_cond\_resched(void)

+{

+#ifdef CONFIG\_DEBUG\_SPINLOCK\_SLEEP

+ \_\_might\_sleep(\_\_FILE\_\_, \_\_LINE\_\_);

+#endif

+ /\*

+ \* The BKS might be reacquired before we have dropped

+ \* PREEMPT\_ACTIVE, which could trigger a second

+ \* cond\_resched() call.

+ \*/

+ do {

+ add\_preempt\_count(PREEMPT\_ACTIVE);

+ schedule();

+ sub\_preempt\_count(PREEMPT\_ACTIVE);

+ } while (need\_resched());

+}

+

+int \_\_sched cond\_resched(void)

+{

+ if (need\_resched() && !(preempt\_count() & PREEMPT\_ACTIVE) &&

+ system\_state == SYSTEM\_RUNNING) {

+ \_\_cond\_resched();

+ return 1;

+ }

+ return 0;

+}

+EXPORT\_SYMBOL(cond\_resched);

+

+/\*

+ \* cond\_resched\_lock() - if a reschedule is pending, drop the given lock,

+ \* call schedule, and on return reacquire the lock.

+ \*

+ \* This works OK both with and without CONFIG\_PREEMPT. We do strange low-level

+ \* operations here to prevent schedule() from being called twice (once via

+ \* spin\_unlock(), once by hand).

+ \*/

+int cond\_resched\_lock(spinlock\_t \*lock)

+{

+ int ret = 0;

+

+ if (need\_lockbreak(lock)) {

+ spin\_unlock(lock);

+ cpu\_relax();

+ ret = 1;

+ spin\_lock(lock);

+ }

+ if (need\_resched() && system\_state == SYSTEM\_RUNNING) {

+ spin\_release(&lock->dep\_map, 1, \_THIS\_IP\_);

+ \_raw\_spin\_unlock(lock);

+ preempt\_enable\_no\_resched();

+ \_\_cond\_resched();

+ ret = 1;

+ spin\_lock(lock);

+ }

+ return ret;

+}

+EXPORT\_SYMBOL(cond\_resched\_lock);

+

+int \_\_sched cond\_resched\_softirq(void)

+{

+ BUG\_ON(!in\_softirq());

+

+ if (need\_resched() && system\_state == SYSTEM\_RUNNING) {

+ local\_bh\_enable();

+ \_\_cond\_resched();

+ local\_bh\_disable();

+ return 1;

+ }

+ return 0;

+}

+EXPORT\_SYMBOL(cond\_resched\_softirq);

+

+/\*\*

+ \* yield - yield the current processor to other threads.

+ \*

+ \* This is a shortcut for kernel-space yielding - it marks the

+ \* thread runnable and calls sys\_sched\_yield().

+ \*/

+void \_\_sched yield(void)

+{

+ set\_current\_state(TASK\_RUNNING);

+ sys\_sched\_yield();

+}

+EXPORT\_SYMBOL(yield);

+

+/\*

+ \* This task is about to go to sleep on IO. Increment rq->nr\_iowait so

+ \* that process accounting knows that this is a task in IO wait state.

+ \*

+ \* But don't do that if it is a deliberate, throttling IO wait (this task

+ \* has set its backing\_dev\_info: the queue against which it should throttle)

+ \*/

+void \_\_sched io\_schedule(void)

+{

+ struct rq \*rq = &\_\_raw\_get\_cpu\_var(runqueues);

+

+ delayacct\_blkio\_start();

+ atomic\_inc(&rq->nr\_iowait);

+ schedule();

+ atomic\_dec(&rq->nr\_iowait);

+ delayacct\_blkio\_end();

+}

+EXPORT\_SYMBOL(io\_schedule);

+

+long \_\_sched io\_schedule\_timeout(long timeout)

+{

+ struct rq \*rq = &\_\_raw\_get\_cpu\_var(runqueues);

+ long ret;

+

+ delayacct\_blkio\_start();

+ atomic\_inc(&rq->nr\_iowait);

+ ret = schedule\_timeout(timeout);

+ atomic\_dec(&rq->nr\_iowait);

+ delayacct\_blkio\_end();

+ return ret;

+}

+

+/\*\*

+ \* sys\_sched\_get\_priority\_max - return maximum RT priority.

+ \* @policy: scheduling class.

+ \*

+ \* this syscall returns the maximum rt\_priority that can be used

+ \* by a given scheduling class.

+ \*/

+asmlinkage long sys\_sched\_get\_priority\_max(int policy)

+{

+ int ret = -EINVAL;

+

+ switch (policy) {

+ case SCHED\_FIFO:

+ case SCHED\_RR:

+ ret = MAX\_USER\_RT\_PRIO-1;

+ break;

+ case SCHED\_NORMAL:

+ case SCHED\_BATCH:

+ case SCHED\_IDLE:

+ ret = 0;

+ break;

+ }

+ return ret;

+}

+

+/\*\*

+ \* sys\_sched\_get\_priority\_min - return minimum RT priority.

+ \* @policy: scheduling class.

+ \*

+ \* this syscall returns the minimum rt\_priority that can be used

+ \* by a given scheduling class.

+ \*/

+asmlinkage long sys\_sched\_get\_priority\_min(int policy)

+{

+ int ret = -EINVAL;

+

+ switch (policy) {

+ case SCHED\_FIFO:

+ case SCHED\_RR:

+ ret = 1;

+ break;

+ case SCHED\_NORMAL:

+ case SCHED\_BATCH:

+ case SCHED\_IDLE:

+ ret = 0;

+ }

+ return ret;

+}

+

+/\*\*

+ \* sys\_sched\_rr\_get\_interval - return the default timeslice of a process.

+ \* @pid: pid of the process.

+ \* @interval: userspace pointer to the timeslice value.

+ \*

+ \* this syscall writes the default timeslice value of a given process

+ \* into the user-space timespec buffer. A value of '0' means infinity.

+ \*/

+asmlinkage

+long sys\_sched\_rr\_get\_interval(pid\_t pid, struct timespec \_\_user \*interval)

+{

+ struct task\_struct \*p;

+ unsigned int time\_slice;

+ int retval;

+ struct timespec t;

+

+ if (pid < 0)

+ return -EINVAL;

+

+ retval = -ESRCH;

+ read\_lock(&tasklist\_lock);

+ p = find\_process\_by\_pid(pid);

+ if (!p)

+ goto out\_unlock;

+

+ retval = security\_task\_getscheduler(p);

+ if (retval)

+ goto out\_unlock;

+

+ /\*

+ \* Time slice is 0 for SCHED\_FIFO tasks and for SCHED\_OTHER

+ \* tasks that are on an otherwise idle runqueue:

+ \*/

+ time\_slice = 0;

+ if (p->policy == SCHED\_RR) {

+ time\_slice = DEF\_TIMESLICE;

+ } else {

+ struct sched\_entity \*se = &p->se;

+ unsigned long flags;

+ struct rq \*rq;

+

+ rq = task\_rq\_lock(p, &flags);

+ if (rq->cfs.load.weight)

+ time\_slice = NS\_TO\_JIFFIES(sched\_slice(&rq->cfs, se));

+ task\_rq\_unlock(rq, &flags);

+ }

+ read\_unlock(&tasklist\_lock);

+ jiffies\_to\_timespec(time\_slice, &t);

+ retval = copy\_to\_user(interval, &t, sizeof(t)) ? -EFAULT : 0;

+ return retval;

+

+out\_unlock:

+ read\_unlock(&tasklist\_lock);

+ return retval;

+}

+

+static const char stat\_nam[] = "RSDTtZX";

+

+static void show\_task(struct task\_struct \*p)

+{

+ unsigned long free = 0;

+ unsigned state;

+

+ state = p->state ? \_\_ffs(p->state) + 1 : 0;

+ printk(KERN\_INFO "%-13.13s %c", p->comm,

+ state < sizeof(stat\_nam) - 1 ? stat\_nam[state] : '?');

+#if BITS\_PER\_LONG == 32

+ if (state == TASK\_RUNNING)

+ printk(KERN\_CONT " running ");

+ else

+ printk(KERN\_CONT " %08lx ", thread\_saved\_pc(p));

+#else

+ if (state == TASK\_RUNNING)

+ printk(KERN\_CONT " running task ");

+ else

+ printk(KERN\_CONT " %016lx ", thread\_saved\_pc(p));

+#endif

+#ifdef CONFIG\_DEBUG\_STACK\_USAGE

+ {

+ unsigned long \*n = end\_of\_stack(p);

+ while (!\*n)

+ n++;

+ free = (unsigned long)n - (unsigned long)end\_of\_stack(p);

+ }

+#endif

+ printk(KERN\_CONT "%5lu %5d %6d\n", free,

+ task\_pid\_nr(p), task\_pid\_nr(p->real\_parent));

+

+ if (state != TASK\_RUNNING)

+ show\_stack(p, NULL);

+}

+

+void show\_state\_filter(unsigned long state\_filter)

+{

+ struct task\_struct \*g, \*p;

+

+#if BITS\_PER\_LONG == 32

+ printk(KERN\_INFO

+ " task PC stack pid father\n");

+#else

+ printk(KERN\_INFO

+ " task PC stack pid father\n");

+#endif

+ read\_lock(&tasklist\_lock);

+ do\_each\_thread(g, p) {

+ /\*

+ \* reset the NMI-timeout, listing all files on a slow

+ \* console might take alot of time:

+ \*/

+ touch\_nmi\_watchdog();

+ if (!state\_filter || (p->state & state\_filter))

+ show\_task(p);

+ } while\_each\_thread(g, p);

+

+ touch\_all\_softlockup\_watchdogs();

+

+#ifdef CONFIG\_SCHED\_DEBUG

+ sysrq\_sched\_debug\_show();

+#endif

+ read\_unlock(&tasklist\_lock);

+ /\*

+ \* Only show locks if all tasks are dumped:

+ \*/

+ if (state\_filter == -1)

+ debug\_show\_all\_locks();

+}

+

+void \_\_cpuinit init\_idle\_bootup\_task(struct task\_struct \*idle)

+{

+ idle->sched\_class = &idle\_sched\_class;

+}

+

+/\*\*

+ \* init\_idle - set up an idle thread for a given CPU

+ \* @idle: task in question

+ \* @cpu: cpu the idle task belongs to

+ \*

+ \* NOTE: this function does not set the idle thread's NEED\_RESCHED

+ \* flag, to make booting more robust.

+ \*/

+void \_\_cpuinit init\_idle(struct task\_struct \*idle, int cpu)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ unsigned long flags;

+

+ \_\_sched\_fork(idle);

+ idle->se.exec\_start = sched\_clock();

+

+ idle->prio = idle->normal\_prio = MAX\_PRIO;

+ idle->cpus\_allowed = cpumask\_of\_cpu(cpu);

+ \_\_set\_task\_cpu(idle, cpu);

+

+ spin\_lock\_irqsave(&rq->lock, flags);

+ rq->curr = rq->idle = idle;

+#if defined(CONFIG\_SMP) && defined(\_\_ARCH\_WANT\_UNLOCKED\_CTXSW)

+ idle->oncpu = 1;

+#endif

+ spin\_unlock\_irqrestore(&rq->lock, flags);

+

+ /\* Set the preempt count \_outside\_ the spinlocks! \*/

+#if defined(CONFIG\_PREEMPT) && !defined(CONFIG\_PREEMPT\_BKL)

+ task\_thread\_info(idle)->preempt\_count = (idle->lock\_depth >= 0);

+#else

+ task\_thread\_info(idle)->preempt\_count = 0;

+#endif

+ /\*

+ \* The idle tasks have their own, simple scheduling class:

+ \*/

+ idle->sched\_class = &idle\_sched\_class;

+}

+

+/\*

+ \* In a system that switches off the HZ timer nohz\_cpu\_mask

+ \* indicates which cpus entered this state. This is used

+ \* in the rcu update to wait only for active cpus. For system

+ \* which do not switch off the HZ timer nohz\_cpu\_mask should

+ \* always be CPU\_MASK\_NONE.

+ \*/

+cpumask\_t nohz\_cpu\_mask = CPU\_MASK\_NONE;

+

+/\*

+ \* Increase the granularity value when there are more CPUs,

+ \* because with more CPUs the 'effective latency' as visible

+ \* to users decreases. But the relationship is not linear,

+ \* so pick a second-best guess by going with the log2 of the

+ \* number of CPUs.

+ \*

+ \* This idea comes from the SD scheduler of Con Kolivas:

+ \*/

+static inline void sched\_init\_granularity(void)

+{

+ unsigned int factor = 1 + ilog2(num\_online\_cpus());

+ const unsigned long limit = 200000000;

+

+ sysctl\_sched\_min\_granularity \*= factor;

+ if (sysctl\_sched\_min\_granularity > limit)

+ sysctl\_sched\_min\_granularity = limit;

+

+ sysctl\_sched\_latency \*= factor;

+ if (sysctl\_sched\_latency > limit)

+ sysctl\_sched\_latency = limit;

+

+ sysctl\_sched\_wakeup\_granularity \*= factor;

+ sysctl\_sched\_batch\_wakeup\_granularity \*= factor;

+}

+

+#ifdef CONFIG\_SMP

+/\*

+ \* This is how migration works:

+ \*

+ \* 1) we queue a struct migration\_req structure in the source CPU's

+ \* runqueue and wake up that CPU's migration thread.

+ \* 2) we down() the locked semaphore => thread blocks.

+ \* 3) migration thread wakes up (implicitly it forces the migrated

+ \* thread off the CPU)

+ \* 4) it gets the migration request and checks whether the migrated

+ \* task is still in the wrong runqueue.

+ \* 5) if it's in the wrong runqueue then the migration thread removes

+ \* it and puts it into the right queue.

+ \* 6) migration thread up()s the semaphore.

+ \* 7) we wake up and the migration is done.

+ \*/

+

+/\*

+ \* Change a given task's CPU affinity. Migrate the thread to a

+ \* proper CPU and schedule it away if the CPU it's executing on

+ \* is removed from the allowed bitmask.

+ \*

+ \* NOTE: the caller must have a valid reference to the task, the

+ \* task must not exit() & deallocate itself prematurely. The

+ \* call is not atomic; no spinlocks may be held.

+ \*/

+int set\_cpus\_allowed(struct task\_struct \*p, cpumask\_t new\_mask)

+{

+ struct migration\_req req;

+ unsigned long flags;

+ struct rq \*rq;

+ int ret = 0;

+

+ rq = task\_rq\_lock(p, &flags);

+ if (!cpus\_intersects(new\_mask, cpu\_online\_map)) {

+ ret = -EINVAL;

+ goto out;

+ }

+

+ p->cpus\_allowed = new\_mask;

+ /\* Can the task run on the task's current CPU? If so, we're done \*/

+ if (cpu\_isset(task\_cpu(p), new\_mask))

+ goto out;

+

+ if (migrate\_task(p, any\_online\_cpu(new\_mask), &req)) {

+ /\* Need help from migration thread: drop lock and wait. \*/

+ task\_rq\_unlock(rq, &flags);

+ wake\_up\_process(rq->migration\_thread);

+ wait\_for\_completion(&req.done);

+ tlb\_migrate\_finish(p->mm);

+ return 0;

+ }

+out:

+ task\_rq\_unlock(rq, &flags);

+

+ return ret;

+}

+EXPORT\_SYMBOL\_GPL(set\_cpus\_allowed);

+

+/\*

+ \* Move (not current) task off this cpu, onto dest cpu. We're doing

+ \* this because either it can't run here any more (set\_cpus\_allowed()

+ \* away from this CPU, or CPU going down), or because we're

+ \* attempting to rebalance this task on exec (sched\_exec).

+ \*

+ \* So we race with normal scheduler movements, but that's OK, as long

+ \* as the task is no longer on this CPU.

+ \*

+ \* Returns non-zero if task was successfully migrated.

+ \*/

+static int \_\_migrate\_task(struct task\_struct \*p, int src\_cpu, int dest\_cpu)

+{

+ struct rq \*rq\_dest, \*rq\_src;

+ int ret = 0, on\_rq;

+

+ if (unlikely(cpu\_is\_offline(dest\_cpu)))

+ return ret;

+

+ rq\_src = cpu\_rq(src\_cpu);

+ rq\_dest = cpu\_rq(dest\_cpu);

+

+ double\_rq\_lock(rq\_src, rq\_dest);

+ /\* Already moved. \*/

+ if (task\_cpu(p) != src\_cpu)

+ goto out;

+ /\* Affinity changed (again). \*/

+ if (!cpu\_isset(dest\_cpu, p->cpus\_allowed))

+ goto out;

+

+ on\_rq = p->se.on\_rq;

+ if (on\_rq)

+ deactivate\_task(rq\_src, p, 0);

+

+ set\_task\_cpu(p, dest\_cpu);

+ if (on\_rq) {

+ activate\_task(rq\_dest, p, 0);

+ check\_preempt\_curr(rq\_dest, p);

+ }

+ ret = 1;

+out:

+ double\_rq\_unlock(rq\_src, rq\_dest);

+ return ret;

+}

+

+/\*

+ \* migration\_thread - this is a highprio system thread that performs

+ \* thread migration by bumping thread off CPU then 'pushing' onto

+ \* another runqueue.

+ \*/

+static int migration\_thread(void \*data)

+{

+ int cpu = (long)data;

+ struct rq \*rq;

+

+ rq = cpu\_rq(cpu);

+ BUG\_ON(rq->migration\_thread != current);

+

+ set\_current\_state(TASK\_INTERRUPTIBLE);

+ while (!kthread\_should\_stop()) {

+ struct migration\_req \*req;

+ struct list\_head \*head;

+

+ spin\_lock\_irq(&rq->lock);

+

+ if (cpu\_is\_offline(cpu)) {

+ spin\_unlock\_irq(&rq->lock);

+ goto wait\_to\_die;

+ }

+

+ if (rq->active\_balance) {

+ active\_load\_balance(rq, cpu);

+ rq->active\_balance = 0;

+ }

+

+ head = &rq->migration\_queue;

+

+ if (list\_empty(head)) {

+ spin\_unlock\_irq(&rq->lock);

+ schedule();

+ set\_current\_state(TASK\_INTERRUPTIBLE);

+ continue;

+ }

+ req = list\_entry(head->next, struct migration\_req, list);

+ list\_del\_init(head->next);

+

+ spin\_unlock(&rq->lock);

+ \_\_migrate\_task(req->task, cpu, req->dest\_cpu);

+ local\_irq\_enable();

+

+ complete(&req->done);

+ }

+ \_\_set\_current\_state(TASK\_RUNNING);

+ return 0;

+

+wait\_to\_die:

+ /\* Wait for kthread\_stop \*/

+ set\_current\_state(TASK\_INTERRUPTIBLE);

+ while (!kthread\_should\_stop()) {

+ schedule();

+ set\_current\_state(TASK\_INTERRUPTIBLE);

+ }

+ \_\_set\_current\_state(TASK\_RUNNING);

+ return 0;

+}

+

+#ifdef CONFIG\_HOTPLUG\_CPU

+

+static int \_\_migrate\_task\_irq(struct task\_struct \*p, int src\_cpu, int dest\_cpu)

+{

+ int ret;

+

+ local\_irq\_disable();

+ ret = \_\_migrate\_task(p, src\_cpu, dest\_cpu);

+ local\_irq\_enable();

+ return ret;

+}

+

+/\*

+ \* Figure out where task on dead CPU should go, use force if necessary.

+ \* NOTE: interrupts should be disabled by the caller

+ \*/

+static void move\_task\_off\_dead\_cpu(int dead\_cpu, struct task\_struct \*p)

+{

+ unsigned long flags;

+ cpumask\_t mask;

+ struct rq \*rq;

+ int dest\_cpu;

+

+ do {

+ /\* On same node? \*/

+ mask = node\_to\_cpumask(cpu\_to\_node(dead\_cpu));

+ cpus\_and(mask, mask, p->cpus\_allowed);

+ dest\_cpu = any\_online\_cpu(mask);

+

+ /\* On any allowed CPU? \*/

+ if (dest\_cpu == NR\_CPUS)

+ dest\_cpu = any\_online\_cpu(p->cpus\_allowed);

+

+ /\* No more Mr. Nice Guy. \*/

+ if (dest\_cpu == NR\_CPUS) {

+ cpumask\_t cpus\_allowed = cpuset\_cpus\_allowed\_locked(p);

+ /\*

+ \* Try to stay on the same cpuset, where the

+ \* current cpuset may be a subset of all cpus.

+ \* The cpuset\_cpus\_allowed\_locked() variant of

+ \* cpuset\_cpus\_allowed() will not block. It must be

+ \* called within calls to cpuset\_lock/cpuset\_unlock.

+ \*/

+ rq = task\_rq\_lock(p, &flags);

+ p->cpus\_allowed = cpus\_allowed;

+ dest\_cpu = any\_online\_cpu(p->cpus\_allowed);

+ task\_rq\_unlock(rq, &flags);

+

+ /\*

+ \* Don't tell them about moving exiting tasks or

+ \* kernel threads (both mm NULL), since they never

+ \* leave kernel.

+ \*/

+ if (p->mm && printk\_ratelimit()) {

+ printk(KERN\_INFO "process %d (%s) no "

+ "longer affine to cpu%d\n",

+ task\_pid\_nr(p), p->comm, dead\_cpu);

+ }

+ }

+ } while (!\_\_migrate\_task\_irq(p, dead\_cpu, dest\_cpu));

+}

+

+/\*

+ \* While a dead CPU has no uninterruptible tasks queued at this point,

+ \* it might still have a nonzero ->nr\_uninterruptible counter, because

+ \* for performance reasons the counter is not stricly tracking tasks to

+ \* their home CPUs. So we just add the counter to another CPU's counter,

+ \* to keep the global sum constant after CPU-down:

+ \*/

+static void migrate\_nr\_uninterruptible(struct rq \*rq\_src)

+{

+ struct rq \*rq\_dest = cpu\_rq(any\_online\_cpu(CPU\_MASK\_ALL));

+ unsigned long flags;

+

+ local\_irq\_save(flags);

+ double\_rq\_lock(rq\_src, rq\_dest);

+ rq\_dest->nr\_uninterruptible += rq\_src->nr\_uninterruptible;

+ rq\_src->nr\_uninterruptible = 0;

+ double\_rq\_unlock(rq\_src, rq\_dest);

+ local\_irq\_restore(flags);

+}

+

+/\* Run through task list and migrate tasks from the dead cpu. \*/

+static void migrate\_live\_tasks(int src\_cpu)

+{

+ struct task\_struct \*p, \*t;

+

+ read\_lock(&tasklist\_lock);

+

+ do\_each\_thread(t, p) {

+ if (p == current)

+ continue;

+

+ if (task\_cpu(p) == src\_cpu)

+ move\_task\_off\_dead\_cpu(src\_cpu, p);

+ } while\_each\_thread(t, p);

+

+ read\_unlock(&tasklist\_lock);

+}

+

+/\*

+ \* Schedules idle task to be the next runnable task on current CPU.

+ \* It does so by boosting its priority to highest possible.

+ \* Used by CPU offline code.

+ \*/

+void sched\_idle\_next(void)

+{

+ int this\_cpu = smp\_processor\_id();

+ struct rq \*rq = cpu\_rq(this\_cpu);

+ struct task\_struct \*p = rq->idle;

+ unsigned long flags;

+

+ /\* cpu has to be offline \*/

+ BUG\_ON(cpu\_online(this\_cpu));

+

+ /\*

+ \* Strictly not necessary since rest of the CPUs are stopped by now

+ \* and interrupts disabled on the current cpu.

+ \*/

+ spin\_lock\_irqsave(&rq->lock, flags);

+

+ \_\_setscheduler(rq, p, SCHED\_FIFO, MAX\_RT\_PRIO-1);

+

+ update\_rq\_clock(rq);

+ activate\_task(rq, p, 0);

+

+ spin\_unlock\_irqrestore(&rq->lock, flags);

+}

+

+/\*

+ \* Ensures that the idle task is using init\_mm right before its cpu goes

+ \* offline.

+ \*/

+void idle\_task\_exit(void)

+{

+ struct mm\_struct \*mm = current->active\_mm;

+

+ BUG\_ON(cpu\_online(smp\_processor\_id()));

+

+ if (mm != &init\_mm)

+ switch\_mm(mm, &init\_mm, current);

+ mmdrop(mm);

+}

+

+/\* called under rq->lock with disabled interrupts \*/

+static void migrate\_dead(unsigned int dead\_cpu, struct task\_struct \*p)

+{

+ struct rq \*rq = cpu\_rq(dead\_cpu);

+

+ /\* Must be exiting, otherwise would be on tasklist. \*/

+ BUG\_ON(!p->exit\_state);

+

+ /\* Cannot have done final schedule yet: would have vanished. \*/

+ BUG\_ON(p->state == TASK\_DEAD);

+

+ get\_task\_struct(p);

+

+ /\*

+ \* Drop lock around migration; if someone else moves it,

+ \* that's OK. No task can be added to this CPU, so iteration is

+ \* fine.

+ \*/

+ spin\_unlock\_irq(&rq->lock);

+ move\_task\_off\_dead\_cpu(dead\_cpu, p);

+ spin\_lock\_irq(&rq->lock);

+

+ put\_task\_struct(p);

+}

+

+/\* release\_task() removes task from tasklist, so we won't find dead tasks. \*/

+static void migrate\_dead\_tasks(unsigned int dead\_cpu)

+{

+ struct rq \*rq = cpu\_rq(dead\_cpu);

+ struct task\_struct \*next;

+

+ for ( ; ; ) {

+ if (!rq->nr\_running)

+ break;

+ update\_rq\_clock(rq);

+ next = pick\_next\_task(rq, rq->curr);

+ if (!next)

+ break;

+ migrate\_dead(dead\_cpu, next);

+

+ }

+}

+#endif /\* CONFIG\_HOTPLUG\_CPU \*/

+

+#if defined(CONFIG\_SCHED\_DEBUG) && defined(CONFIG\_SYSCTL)

+

+static struct ctl\_table sd\_ctl\_dir[] = {

+ {

+ .procname = "sched\_domain",

+ .mode = 0555,

+ },

+ {0, },

+};

+

+static struct ctl\_table sd\_ctl\_root[] = {

+ {

+ .ctl\_name = CTL\_KERN,

+ .procname = "kernel",

+ .mode = 0555,

+ .child = sd\_ctl\_dir,

+ },

+ {0, },

+};

+

+static struct ctl\_table \*sd\_alloc\_ctl\_entry(int n)

+{

+ struct ctl\_table \*entry =

+ kcalloc(n, sizeof(struct ctl\_table), GFP\_KERNEL);

+

+ return entry;

+}

+

+static void sd\_free\_ctl\_entry(struct ctl\_table \*\*tablep)

+{

+ struct ctl\_table \*entry;

+

+ /\*

+ \* In the intermediate directories, both the child directory and

+ \* procname are dynamically allocated and could fail but the mode

+ \* will always be set. In the lowest directory the names are

+ \* static strings and all have proc handlers.

+ \*/

+ for (entry = \*tablep; entry->mode; entry++) {

+ if (entry->child)

+ sd\_free\_ctl\_entry(&entry->child);

+ if (entry->proc\_handler == NULL)

+ kfree(entry->procname);

+ }

+

+ kfree(\*tablep);

+ \*tablep = NULL;

+}

+

+static void

+set\_table\_entry(struct ctl\_table \*entry,

+ const char \*procname, void \*data, int maxlen,

+ mode\_t mode, proc\_handler \*proc\_handler)

+{

+ entry->procname = procname;

+ entry->data = data;

+ entry->maxlen = maxlen;

+ entry->mode = mode;

+ entry->proc\_handler = proc\_handler;

+}

+

+static struct ctl\_table \*

+sd\_alloc\_ctl\_domain\_table(struct sched\_domain \*sd)

+{

+ struct ctl\_table \*table = sd\_alloc\_ctl\_entry(12);

+

+ if (table == NULL)

+ return NULL;

+

+ set\_table\_entry(&table[0], "min\_interval", &sd->min\_interval,

+ sizeof(long), 0644, proc\_doulongvec\_minmax);

+ set\_table\_entry(&table[1], "max\_interval", &sd->max\_interval,

+ sizeof(long), 0644, proc\_doulongvec\_minmax);

+ set\_table\_entry(&table[2], "busy\_idx", &sd->busy\_idx,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[3], "idle\_idx", &sd->idle\_idx,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[4], "newidle\_idx", &sd->newidle\_idx,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[5], "wake\_idx", &sd->wake\_idx,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[6], "forkexec\_idx", &sd->forkexec\_idx,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[7], "busy\_factor", &sd->busy\_factor,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[8], "imbalance\_pct", &sd->imbalance\_pct,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[9], "cache\_nice\_tries",

+ &sd->cache\_nice\_tries,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ set\_table\_entry(&table[10], "flags", &sd->flags,

+ sizeof(int), 0644, proc\_dointvec\_minmax);

+ /\* &table[11] is terminator \*/

+

+ return table;

+}

+

+static ctl\_table \*sd\_alloc\_ctl\_cpu\_table(int cpu)

+{

+ struct ctl\_table \*entry, \*table;

+ struct sched\_domain \*sd;

+ int domain\_num = 0, i;

+ char buf[32];

+

+ for\_each\_domain(cpu, sd)

+ domain\_num++;

+ entry = table = sd\_alloc\_ctl\_entry(domain\_num + 1);

+ if (table == NULL)

+ return NULL;

+

+ i = 0;

+ for\_each\_domain(cpu, sd) {

+ snprintf(buf, 32, "domain%d", i);

+ entry->procname = kstrdup(buf, GFP\_KERNEL);

+ entry->mode = 0555;

+ entry->child = sd\_alloc\_ctl\_domain\_table(sd);

+ entry++;

+ i++;

+ }

+ return table;

+}

+

+static struct ctl\_table\_header \*sd\_sysctl\_header;

+static void register\_sched\_domain\_sysctl(void)

+{

+ int i, cpu\_num = num\_online\_cpus();

+ struct ctl\_table \*entry = sd\_alloc\_ctl\_entry(cpu\_num + 1);

+ char buf[32];

+

+ WARN\_ON(sd\_ctl\_dir[0].child);

+ sd\_ctl\_dir[0].child = entry;

+

+ if (entry == NULL)

+ return;

+

+ for\_each\_online\_cpu(i) {

+ snprintf(buf, 32, "cpu%d", i);

+ entry->procname = kstrdup(buf, GFP\_KERNEL);

+ entry->mode = 0555;

+ entry->child = sd\_alloc\_ctl\_cpu\_table(i);

+ entry++;

+ }

+

+ WARN\_ON(sd\_sysctl\_header);

+ sd\_sysctl\_header = register\_sysctl\_table(sd\_ctl\_root);

+}

+

+/\* may be called multiple times per register \*/

+static void unregister\_sched\_domain\_sysctl(void)

+{

+ if (sd\_sysctl\_header)

+ unregister\_sysctl\_table(sd\_sysctl\_header);

+ sd\_sysctl\_header = NULL;

+ if (sd\_ctl\_dir[0].child)

+ sd\_free\_ctl\_entry(&sd\_ctl\_dir[0].child);

+}

+#else

+static void register\_sched\_domain\_sysctl(void)

+{

+}

+static void unregister\_sched\_domain\_sysctl(void)

+{

+}

+#endif

+

+/\*

+ \* migration\_call - callback that gets triggered when a CPU is added.

+ \* Here we can start up the necessary migration thread for the new CPU.

+ \*/

+static int \_\_cpuinit

+migration\_call(struct notifier\_block \*nfb, unsigned long action, void \*hcpu)

+{

+ struct task\_struct \*p;

+ int cpu = (long)hcpu;

+ unsigned long flags;

+ struct rq \*rq;

+

+ switch (action) {

+ case CPU\_LOCK\_ACQUIRE:

+ mutex\_lock(&sched\_hotcpu\_mutex);

+ break;

+

+ case CPU\_UP\_PREPARE:

+ case CPU\_UP\_PREPARE\_FROZEN:

+ p = kthread\_create(migration\_thread, hcpu, "migration/%d", cpu);

+ if (IS\_ERR(p))

+ return NOTIFY\_BAD;

+ kthread\_bind(p, cpu);

+ /\* Must be high prio: stop\_machine expects to yield to it. \*/

+ rq = task\_rq\_lock(p, &flags);

+ \_\_setscheduler(rq, p, SCHED\_FIFO, MAX\_RT\_PRIO-1);

+ task\_rq\_unlock(rq, &flags);

+ cpu\_rq(cpu)->migration\_thread = p;

+ break;

+

+ case CPU\_ONLINE:

+ case CPU\_ONLINE\_FROZEN:

+ /\* Strictly unnecessary, as first user will wake it. \*/

+ wake\_up\_process(cpu\_rq(cpu)->migration\_thread);

+ break;

+

+#ifdef CONFIG\_HOTPLUG\_CPU

+ case CPU\_UP\_CANCELED:

+ case CPU\_UP\_CANCELED\_FROZEN:

+ if (!cpu\_rq(cpu)->migration\_thread)

+ break;

+ /\* Unbind it from offline cpu so it can run. Fall thru. \*/

+ kthread\_bind(cpu\_rq(cpu)->migration\_thread,

+ any\_online\_cpu(cpu\_online\_map));

+ kthread\_stop(cpu\_rq(cpu)->migration\_thread);

+ cpu\_rq(cpu)->migration\_thread = NULL;

+ break;

+

+ case CPU\_DEAD:

+ case CPU\_DEAD\_FROZEN:

+ cpuset\_lock(); /\* around calls to cpuset\_cpus\_allowed\_lock() \*/

+ migrate\_live\_tasks(cpu);

+ rq = cpu\_rq(cpu);

+ kthread\_stop(rq->migration\_thread);

+ rq->migration\_thread = NULL;

+ /\* Idle task back to normal (off runqueue, low prio) \*/

+ spin\_lock\_irq(&rq->lock);

+ update\_rq\_clock(rq);

+ deactivate\_task(rq, rq->idle, 0);

+ rq->idle->static\_prio = MAX\_PRIO;

+ \_\_setscheduler(rq, rq->idle, SCHED\_NORMAL, 0);

+ rq->idle->sched\_class = &idle\_sched\_class;

+ migrate\_dead\_tasks(cpu);

+ spin\_unlock\_irq(&rq->lock);

+ cpuset\_unlock();

+ migrate\_nr\_uninterruptible(rq);

+ BUG\_ON(rq->nr\_running != 0);

+

+ /\*

+ \* No need to migrate the tasks: it was best-effort if

+ \* they didn't take sched\_hotcpu\_mutex. Just wake up

+ \* the requestors.

+ \*/

+ spin\_lock\_irq(&rq->lock);

+ while (!list\_empty(&rq->migration\_queue)) {

+ struct migration\_req \*req;

+

+ req = list\_entry(rq->migration\_queue.next,

+ struct migration\_req, list);

+ list\_del\_init(&req->list);

+ complete(&req->done);

+ }

+ spin\_unlock\_irq(&rq->lock);

+ break;

+#endif

+ case CPU\_LOCK\_RELEASE:

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+ break;

+ }

+ return NOTIFY\_OK;

+}

+

+/\* Register at highest priority so that task migration (migrate\_all\_tasks)

+ \* happens before everything else.

+ \*/

+static struct notifier\_block \_\_cpuinitdata migration\_notifier = {

+ .notifier\_call = migration\_call,

+ .priority = 10

+};

+

+void \_\_init migration\_init(void)

+{

+ void \*cpu = (void \*)(long)smp\_processor\_id();

+ int err;

+

+ /\* Start one for the boot CPU: \*/

+ err = migration\_call(&migration\_notifier, CPU\_UP\_PREPARE, cpu);

+ BUG\_ON(err == NOTIFY\_BAD);

+ migration\_call(&migration\_notifier, CPU\_ONLINE, cpu);

+ register\_cpu\_notifier(&migration\_notifier);

+}

+#endif

+

+#ifdef CONFIG\_SMP

+

+/\* Number of possible processor ids \*/

+int nr\_cpu\_ids \_\_read\_mostly = NR\_CPUS;

+EXPORT\_SYMBOL(nr\_cpu\_ids);

+

+#ifdef CONFIG\_SCHED\_DEBUG

+

+static int sched\_domain\_debug\_one(struct sched\_domain \*sd, int cpu, int level)

+{

+ struct sched\_group \*group = sd->groups;

+ cpumask\_t groupmask;

+ char str[NR\_CPUS];

+

+ cpumask\_scnprintf(str, NR\_CPUS, sd->span);

+ cpus\_clear(groupmask);

+

+ printk(KERN\_DEBUG "%\*s domain %d: ", level, "", level);

+

+ if (!(sd->flags & SD\_LOAD\_BALANCE)) {

+ printk("does not load-balance\n");

+ if (sd->parent)

+ printk(KERN\_ERR "ERROR: !SD\_LOAD\_BALANCE domain"

+ " has parent");

+ return -1;

+ }

+

+ printk(KERN\_CONT "span %s\n", str);

+

+ if (!cpu\_isset(cpu, sd->span)) {

+ printk(KERN\_ERR "ERROR: domain->span does not contain "

+ "CPU%d\n", cpu);

+ }

+ if (!cpu\_isset(cpu, group->cpumask)) {

+ printk(KERN\_ERR "ERROR: domain->groups does not contain"

+ " CPU%d\n", cpu);

+ }

+

+ printk(KERN\_DEBUG "%\*s groups:", level + 1, "");

+ do {

+ if (!group) {

+ printk("\n");

+ printk(KERN\_ERR "ERROR: group is NULL\n");

+ break;

+ }

+

+ if (!group->\_\_cpu\_power) {

+ printk(KERN\_CONT "\n");

+ printk(KERN\_ERR "ERROR: domain->cpu\_power not "

+ "set\n");

+ break;

+ }

+

+ if (!cpus\_weight(group->cpumask)) {

+ printk(KERN\_CONT "\n");

+ printk(KERN\_ERR "ERROR: empty group\n");

+ break;

+ }

+

+ if (cpus\_intersects(groupmask, group->cpumask)) {

+ printk(KERN\_CONT "\n");

+ printk(KERN\_ERR "ERROR: repeated CPUs\n");

+ break;

+ }

+

+ cpus\_or(groupmask, groupmask, group->cpumask);

+

+ cpumask\_scnprintf(str, NR\_CPUS, group->cpumask);

+ printk(KERN\_CONT " %s", str);

+

+ group = group->next;

+ } while (group != sd->groups);

+ printk(KERN\_CONT "\n");

+

+ if (!cpus\_equal(sd->span, groupmask))

+ printk(KERN\_ERR "ERROR: groups don't span domain->span\n");

+

+ if (sd->parent && !cpus\_subset(groupmask, sd->parent->span))

+ printk(KERN\_ERR "ERROR: parent span is not a superset "

+ "of domain->span\n");

+ return 0;

+}

+

+static void sched\_domain\_debug(struct sched\_domain \*sd, int cpu)

+{

+ int level = 0;

+

+ if (!sd) {

+ printk(KERN\_DEBUG "CPU%d attaching NULL sched-domain.\n", cpu);

+ return;

+ }

+

+ printk(KERN\_DEBUG "CPU%d attaching sched-domain:\n", cpu);

+

+ for (;;) {

+ if (sched\_domain\_debug\_one(sd, cpu, level))

+ break;

+ level++;

+ sd = sd->parent;

+ if (!sd)

+ break;

+ }

+}

+#else

+# define sched\_domain\_debug(sd, cpu) do { } while (0)

+#endif

+

+static int sd\_degenerate(struct sched\_domain \*sd)

+{

+ if (cpus\_weight(sd->span) == 1)

+ return 1;

+

+ /\* Following flags need at least 2 groups \*/

+ if (sd->flags & (SD\_LOAD\_BALANCE |

+ SD\_BALANCE\_NEWIDLE |

+ SD\_BALANCE\_FORK |

+ SD\_BALANCE\_EXEC |

+ SD\_SHARE\_CPUPOWER |

+ SD\_SHARE\_PKG\_RESOURCES)) {

+ if (sd->groups != sd->groups->next)

+ return 0;

+ }

+

+ /\* Following flags don't use groups \*/

+ if (sd->flags & (SD\_WAKE\_IDLE |

+ SD\_WAKE\_AFFINE |

+ SD\_WAKE\_BALANCE))

+ return 0;

+

+ return 1;

+}

+

+static int

+sd\_parent\_degenerate(struct sched\_domain \*sd, struct sched\_domain \*parent)

+{

+ unsigned long cflags = sd->flags, pflags = parent->flags;

+

+ if (sd\_degenerate(parent))

+ return 1;

+

+ if (!cpus\_equal(sd->span, parent->span))

+ return 0;

+

+ /\* Does parent contain flags not in child? \*/

+ /\* WAKE\_BALANCE is a subset of WAKE\_AFFINE \*/

+ if (cflags & SD\_WAKE\_AFFINE)

+ pflags &= ~SD\_WAKE\_BALANCE;

+ /\* Flags needing groups don't count if only 1 group in parent \*/

+ if (parent->groups == parent->groups->next) {

+ pflags &= ~(SD\_LOAD\_BALANCE |

+ SD\_BALANCE\_NEWIDLE |

+ SD\_BALANCE\_FORK |

+ SD\_BALANCE\_EXEC |

+ SD\_SHARE\_CPUPOWER |

+ SD\_SHARE\_PKG\_RESOURCES);

+ }

+ if (~cflags & pflags)

+ return 0;

+

+ return 1;

+}

+

+/\*

+ \* Attach the domain 'sd' to 'cpu' as its base domain. Callers must

+ \* hold the hotplug lock.

+ \*/

+static void cpu\_attach\_domain(struct sched\_domain \*sd, int cpu)

+{

+ struct rq \*rq = cpu\_rq(cpu);

+ struct sched\_domain \*tmp;

+

+ /\* Remove the sched domains which do not contribute to scheduling. \*/

+ for (tmp = sd; tmp; tmp = tmp->parent) {

+ struct sched\_domain \*parent = tmp->parent;

+ if (!parent)

+ break;

+ if (sd\_parent\_degenerate(tmp, parent)) {

+ tmp->parent = parent->parent;

+ if (parent->parent)

+ parent->parent->child = tmp;

+ }

+ }

+

+ if (sd && sd\_degenerate(sd)) {

+ sd = sd->parent;

+ if (sd)

+ sd->child = NULL;

+ }

+

+ sched\_domain\_debug(sd, cpu);

+

+ rcu\_assign\_pointer(rq->sd, sd);

+}

+

+/\* cpus with isolated domains \*/

+static cpumask\_t cpu\_isolated\_map = CPU\_MASK\_NONE;

+

+/\* Setup the mask of cpus configured for isolated domains \*/

+static int \_\_init isolated\_cpu\_setup(char \*str)

+{

+ int ints[NR\_CPUS], i;

+

+ str = get\_options(str, ARRAY\_SIZE(ints), ints);

+ cpus\_clear(cpu\_isolated\_map);

+ for (i = 1; i <= ints[0]; i++)

+ if (ints[i] < NR\_CPUS)

+ cpu\_set(ints[i], cpu\_isolated\_map);

+ return 1;

+}

+

+\_\_setup("isolcpus=", isolated\_cpu\_setup);

+

+/\*

+ \* init\_sched\_build\_groups takes the cpumask we wish to span, and a pointer

+ \* to a function which identifies what group(along with sched group) a CPU

+ \* belongs to. The return value of group\_fn must be a >= 0 and < NR\_CPUS

+ \* (due to the fact that we keep track of groups covered with a cpumask\_t).

+ \*

+ \* init\_sched\_build\_groups will build a circular linked list of the groups

+ \* covered by the given span, and will set each group's ->cpumask correctly,

+ \* and ->cpu\_power to 0.

+ \*/

+static void

+init\_sched\_build\_groups(cpumask\_t span, const cpumask\_t \*cpu\_map,

+ int (\*group\_fn)(int cpu, const cpumask\_t \*cpu\_map,

+ struct sched\_group \*\*sg))

+{

+ struct sched\_group \*first = NULL, \*last = NULL;

+ cpumask\_t covered = CPU\_MASK\_NONE;

+ int i;

+

+ for\_each\_cpu\_mask(i, span) {

+ struct sched\_group \*sg;

+ int group = group\_fn(i, cpu\_map, &sg);

+ int j;

+

+ if (cpu\_isset(i, covered))

+ continue;

+

+ sg->cpumask = CPU\_MASK\_NONE;

+ sg->\_\_cpu\_power = 0;

+

+ for\_each\_cpu\_mask(j, span) {

+ if (group\_fn(j, cpu\_map, NULL) != group)

+ continue;

+

+ cpu\_set(j, covered);

+ cpu\_set(j, sg->cpumask);

+ }

+ if (!first)

+ first = sg;

+ if (last)

+ last->next = sg;

+ last = sg;

+ }

+ last->next = first;

+}

+

+#define SD\_NODES\_PER\_DOMAIN 16

+

+#ifdef CONFIG\_NUMA

+

+/\*\*

+ \* find\_next\_best\_node - find the next node to include in a sched\_domain

+ \* @node: node whose sched\_domain we're building

+ \* @used\_nodes: nodes already in the sched\_domain

+ \*

+ \* Find the next node to include in a given scheduling domain. Simply

+ \* finds the closest node not already in the @used\_nodes map.

+ \*

+ \* Should use nodemask\_t.

+ \*/

+static int find\_next\_best\_node(int node, unsigned long \*used\_nodes)

+{

+ int i, n, val, min\_val, best\_node = 0;

+

+ min\_val = INT\_MAX;

+

+ for (i = 0; i < MAX\_NUMNODES; i++) {

+ /\* Start at @node \*/

+ n = (node + i) % MAX\_NUMNODES;

+

+ if (!nr\_cpus\_node(n))

+ continue;

+

+ /\* Skip already used nodes \*/

+ if (test\_bit(n, used\_nodes))

+ continue;

+

+ /\* Simple min distance search \*/

+ val = node\_distance(node, n);

+

+ if (val < min\_val) {

+ min\_val = val;

+ best\_node = n;

+ }

+ }

+

+ set\_bit(best\_node, used\_nodes);

+ return best\_node;

+}

+

+/\*\*

+ \* sched\_domain\_node\_span - get a cpumask for a node's sched\_domain

+ \* @node: node whose cpumask we're constructing

+ \* @size: number of nodes to include in this span

+ \*

+ \* Given a node, construct a good cpumask for its sched\_domain to span. It

+ \* should be one that prevents unnecessary balancing, but also spreads tasks

+ \* out optimally.

+ \*/

+static cpumask\_t sched\_domain\_node\_span(int node)

+{

+ DECLARE\_BITMAP(used\_nodes, MAX\_NUMNODES);

+ cpumask\_t span, nodemask;

+ int i;

+

+ cpus\_clear(span);

+ bitmap\_zero(used\_nodes, MAX\_NUMNODES);

+

+ nodemask = node\_to\_cpumask(node);

+ cpus\_or(span, span, nodemask);

+ set\_bit(node, used\_nodes);

+

+ for (i = 1; i < SD\_NODES\_PER\_DOMAIN; i++) {

+ int next\_node = find\_next\_best\_node(node, used\_nodes);

+

+ nodemask = node\_to\_cpumask(next\_node);

+ cpus\_or(span, span, nodemask);

+ }

+

+ return span;

+}

+#endif

+

+int sched\_smt\_power\_savings = 0, sched\_mc\_power\_savings = 0;

+

+/\*

+ \* SMT sched-domains:

+ \*/

+#ifdef CONFIG\_SCHED\_SMT

+static DEFINE\_PER\_CPU(struct sched\_domain, cpu\_domains);

+static DEFINE\_PER\_CPU(struct sched\_group, sched\_group\_cpus);

+

+static int

+cpu\_to\_cpu\_group(int cpu, const cpumask\_t \*cpu\_map, struct sched\_group \*\*sg)

+{

+ if (sg)

+ \*sg = &per\_cpu(sched\_group\_cpus, cpu);

+ return cpu;

+}

+#endif

+

+/\*

+ \* multi-core sched-domains:

+ \*/

+#ifdef CONFIG\_SCHED\_MC

+static DEFINE\_PER\_CPU(struct sched\_domain, core\_domains);

+static DEFINE\_PER\_CPU(struct sched\_group, sched\_group\_core);

+#endif

+

+#if defined(CONFIG\_SCHED\_MC) && defined(CONFIG\_SCHED\_SMT)

+static int

+cpu\_to\_core\_group(int cpu, const cpumask\_t \*cpu\_map, struct sched\_group \*\*sg)

+{

+ int group;

+ cpumask\_t mask = per\_cpu(cpu\_sibling\_map, cpu);

+ cpus\_and(mask, mask, \*cpu\_map);

+ group = first\_cpu(mask);

+ if (sg)

+ \*sg = &per\_cpu(sched\_group\_core, group);

+ return group;

+}

+#elif defined(CONFIG\_SCHED\_MC)

+static int

+cpu\_to\_core\_group(int cpu, const cpumask\_t \*cpu\_map, struct sched\_group \*\*sg)

+{

+ if (sg)

+ \*sg = &per\_cpu(sched\_group\_core, cpu);

+ return cpu;

+}

+#endif

+

+static DEFINE\_PER\_CPU(struct sched\_domain, phys\_domains);

+static DEFINE\_PER\_CPU(struct sched\_group, sched\_group\_phys);

+

+static int

+cpu\_to\_phys\_group(int cpu, const cpumask\_t \*cpu\_map, struct sched\_group \*\*sg)

+{

+ int group;

+#ifdef CONFIG\_SCHED\_MC

+ cpumask\_t mask = cpu\_coregroup\_map(cpu);

+ cpus\_and(mask, mask, \*cpu\_map);

+ group = first\_cpu(mask);

+#elif defined(CONFIG\_SCHED\_SMT)

+ cpumask\_t mask = per\_cpu(cpu\_sibling\_map, cpu);

+ cpus\_and(mask, mask, \*cpu\_map);

+ group = first\_cpu(mask);

+#else

+ group = cpu;

+#endif

+ if (sg)

+ \*sg = &per\_cpu(sched\_group\_phys, group);

+ return group;

+}

+

+#ifdef CONFIG\_NUMA

+/\*

+ \* The init\_sched\_build\_groups can't handle what we want to do with node

+ \* groups, so roll our own. Now each node has its own list of groups which

+ \* gets dynamically allocated.

+ \*/

+static DEFINE\_PER\_CPU(struct sched\_domain, node\_domains);

+static struct sched\_group \*\*sched\_group\_nodes\_bycpu[NR\_CPUS];

+

+static DEFINE\_PER\_CPU(struct sched\_domain, allnodes\_domains);

+static DEFINE\_PER\_CPU(struct sched\_group, sched\_group\_allnodes);

+

+static int cpu\_to\_allnodes\_group(int cpu, const cpumask\_t \*cpu\_map,

+ struct sched\_group \*\*sg)

+{

+ cpumask\_t nodemask = node\_to\_cpumask(cpu\_to\_node(cpu));

+ int group;

+

+ cpus\_and(nodemask, nodemask, \*cpu\_map);

+ group = first\_cpu(nodemask);

+

+ if (sg)

+ \*sg = &per\_cpu(sched\_group\_allnodes, group);

+ return group;

+}

+

+static void init\_numa\_sched\_groups\_power(struct sched\_group \*group\_head)

+{

+ struct sched\_group \*sg = group\_head;

+ int j;

+

+ if (!sg)

+ return;

+ do {

+ for\_each\_cpu\_mask(j, sg->cpumask) {

+ struct sched\_domain \*sd;

+

+ sd = &per\_cpu(phys\_domains, j);

+ if (j != first\_cpu(sd->groups->cpumask)) {

+ /\*

+ \* Only add "power" once for each

+ \* physical package.

+ \*/

+ continue;

+ }

+

+ sg\_inc\_cpu\_power(sg, sd->groups->\_\_cpu\_power);

+ }

+ sg = sg->next;

+ } while (sg != group\_head);

+}

+#endif

+

+#ifdef CONFIG\_NUMA

+/\* Free memory allocated for various sched\_group structures \*/

+static void free\_sched\_groups(const cpumask\_t \*cpu\_map)

+{

+ int cpu, i;

+

+ for\_each\_cpu\_mask(cpu, \*cpu\_map) {

+ struct sched\_group \*\*sched\_group\_nodes

+ = sched\_group\_nodes\_bycpu[cpu];

+

+ if (!sched\_group\_nodes)

+ continue;

+

+ for (i = 0; i < MAX\_NUMNODES; i++) {

+ cpumask\_t nodemask = node\_to\_cpumask(i);

+ struct sched\_group \*oldsg, \*sg = sched\_group\_nodes[i];

+

+ cpus\_and(nodemask, nodemask, \*cpu\_map);

+ if (cpus\_empty(nodemask))

+ continue;

+

+ if (sg == NULL)

+ continue;

+ sg = sg->next;

+next\_sg:

+ oldsg = sg;

+ sg = sg->next;

+ kfree(oldsg);

+ if (oldsg != sched\_group\_nodes[i])

+ goto next\_sg;

+ }

+ kfree(sched\_group\_nodes);

+ sched\_group\_nodes\_bycpu[cpu] = NULL;

+ }

+}

+#else

+static void free\_sched\_groups(const cpumask\_t \*cpu\_map)

+{

+}

+#endif

+

+/\*

+ \* Initialize sched groups cpu\_power.

+ \*

+ \* cpu\_power indicates the capacity of sched group, which is used while

+ \* distributing the load between different sched groups in a sched domain.

+ \* Typically cpu\_power for all the groups in a sched domain will be same unless

+ \* there are asymmetries in the topology. If there are asymmetries, group

+ \* having more cpu\_power will pickup more load compared to the group having

+ \* less cpu\_power.

+ \*

+ \* cpu\_power will be a multiple of SCHED\_LOAD\_SCALE. This multiple represents

+ \* the maximum number of tasks a group can handle in the presence of other idle

+ \* or lightly loaded groups in the same sched domain.

+ \*/

+static void init\_sched\_groups\_power(int cpu, struct sched\_domain \*sd)

+{

+ struct sched\_domain \*child;

+ struct sched\_group \*group;

+

+ WARN\_ON(!sd || !sd->groups);

+

+ if (cpu != first\_cpu(sd->groups->cpumask))

+ return;

+

+ child = sd->child;

+

+ sd->groups->\_\_cpu\_power = 0;

+

+ /\*

+ \* For perf policy, if the groups in child domain share resources

+ \* (for example cores sharing some portions of the cache hierarchy

+ \* or SMT), then set this domain groups cpu\_power such that each group

+ \* can handle only one task, when there are other idle groups in the

+ \* same sched domain.

+ \*/

+ if (!child || (!(sd->flags & SD\_POWERSAVINGS\_BALANCE) &&

+ (child->flags &

+ (SD\_SHARE\_CPUPOWER | SD\_SHARE\_PKG\_RESOURCES)))) {

+ sg\_inc\_cpu\_power(sd->groups, SCHED\_LOAD\_SCALE);

+ return;

+ }

+

+ /\*

+ \* add cpu\_power of each child group to this groups cpu\_power

+ \*/

+ group = child->groups;

+ do {

+ sg\_inc\_cpu\_power(sd->groups, group->\_\_cpu\_power);

+ group = group->next;

+ } while (group != child->groups);

+}

+

+/\*

+ \* Build sched domains for a given set of cpus and attach the sched domains

+ \* to the individual cpus

+ \*/

+static int build\_sched\_domains(const cpumask\_t \*cpu\_map)

+{

+ int i;

+#ifdef CONFIG\_NUMA

+ struct sched\_group \*\*sched\_group\_nodes = NULL;

+ int sd\_allnodes = 0;

+

+ /\*

+ \* Allocate the per-node list of sched groups

+ \*/

+ sched\_group\_nodes = kcalloc(MAX\_NUMNODES, sizeof(struct sched\_group \*),

+ GFP\_KERNEL);

+ if (!sched\_group\_nodes) {

+ printk(KERN\_WARNING "Can not alloc sched group node list\n");

+ return -ENOMEM;

+ }

+ sched\_group\_nodes\_bycpu[first\_cpu(\*cpu\_map)] = sched\_group\_nodes;

+#endif

+

+ /\*

+ \* Set up domains for cpus specified by the cpu\_map.

+ \*/

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ struct sched\_domain \*sd = NULL, \*p;

+ cpumask\_t nodemask = node\_to\_cpumask(cpu\_to\_node(i));

+

+ cpus\_and(nodemask, nodemask, \*cpu\_map);

+

+#ifdef CONFIG\_NUMA

+ if (cpus\_weight(\*cpu\_map) >

+ SD\_NODES\_PER\_DOMAIN\*cpus\_weight(nodemask)) {

+ sd = &per\_cpu(allnodes\_domains, i);

+ \*sd = SD\_ALLNODES\_INIT;

+ sd->span = \*cpu\_map;

+ cpu\_to\_allnodes\_group(i, cpu\_map, &sd->groups);

+ p = sd;

+ sd\_allnodes = 1;

+ } else

+ p = NULL;

+

+ sd = &per\_cpu(node\_domains, i);

+ \*sd = SD\_NODE\_INIT;

+ sd->span = sched\_domain\_node\_span(cpu\_to\_node(i));

+ sd->parent = p;

+ if (p)

+ p->child = sd;

+ cpus\_and(sd->span, sd->span, \*cpu\_map);

+#endif

+

+ p = sd;

+ sd = &per\_cpu(phys\_domains, i);

+ \*sd = SD\_CPU\_INIT;

+ sd->span = nodemask;

+ sd->parent = p;

+ if (p)

+ p->child = sd;

+ cpu\_to\_phys\_group(i, cpu\_map, &sd->groups);

+

+#ifdef CONFIG\_SCHED\_MC

+ p = sd;

+ sd = &per\_cpu(core\_domains, i);

+ \*sd = SD\_MC\_INIT;

+ sd->span = cpu\_coregroup\_map(i);

+ cpus\_and(sd->span, sd->span, \*cpu\_map);

+ sd->parent = p;

+ p->child = sd;

+ cpu\_to\_core\_group(i, cpu\_map, &sd->groups);

+#endif

+

+#ifdef CONFIG\_SCHED\_SMT

+ p = sd;

+ sd = &per\_cpu(cpu\_domains, i);

+ \*sd = SD\_SIBLING\_INIT;

+ sd->span = per\_cpu(cpu\_sibling\_map, i);

+ cpus\_and(sd->span, sd->span, \*cpu\_map);

+ sd->parent = p;

+ p->child = sd;

+ cpu\_to\_cpu\_group(i, cpu\_map, &sd->groups);

+#endif

+ }

+

+#ifdef CONFIG\_SCHED\_SMT

+ /\* Set up CPU (sibling) groups \*/

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ cpumask\_t this\_sibling\_map = per\_cpu(cpu\_sibling\_map, i);

+ cpus\_and(this\_sibling\_map, this\_sibling\_map, \*cpu\_map);

+ if (i != first\_cpu(this\_sibling\_map))

+ continue;

+

+ init\_sched\_build\_groups(this\_sibling\_map, cpu\_map,

+ &cpu\_to\_cpu\_group);

+ }

+#endif

+

+#ifdef CONFIG\_SCHED\_MC

+ /\* Set up multi-core groups \*/

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ cpumask\_t this\_core\_map = cpu\_coregroup\_map(i);

+ cpus\_and(this\_core\_map, this\_core\_map, \*cpu\_map);

+ if (i != first\_cpu(this\_core\_map))

+ continue;

+ init\_sched\_build\_groups(this\_core\_map, cpu\_map,

+ &cpu\_to\_core\_group);

+ }

+#endif

+

+ /\* Set up physical groups \*/

+ for (i = 0; i < MAX\_NUMNODES; i++) {

+ cpumask\_t nodemask = node\_to\_cpumask(i);

+

+ cpus\_and(nodemask, nodemask, \*cpu\_map);

+ if (cpus\_empty(nodemask))

+ continue;

+

+ init\_sched\_build\_groups(nodemask, cpu\_map, &cpu\_to\_phys\_group);

+ }

+

+#ifdef CONFIG\_NUMA

+ /\* Set up node groups \*/

+ if (sd\_allnodes)

+ init\_sched\_build\_groups(\*cpu\_map, cpu\_map,

+ &cpu\_to\_allnodes\_group);

+

+ for (i = 0; i < MAX\_NUMNODES; i++) {

+ /\* Set up node groups \*/

+ struct sched\_group \*sg, \*prev;

+ cpumask\_t nodemask = node\_to\_cpumask(i);

+ cpumask\_t domainspan;

+ cpumask\_t covered = CPU\_MASK\_NONE;

+ int j;

+

+ cpus\_and(nodemask, nodemask, \*cpu\_map);

+ if (cpus\_empty(nodemask)) {

+ sched\_group\_nodes[i] = NULL;

+ continue;

+ }

+

+ domainspan = sched\_domain\_node\_span(i);

+ cpus\_and(domainspan, domainspan, \*cpu\_map);

+

+ sg = kmalloc\_node(sizeof(struct sched\_group), GFP\_KERNEL, i);

+ if (!sg) {

+ printk(KERN\_WARNING "Can not alloc domain group for "

+ "node %d\n", i);

+ goto error;

+ }

+ sched\_group\_nodes[i] = sg;

+ for\_each\_cpu\_mask(j, nodemask) {

+ struct sched\_domain \*sd;

+

+ sd = &per\_cpu(node\_domains, j);

+ sd->groups = sg;

+ }

+ sg->\_\_cpu\_power = 0;

+ sg->cpumask = nodemask;

+ sg->next = sg;

+ cpus\_or(covered, covered, nodemask);

+ prev = sg;

+

+ for (j = 0; j < MAX\_NUMNODES; j++) {

+ cpumask\_t tmp, notcovered;

+ int n = (i + j) % MAX\_NUMNODES;

+

+ cpus\_complement(notcovered, covered);

+ cpus\_and(tmp, notcovered, \*cpu\_map);

+ cpus\_and(tmp, tmp, domainspan);

+ if (cpus\_empty(tmp))

+ break;

+

+ nodemask = node\_to\_cpumask(n);

+ cpus\_and(tmp, tmp, nodemask);

+ if (cpus\_empty(tmp))

+ continue;

+

+ sg = kmalloc\_node(sizeof(struct sched\_group),

+ GFP\_KERNEL, i);

+ if (!sg) {

+ printk(KERN\_WARNING

+ "Can not alloc domain group for node %d\n", j);

+ goto error;

+ }

+ sg->\_\_cpu\_power = 0;

+ sg->cpumask = tmp;

+ sg->next = prev->next;

+ cpus\_or(covered, covered, tmp);

+ prev->next = sg;

+ prev = sg;

+ }

+ }

+#endif

+

+ /\* Calculate CPU power for physical packages and nodes \*/

+#ifdef CONFIG\_SCHED\_SMT

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ struct sched\_domain \*sd = &per\_cpu(cpu\_domains, i);

+

+ init\_sched\_groups\_power(i, sd);

+ }

+#endif

+#ifdef CONFIG\_SCHED\_MC

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ struct sched\_domain \*sd = &per\_cpu(core\_domains, i);

+

+ init\_sched\_groups\_power(i, sd);

+ }

+#endif

+

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ struct sched\_domain \*sd = &per\_cpu(phys\_domains, i);

+

+ init\_sched\_groups\_power(i, sd);

+ }

+

+#ifdef CONFIG\_NUMA

+ for (i = 0; i < MAX\_NUMNODES; i++)

+ init\_numa\_sched\_groups\_power(sched\_group\_nodes[i]);

+

+ if (sd\_allnodes) {

+ struct sched\_group \*sg;

+

+ cpu\_to\_allnodes\_group(first\_cpu(\*cpu\_map), cpu\_map, &sg);

+ init\_numa\_sched\_groups\_power(sg);

+ }

+#endif

+

+ /\* Attach the domains \*/

+ for\_each\_cpu\_mask(i, \*cpu\_map) {

+ struct sched\_domain \*sd;

+#ifdef CONFIG\_SCHED\_SMT

+ sd = &per\_cpu(cpu\_domains, i);

+#elif defined(CONFIG\_SCHED\_MC)

+ sd = &per\_cpu(core\_domains, i);

+#else

+ sd = &per\_cpu(phys\_domains, i);

+#endif

+ cpu\_attach\_domain(sd, i);

+ }

+

+ return 0;

+

+#ifdef CONFIG\_NUMA

+error:

+ free\_sched\_groups(cpu\_map);

+ return -ENOMEM;

+#endif

+}

+

+static cpumask\_t \*doms\_cur; /\* current sched domains \*/

+static int ndoms\_cur; /\* number of sched domains in 'doms\_cur' \*/

+

+/\*

+ \* Special case: If a kmalloc of a doms\_cur partition (array of

+ \* cpumask\_t) fails, then fallback to a single sched domain,

+ \* as determined by the single cpumask\_t fallback\_doms.

+ \*/

+static cpumask\_t fallback\_doms;

+

+/\*

+ \* Set up scheduler domains and groups. Callers must hold the hotplug lock.

+ \* For now this just excludes isolated cpus, but could be used to

+ \* exclude other special cases in the future.

+ \*/

+static int arch\_init\_sched\_domains(const cpumask\_t \*cpu\_map)

+{

+ int err;

+

+ ndoms\_cur = 1;

+ doms\_cur = kmalloc(sizeof(cpumask\_t), GFP\_KERNEL);

+ if (!doms\_cur)

+ doms\_cur = &fallback\_doms;

+ cpus\_andnot(\*doms\_cur, \*cpu\_map, cpu\_isolated\_map);

+ err = build\_sched\_domains(doms\_cur);

+ register\_sched\_domain\_sysctl();

+

+ return err;

+}

+

+static void arch\_destroy\_sched\_domains(const cpumask\_t \*cpu\_map)

+{

+ free\_sched\_groups(cpu\_map);

+}

+

+/\*

+ \* Detach sched domains from a group of cpus specified in cpu\_map

+ \* These cpus will now be attached to the NULL domain

+ \*/

+static void detach\_destroy\_domains(const cpumask\_t \*cpu\_map)

+{

+ int i;

+

+ unregister\_sched\_domain\_sysctl();

+

+ for\_each\_cpu\_mask(i, \*cpu\_map)

+ cpu\_attach\_domain(NULL, i);

+ synchronize\_sched();

+ arch\_destroy\_sched\_domains(cpu\_map);

+}

+

+/\*

+ \* Partition sched domains as specified by the 'ndoms\_new'

+ \* cpumasks in the array doms\_new[] of cpumasks. This compares

+ \* doms\_new[] to the current sched domain partitioning, doms\_cur[].

+ \* It destroys each deleted domain and builds each new domain.

+ \*

+ \* 'doms\_new' is an array of cpumask\_t's of length 'ndoms\_new'.

+ \* The masks don't intersect (don't overlap.) We should setup one

+ \* sched domain for each mask. CPUs not in any of the cpumasks will

+ \* not be load balanced. If the same cpumask appears both in the

+ \* current 'doms\_cur' domains and in the new 'doms\_new', we can leave

+ \* it as it is.

+ \*

+ \* The passed in 'doms\_new' should be kmalloc'd. This routine takes

+ \* ownership of it and will kfree it when done with it. If the caller

+ \* failed the kmalloc call, then it can pass in doms\_new == NULL,

+ \* and partition\_sched\_domains() will fallback to the single partition

+ \* 'fallback\_doms'.

+ \*

+ \* Call with hotplug lock held

+ \*/

+void partition\_sched\_domains(int ndoms\_new, cpumask\_t \*doms\_new)

+{

+ int i, j;

+

+ /\* always unregister in case we don't destroy any domains \*/

+ unregister\_sched\_domain\_sysctl();

+

+ if (doms\_new == NULL) {

+ ndoms\_new = 1;

+ doms\_new = &fallback\_doms;

+ cpus\_andnot(doms\_new[0], cpu\_online\_map, cpu\_isolated\_map);

+ }

+

+ /\* Destroy deleted domains \*/

+ for (i = 0; i < ndoms\_cur; i++) {

+ for (j = 0; j < ndoms\_new; j++) {

+ if (cpus\_equal(doms\_cur[i], doms\_new[j]))

+ goto match1;

+ }

+ /\* no match - a current sched domain not in new doms\_new[] \*/

+ detach\_destroy\_domains(doms\_cur + i);

+match1:

+ ;

+ }

+

+ /\* Build new domains \*/

+ for (i = 0; i < ndoms\_new; i++) {

+ for (j = 0; j < ndoms\_cur; j++) {

+ if (cpus\_equal(doms\_new[i], doms\_cur[j]))

+ goto match2;

+ }

+ /\* no match - add a new doms\_new \*/

+ build\_sched\_domains(doms\_new + i);

+match2:

+ ;

+ }

+

+ /\* Remember the new sched domains \*/

+ if (doms\_cur != &fallback\_doms)

+ kfree(doms\_cur);

+ doms\_cur = doms\_new;

+ ndoms\_cur = ndoms\_new;

+

+ register\_sched\_domain\_sysctl();

+}

+

+#if defined(CONFIG\_SCHED\_MC) || defined(CONFIG\_SCHED\_SMT)

+static int arch\_reinit\_sched\_domains(void)

+{

+ int err;

+

+ mutex\_lock(&sched\_hotcpu\_mutex);

+ detach\_destroy\_domains(&cpu\_online\_map);

+ err = arch\_init\_sched\_domains(&cpu\_online\_map);

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+

+ return err;

+}

+

+static ssize\_t sched\_power\_savings\_store(const char \*buf, size\_t count, int smt)

+{

+ int ret;

+

+ if (buf[0] != '0' && buf[0] != '1')

+ return -EINVAL;

+

+ if (smt)

+ sched\_smt\_power\_savings = (buf[0] == '1');

+ else

+ sched\_mc\_power\_savings = (buf[0] == '1');

+

+ ret = arch\_reinit\_sched\_domains();

+

+ return ret ? ret : count;

+}

+

+#ifdef CONFIG\_SCHED\_MC

+static ssize\_t sched\_mc\_power\_savings\_show(struct sys\_device \*dev, char \*page)

+{

+ return sprintf(page, "%u\n", sched\_mc\_power\_savings);

+}

+static ssize\_t sched\_mc\_power\_savings\_store(struct sys\_device \*dev,

+ const char \*buf, size\_t count)

+{

+ return sched\_power\_savings\_store(buf, count, 0);

+}

+static SYSDEV\_ATTR(sched\_mc\_power\_savings, 0644, sched\_mc\_power\_savings\_show,

+ sched\_mc\_power\_savings\_store);

+#endif

+

+#ifdef CONFIG\_SCHED\_SMT

+static ssize\_t sched\_smt\_power\_savings\_show(struct sys\_device \*dev, char \*page)

+{

+ return sprintf(page, "%u\n", sched\_smt\_power\_savings);

+}

+static ssize\_t sched\_smt\_power\_savings\_store(struct sys\_device \*dev,

+ const char \*buf, size\_t count)

+{

+ return sched\_power\_savings\_store(buf, count, 1);

+}

+static SYSDEV\_ATTR(sched\_smt\_power\_savings, 0644, sched\_smt\_power\_savings\_show,

+ sched\_smt\_power\_savings\_store);

+#endif

+

+int sched\_create\_sysfs\_power\_savings\_entries(struct sysdev\_class \*cls)

+{

+ int err = 0;

+

+#ifdef CONFIG\_SCHED\_SMT

+ if (smt\_capable())

+ err = sysfs\_create\_file(&cls->kset.kobj,

+ &attr\_sched\_smt\_power\_savings.attr);

+#endif

+#ifdef CONFIG\_SCHED\_MC

+ if (!err && mc\_capable())

+ err = sysfs\_create\_file(&cls->kset.kobj,

+ &attr\_sched\_mc\_power\_savings.attr);

+#endif

+ return err;

+}

+#endif

+

+/\*

+ \* Force a reinitialization of the sched domains hierarchy. The domains

+ \* and groups cannot be updated in place without racing with the balancing

+ \* code, so we temporarily attach all running cpus to the NULL domain

+ \* which will prevent rebalancing while the sched domains are recalculated.

+ \*/

+static int update\_sched\_domains(struct notifier\_block \*nfb,

+ unsigned long action, void \*hcpu)

+{

+ switch (action) {

+ case CPU\_UP\_PREPARE:

+ case CPU\_UP\_PREPARE\_FROZEN:

+ case CPU\_DOWN\_PREPARE:

+ case CPU\_DOWN\_PREPARE\_FROZEN:

+ detach\_destroy\_domains(&cpu\_online\_map);

+ return NOTIFY\_OK;

+

+ case CPU\_UP\_CANCELED:

+ case CPU\_UP\_CANCELED\_FROZEN:

+ case CPU\_DOWN\_FAILED:

+ case CPU\_DOWN\_FAILED\_FROZEN:

+ case CPU\_ONLINE:

+ case CPU\_ONLINE\_FROZEN:

+ case CPU\_DEAD:

+ case CPU\_DEAD\_FROZEN:

+ /\*

+ \* Fall through and re-initialise the domains.

+ \*/

+ break;

+ default:

+ return NOTIFY\_DONE;

+ }

+

+ /\* The hotplug lock is already held by cpu\_up/cpu\_down \*/

+ arch\_init\_sched\_domains(&cpu\_online\_map);

+

+ return NOTIFY\_OK;

+}

+

+void \_\_init sched\_init\_smp(void)

+{

+ cpumask\_t non\_isolated\_cpus;

+

+ mutex\_lock(&sched\_hotcpu\_mutex);

+ arch\_init\_sched\_domains(&cpu\_online\_map);

+ cpus\_andnot(non\_isolated\_cpus, cpu\_possible\_map, cpu\_isolated\_map);

+ if (cpus\_empty(non\_isolated\_cpus))

+ cpu\_set(smp\_processor\_id(), non\_isolated\_cpus);

+ mutex\_unlock(&sched\_hotcpu\_mutex);

+ /\* XXX: Theoretical race here - CPU may be hotplugged now \*/

+ hotcpu\_notifier(update\_sched\_domains, 0);

+

+ /\* Move init over to a non-isolated CPU \*/

+ if (set\_cpus\_allowed(current, non\_isolated\_cpus) < 0)

+ BUG();

+ sched\_init\_granularity();

+}

+#else

+void \_\_init sched\_init\_smp(void)

+{

+ sched\_init\_granularity();

+}

+#endif /\* CONFIG\_SMP \*/

+

+int in\_sched\_functions(unsigned long addr)

+{

+ return in\_lock\_functions(addr) ||

+ (addr >= (unsigned long)\_\_sched\_text\_start

+ && addr < (unsigned long)\_\_sched\_text\_end);

+}

+

+static void init\_cfs\_rq(struct cfs\_rq \*cfs\_rq, struct rq \*rq)

+{

+ cfs\_rq->tasks\_timeline = RB\_ROOT;

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+ cfs\_rq->rq = rq;

+#endif

+ cfs\_rq->min\_vruntime = (u64)(-(1LL << 20));

+}

+

+void \_\_init sched\_init(void)

+{

+ int highest\_cpu = 0;

+ int i, j;

+

+ for\_each\_possible\_cpu(i) {

+ struct rt\_prio\_array \*array;

+ struct rq \*rq;

+

+ rq = cpu\_rq(i);

+ spin\_lock\_init(&rq->lock);

+ lockdep\_set\_class(&rq->lock, &rq->rq\_lock\_key);

+ rq->nr\_running = 0;

+ rq->clock = 1;

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+ init\_casio\_rq(&rq->casio\_rq);

+#endif

+ init\_cfs\_rq(&rq->cfs, rq);

+

+

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+ INIT\_LIST\_HEAD(&rq->leaf\_cfs\_rq\_list);

+ {

+ struct cfs\_rq \*cfs\_rq = &per\_cpu(init\_cfs\_rq, i);

+ struct sched\_entity \*se =

+ &per\_cpu(init\_sched\_entity, i);

+

+ init\_cfs\_rq\_p[i] = cfs\_rq;

+ init\_cfs\_rq(cfs\_rq, rq);

+ cfs\_rq->tg = &init\_task\_group;

+ list\_add(&cfs\_rq->leaf\_cfs\_rq\_list,

+ &rq->leaf\_cfs\_rq\_list);

+

+ init\_sched\_entity\_p[i] = se;

+ se->cfs\_rq = &rq->cfs;

+ se->my\_q = cfs\_rq;

+ se->load.weight = init\_task\_group\_load;

+ se->load.inv\_weight =

+ div64\_64(1ULL<<32, init\_task\_group\_load);

+ se->parent = NULL;

+ }

+

+ init\_task\_group.shares = init\_task\_group\_load;

+ spin\_lock\_init(&init\_task\_group.lock);

+#endif

+

+#ifdef CONFIG\_SCHED\_CASIO\_POLICY

+

+ init\_casio\_event\_log();

+#endif

+

+ for (j = 0; j < CPU\_LOAD\_IDX\_MAX; j++)

+ rq->cpu\_load[j] = 0;

+#ifdef CONFIG\_SMP

+ rq->sd = NULL;

+ rq->active\_balance = 0;

+ rq->next\_balance = jiffies;

+ rq->push\_cpu = 0;

+ rq->cpu = i;

+ rq->migration\_thread = NULL;

+ INIT\_LIST\_HEAD(&rq->migration\_queue);

+#endif

+ atomic\_set(&rq->nr\_iowait, 0);

+

+ array = &rq->rt.active;

+ for (j = 0; j < MAX\_RT\_PRIO; j++) {

+ INIT\_LIST\_HEAD(array->queue + j);

+ \_\_clear\_bit(j, array->bitmap);

+ }

+ highest\_cpu = i;

+ /\* delimiter for bitsearch: \*/

+ \_\_set\_bit(MAX\_RT\_PRIO, array->bitmap);

+ }

+

+ set\_load\_weight(&init\_task);

+

+#ifdef CONFIG\_PREEMPT\_NOTIFIERS

+ INIT\_HLIST\_HEAD(&init\_task.preempt\_notifiers);

+#endif

+

+#ifdef CONFIG\_SMP

+ nr\_cpu\_ids = highest\_cpu + 1;

+ open\_softirq(SCHED\_SOFTIRQ, run\_rebalance\_domains, NULL);

+#endif

+

+#ifdef CONFIG\_RT\_MUTEXES

+ plist\_head\_init(&init\_task.pi\_waiters, &init\_task.pi\_lock);

+#endif

+

+ /\*

+ \* The boot idle thread does lazy MMU switching as well:

+ \*/

+ atomic\_inc(&init\_mm.mm\_count);

+ enter\_lazy\_tlb(&init\_mm, current);

+

+ /\*

+ \* Make us the idle thread. Technically, schedule() should not be

+ \* called from this thread, however somewhere below it might be,

+ \* but because we are the idle thread, we just pick up running again

+ \* when this runqueue becomes "idle".

+ \*/

+ init\_idle(current, smp\_processor\_id());

+ /\*

+ \* During early bootup we pretend to be a normal task:

+ \*/

+ current->sched\_class = &fair\_sched\_class;

+}

+

+#ifdef CONFIG\_DEBUG\_SPINLOCK\_SLEEP

+void \_\_might\_sleep(char \*file, int line)

+{

+#ifdef in\_atomic

+ static unsigned long prev\_jiffy; /\* ratelimiting \*/

+

+ if ((in\_atomic() || irqs\_disabled()) &&

+ system\_state == SYSTEM\_RUNNING && !oops\_in\_progress) {

+ if (time\_before(jiffies, prev\_jiffy + HZ) && prev\_jiffy)

+ return;

+ prev\_jiffy = jiffies;

+ printk(KERN\_ERR "BUG: sleeping function called from invalid"

+ " context at %s:%d\n", file, line);

+ printk("in\_atomic():%d, irqs\_disabled():%d\n",

+ in\_atomic(), irqs\_disabled());

+ debug\_show\_held\_locks(current);

+ if (irqs\_disabled())

+ print\_irqtrace\_events(current);

+ dump\_stack();

+ }

+#endif

+}

+EXPORT\_SYMBOL(\_\_might\_sleep);

+#endif

+

+#ifdef CONFIG\_MAGIC\_SYSRQ

+static void normalize\_task(struct rq \*rq, struct task\_struct \*p)

+{

+ int on\_rq;

+ update\_rq\_clock(rq);

+ on\_rq = p->se.on\_rq;

+ if (on\_rq)

+ deactivate\_task(rq, p, 0);

+ \_\_setscheduler(rq, p, SCHED\_NORMAL, 0);

+ if (on\_rq) {

+ activate\_task(rq, p, 0);

+ resched\_task(rq->curr);

+ }

+}

+

+void normalize\_rt\_tasks(void)

+{

+ struct task\_struct \*g, \*p;

+ unsigned long flags;

+ struct rq \*rq;

+

+ read\_lock\_irq(&tasklist\_lock);

+ do\_each\_thread(g, p) {

+ /\*

+ \* Only normalize user tasks:

+ \*/

+ if (!p->mm)

+ continue;

+

+ p->se.exec\_start = 0;

+#ifdef CONFIG\_SCHEDSTATS

+ p->se.wait\_start = 0;

+ p->se.sleep\_start = 0;

+ p->se.block\_start = 0;

+#endif

+ task\_rq(p)->clock = 0;

+

+ if (!rt\_task(p)) {

+ /\*

+ \* Renice negative nice level userspace

+ \* tasks back to 0:

+ \*/

+ if (TASK\_NICE(p) < 0 && p->mm)

+ set\_user\_nice(p, 0);

+ continue;

+ }

+

+ spin\_lock\_irqsave(&p->pi\_lock, flags);

+ rq = \_\_task\_rq\_lock(p);

+

+ normalize\_task(rq, p);

+

+ \_\_task\_rq\_unlock(rq);

+ spin\_unlock\_irqrestore(&p->pi\_lock, flags);

+ } while\_each\_thread(g, p);

+

+ read\_unlock\_irq(&tasklist\_lock);

+}

+

+#endif /\* CONFIG\_MAGIC\_SYSRQ \*/

+

+#ifdef CONFIG\_IA64

+/\*

+ \* These functions are only useful for the IA64 MCA handling.

+ \*

+ \* They can only be called when the whole system has been

+ \* stopped - every CPU needs to be quiescent, and no scheduling

+ \* activity can take place. Using them for anything else would

+ \* be a serious bug, and as a result, they aren't even visible

+ \* under any other configuration.

+ \*/

+

+/\*\*

+ \* curr\_task - return the current task for a given cpu.

+ \* @cpu: the processor in question.

+ \*

+ \* ONLY VALID WHEN THE WHOLE SYSTEM IS STOPPED!

+ \*/

+struct task\_struct \*curr\_task(int cpu)

+{

+ return cpu\_curr(cpu);

+}

+

+/\*\*

+ \* set\_curr\_task - set the current task for a given cpu.

+ \* @cpu: the processor in question.

+ \* @p: the task pointer to set.

+ \*

+ \* Description: This function must only be used when non-maskable interrupts

+ \* are serviced on a separate stack. It allows the architecture to switch the

+ \* notion of the current task on a cpu in a non-blocking manner. This function

+ \* must be called with all CPU's synchronized, and interrupts disabled, the

+ \* and caller must save the original value of the current task (see

+ \* curr\_task() above) and restore that value before reenabling interrupts and

+ \* re-starting the system.

+ \*

+ \* ONLY VALID WHEN THE WHOLE SYSTEM IS STOPPED!

+ \*/

+void set\_curr\_task(int cpu, struct task\_struct \*p)

+{

+ cpu\_curr(cpu) = p;

+}

+

+#endif

+

+#ifdef CONFIG\_FAIR\_GROUP\_SCHED

+

+/\* allocate runqueue etc for a new task group \*/

+struct task\_group \*sched\_create\_group(void)

+{

+ struct task\_group \*tg;

+ struct cfs\_rq \*cfs\_rq;

+ struct sched\_entity \*se;

+ struct rq \*rq;

+ int i;

+

+ tg = kzalloc(sizeof(\*tg), GFP\_KERNEL);

+ if (!tg)

+ return ERR\_PTR(-ENOMEM);

+

+ tg->cfs\_rq = kzalloc(sizeof(cfs\_rq) \* NR\_CPUS, GFP\_KERNEL);

+ if (!tg->cfs\_rq)

+ goto err;

+ tg->se = kzalloc(sizeof(se) \* NR\_CPUS, GFP\_KERNEL);

+ if (!tg->se)

+ goto err;

+

+ for\_each\_possible\_cpu(i) {

+ rq = cpu\_rq(i);

+

+ cfs\_rq = kmalloc\_node(sizeof(struct cfs\_rq), GFP\_KERNEL,

+ cpu\_to\_node(i));

+ if (!cfs\_rq)

+ goto err;

+

+ se = kmalloc\_node(sizeof(struct sched\_entity), GFP\_KERNEL,

+ cpu\_to\_node(i));

+ if (!se)

+ goto err;

+

+ memset(cfs\_rq, 0, sizeof(struct cfs\_rq));

+ memset(se, 0, sizeof(struct sched\_entity));

+

+ tg->cfs\_rq[i] = cfs\_rq;

+ init\_cfs\_rq(cfs\_rq, rq);

+ cfs\_rq->tg = tg;

+

+ tg->se[i] = se;

+ se->cfs\_rq = &rq->cfs;

+ se->my\_q = cfs\_rq;

+ se->load.weight = NICE\_0\_LOAD;

+ se->load.inv\_weight = div64\_64(1ULL<<32, NICE\_0\_LOAD);

+ se->parent = NULL;

+ }

+

+ for\_each\_possible\_cpu(i) {

+ rq = cpu\_rq(i);

+ cfs\_rq = tg->cfs\_rq[i];

+ list\_add\_rcu(&cfs\_rq->leaf\_cfs\_rq\_list, &rq->leaf\_cfs\_rq\_list);

+ }

+

+ tg->shares = NICE\_0\_LOAD;

+ spin\_lock\_init(&tg->lock);

+

+ return tg;

+

+err:

+ for\_each\_possible\_cpu(i) {

+ if (tg->cfs\_rq)

+ kfree(tg->cfs\_rq[i]);

+ if (tg->se)

+ kfree(tg->se[i]);

+ }

+ kfree(tg->cfs\_rq);

+ kfree(tg->se);

+ kfree(tg);

+

+ return ERR\_PTR(-ENOMEM);

+}

+

+/\* rcu callback to free various structures associated with a task group \*/

+static void free\_sched\_group(struct rcu\_head \*rhp)

+{

+ struct task\_group \*tg = container\_of(rhp, struct task\_group, rcu);

+ struct cfs\_rq \*cfs\_rq;

+ struct sched\_entity \*se;

+ int i;

+

+ /\* now it should be safe to free those cfs\_rqs \*/

+ for\_each\_possible\_cpu(i) {

+ cfs\_rq = tg->cfs\_rq[i];

+ kfree(cfs\_rq);

+

+ se = tg->se[i];

+ kfree(se);

+ }

+

+ kfree(tg->cfs\_rq);

+ kfree(tg->se);

+ kfree(tg);

+}

+

+/\* Destroy runqueue etc associated with a task group \*/

+void sched\_destroy\_group(struct task\_group \*tg)

+{

+ struct cfs\_rq \*cfs\_rq = NULL;

+ int i;

+

+ for\_each\_possible\_cpu(i) {

+ cfs\_rq = tg->cfs\_rq[i];

+ list\_del\_rcu(&cfs\_rq->leaf\_cfs\_rq\_list);

+ }

+

+ BUG\_ON(!cfs\_rq);

+

+ /\* wait for possible concurrent references to cfs\_rqs complete \*/

+ call\_rcu(&tg->rcu, free\_sched\_group);

+}

+

+/\* change task's runqueue when it moves between groups.

+ \* The caller of this function should have put the task in its new group

+ \* by now. This function just updates tsk->se.cfs\_rq and tsk->se.parent to

+ \* reflect its new group.

+ \*/

+void sched\_move\_task(struct task\_struct \*tsk)

+{

+ int on\_rq, running;

+ unsigned long flags;

+ struct rq \*rq;

+

+ rq = task\_rq\_lock(tsk, &flags);

+

+ if (tsk->sched\_class != &fair\_sched\_class) {

+ set\_task\_cfs\_rq(tsk, task\_cpu(tsk));

+ goto done;

+ }

+

+ update\_rq\_clock(rq);

+

+ running = task\_current(rq, tsk);

+ on\_rq = tsk->se.on\_rq;

+

+ if (on\_rq) {

+ dequeue\_task(rq, tsk, 0);

+ if (unlikely(running))

+ tsk->sched\_class->put\_prev\_task(rq, tsk);

+ }

+

+ set\_task\_cfs\_rq(tsk, task\_cpu(tsk));

+

+ if (on\_rq) {

+ if (unlikely(running))

+ tsk->sched\_class->set\_curr\_task(rq);

+ enqueue\_task(rq, tsk, 0);

+ }

+

+done:

+ task\_rq\_unlock(rq, &flags);

+}

+

+static void set\_se\_shares(struct sched\_entity \*se, unsigned long shares)

+{

+ struct cfs\_rq \*cfs\_rq = se->cfs\_rq;

+ struct rq \*rq = cfs\_rq->rq;

+ int on\_rq;

+

+ spin\_lock\_irq(&rq->lock);

+

+ on\_rq = se->on\_rq;

+ if (on\_rq)

+ dequeue\_entity(cfs\_rq, se, 0);

+

+ se->load.weight = shares;

+ se->load.inv\_weight = div64\_64((1ULL<<32), shares);

+

+ if (on\_rq)

+ enqueue\_entity(cfs\_rq, se, 0);

+

+ spin\_unlock\_irq(&rq->lock);

+}

+

+int sched\_group\_set\_shares(struct task\_group \*tg, unsigned long shares)

+{

+ int i;

+

+ /\*

+ \* A weight of 0 or 1 can cause arithmetics problems.

+ \* (The default weight is 1024 - so there's no practical

+ \* limitation from this.)

+ \*/

+ if (shares < 2)

+ shares = 2;

+

+ spin\_lock(&tg->lock);

+ if (tg->shares == shares)

+ goto done;

+

+ tg->shares = shares;

+ for\_each\_possible\_cpu(i)

+ set\_se\_shares(tg->se[i], shares);

+

+done:

+ spin\_unlock(&tg->lock);

+ return 0;

+}

+

+unsigned long sched\_group\_shares(struct task\_group \*tg)

+{

+ return tg->shares;

+}

+

+#endif /\* CONFIG\_FAIR\_GROUP\_SCHED \*/

+

+#ifdef CONFIG\_FAIR\_CGROUP\_SCHED

+

+/\* return corresponding task\_group object of a cgroup \*/

+static inline struct task\_group \*cgroup\_tg(struct cgroup \*cgrp)

+{

+ return container\_of(cgroup\_subsys\_state(cgrp, cpu\_cgroup\_subsys\_id),

+ struct task\_group, css);

+}

+

+static struct cgroup\_subsys\_state \*

+cpu\_cgroup\_create(struct cgroup\_subsys \*ss, struct cgroup \*cgrp)

+{

+ struct task\_group \*tg;

+

+ if (!cgrp->parent) {

+ /\* This is early initialization for the top cgroup \*/

+ init\_task\_group.css.cgroup = cgrp;

+ return &init\_task\_group.css;

+ }

+

+ /\* we support only 1-level deep hierarchical scheduler atm \*/

+ if (cgrp->parent->parent)

+ return ERR\_PTR(-EINVAL);

+

+ tg = sched\_create\_group();

+ if (IS\_ERR(tg))

+ return ERR\_PTR(-ENOMEM);

+

+ /\* Bind the cgroup to task\_group object we just created \*/

+ tg->css.cgroup = cgrp;

+

+ return &tg->css;

+}

+

+static void

+cpu\_cgroup\_destroy(struct cgroup\_subsys \*ss, struct cgroup \*cgrp)

+{

+ struct task\_group \*tg = cgroup\_tg(cgrp);

+

+ sched\_destroy\_group(tg);

+}

+

+static int

+cpu\_cgroup\_can\_attach(struct cgroup\_subsys \*ss, struct cgroup \*cgrp,

+ struct task\_struct \*tsk)

+{

+ /\* We don't support RT-tasks being in separate groups \*/

+ if (tsk->sched\_class != &fair\_sched\_class)

+ return -EINVAL;

+

+ return 0;

+}

+

+static void

+cpu\_cgroup\_attach(struct cgroup\_subsys \*ss, struct cgroup \*cgrp,

+ struct cgroup \*old\_cont, struct task\_struct \*tsk)

+{

+ sched\_move\_task(tsk);

+}

+

+static int cpu\_shares\_write\_uint(struct cgroup \*cgrp, struct cftype \*cftype,

+ u64 shareval)

+{

+ return sched\_group\_set\_shares(cgroup\_tg(cgrp), shareval);

+}

+

+static u64 cpu\_shares\_read\_uint(struct cgroup \*cgrp, struct cftype \*cft)

+{

+ struct task\_group \*tg = cgroup\_tg(cgrp);

+

+ return (u64) tg->shares;

+}

+

+static struct cftype cpu\_files[] = {

+ {

+ .name = "shares",

+ .read\_uint = cpu\_shares\_read\_uint,

+ .write\_uint = cpu\_shares\_write\_uint,

+ },

+};

+

+static int cpu\_cgroup\_populate(struct cgroup\_subsys \*ss, struct cgroup \*cont)

+{

+ return cgroup\_add\_files(cont, ss, cpu\_files, ARRAY\_SIZE(cpu\_files));

+}

+

+struct cgroup\_subsys cpu\_cgroup\_subsys = {

+ .name = "cpu",

+ .create = cpu\_cgroup\_create,

+ .destroy = cpu\_cgroup\_destroy,

+ .can\_attach = cpu\_cgroup\_can\_attach,

+ .attach = cpu\_cgroup\_attach,

+ .populate = cpu\_cgroup\_populate,

+ .subsys\_id = cpu\_cgroup\_subsys\_id,

+ .early\_init = 1,

+};

+

+#endif /\* CONFIG\_FAIR\_CGROUP\_SCHED \*/

+

+#ifdef CONFIG\_CGROUP\_CPUACCT

+

+/\*

+ \* CPU accounting code for task groups.

+ \*

+ \* Based on the work by Paul Menage (menage@google.com) and Balbir Singh

+ \* (balbir@in.ibm.com).

+ \*/

+

+/\* track cpu usage of a group of tasks \*/

+struct cpuacct {

+ struct cgroup\_subsys\_state css;

+ /\* cpuusage holds pointer to a u64-type object on every cpu \*/

+ u64 \*cpuusage;

+};

+

+struct cgroup\_subsys cpuacct\_subsys;

+

+/\* return cpu accounting group corresponding to this container \*/

+static inline struct cpuacct \*cgroup\_ca(struct cgroup \*cont)

+{

+ return container\_of(cgroup\_subsys\_state(cont, cpuacct\_subsys\_id),

+ struct cpuacct, css);

+}

+

+/\* return cpu accounting group to which this task belongs \*/

+static inline struct cpuacct \*task\_ca(struct task\_struct \*tsk)

+{

+ return container\_of(task\_subsys\_state(tsk, cpuacct\_subsys\_id),

+ struct cpuacct, css);

+}

+

+/\* create a new cpu accounting group \*/

+static struct cgroup\_subsys\_state \*cpuacct\_create(

+ struct cgroup\_subsys \*ss, struct cgroup \*cont)

+{

+ struct cpuacct \*ca = kzalloc(sizeof(\*ca), GFP\_KERNEL);

+

+ if (!ca)

+ return ERR\_PTR(-ENOMEM);

+

+ ca->cpuusage = alloc\_percpu(u64);

+ if (!ca->cpuusage) {

+ kfree(ca);

+ return ERR\_PTR(-ENOMEM);

+ }

+

+ return &ca->css;

+}

+

+/\* destroy an existing cpu accounting group \*/

+static void

+cpuacct\_destroy(struct cgroup\_subsys \*ss, struct cgroup \*cont)

+{

+ struct cpuacct \*ca = cgroup\_ca(cont);

+

+ free\_percpu(ca->cpuusage);

+ kfree(ca);

+}

+

+/\* return total cpu usage (in nanoseconds) of a group \*/

+static u64 cpuusage\_read(struct cgroup \*cont, struct cftype \*cft)

+{

+ struct cpuacct \*ca = cgroup\_ca(cont);

+ u64 totalcpuusage = 0;

+ int i;

+

+ for\_each\_possible\_cpu(i) {

+ u64 \*cpuusage = percpu\_ptr(ca->cpuusage, i);

+

+ /\*

+ \* Take rq->lock to make 64-bit addition safe on 32-bit

+ \* platforms.

+ \*/

+ spin\_lock\_irq(&cpu\_rq(i)->lock);

+ totalcpuusage += \*cpuusage;

+ spin\_unlock\_irq(&cpu\_rq(i)->lock);

+ }

+

+ return totalcpuusage;

+}

+

+static struct cftype files[] = {

+ {

+ .name = "usage",

+ .read\_uint = cpuusage\_read,

+ },

+};

+

+static int cpuacct\_populate(struct cgroup\_subsys \*ss, struct cgroup \*cont)

+{

+ return cgroup\_add\_files(cont, ss, files, ARRAY\_SIZE(files));

+}

+

+/\*

+ \* charge this task's execution time to its accounting group.

+ \*

+ \* called with rq->lock held.

+ \*/

+static void cpuacct\_charge(struct task\_struct \*tsk, u64 cputime)

+{

+ struct cpuacct \*ca;

+

+ if (!cpuacct\_subsys.active)

+ return;

+

+ ca = task\_ca(tsk);

+ if (ca) {

+ u64 \*cpuusage = percpu\_ptr(ca->cpuusage, task\_cpu(tsk));

+

+ \*cpuusage += cputime;

+ }

+}

+

+struct cgroup\_subsys cpuacct\_subsys = {

+ .name = "cpuacct",

+ .create = cpuacct\_create,

+ .destroy = cpuacct\_destroy,

+ .populate = cpuacct\_populate,

+ .subsys\_id = cpuacct\_subsys\_id,

+};

+#endif /\* CONFIG\_CGROUP\_CPUACCT \*/

diff -Nur linux-2.6.24/kernel/sched\_casio.c linux-2.6.24-casio/kernel/sched\_casio.c

--- linux-2.6.24/kernel/sched\_casio.c 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/kernel/sched\_casio.c 2009-06-29 06:57:19.000000000 +0100

@@ -0,0 +1,281 @@

+/\*

+ \* casio-task scheduling class.

+ \*

+ \*

+ \*/

+/\*

+ \* log functions.

+ \*/

+

+struct casio\_event\_log casio\_event\_log;

+

+struct casio\_event\_log \* get\_casio\_event\_log()

+{

+ return &casio\_event\_log;

+}

+void init\_casio\_event\_log()

+{

+ char msg[CASIO\_MSG\_SIZE];

+ casio\_event\_log.lines=casio\_event\_log.cursor=0;

+ snprintf(msg,CASIO\_MSG\_SIZE,"init\_casio\_event\_log:(%lu:%lu)", casio\_event\_log.lines, casio\_event\_log.cursor);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+

+}

+void register\_casio\_event(unsigned long long t, char \*m, int a)

+{

+

+ if(casio\_event\_log.lines < CASIO\_MAX\_EVENT\_LINES){

+ casio\_event\_log.casio\_event[casio\_event\_log.lines].action=a;

+ casio\_event\_log.casio\_event[casio\_event\_log.lines].timestamp=t;

+ strncpy(casio\_event\_log.casio\_event[casio\_event\_log.lines].msg,m,CASIO\_MSG\_SIZE-1);

+ casio\_event\_log.lines++;

+ }

+ else{

+ printk(KERN\_ALERT "register\_casio\_event: full\n");

+ }

+

+}

+/\*

+ \*casio tasks and casio rq

+ \*/

+void init\_casio\_rq(struct casio\_rq \*casio\_rq)

+{

+ casio\_rq->casio\_rb\_root=RB\_ROOT;

+ INIT\_LIST\_HEAD(&casio\_rq->casio\_list\_head);

+ atomic\_set(&casio\_rq->nr\_running,0);

+}

+void add\_casio\_task\_2\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL;

+ struct casio\_task \*new=NULL, \*casio\_task=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(rq && p){

+ new=(struct casio\_task \*) kzalloc(sizeof(struct casio\_task),GFP\_KERNEL);

+ if(new){

+ casio\_task=NULL;

+ new->task=p;

+ new->absolute\_deadline=0;

+ list\_for\_each(ptr,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(new->task->casio\_id < casio\_task->task->casio\_id){

+ list\_add(&new->casio\_list\_node,ptr);

+ }

+ }

+ }

+ list\_add(&new->casio\_list\_node,&rq->casio\_list\_head);

+ snprintf(msg,CASIO\_MSG\_SIZE,"add\_casio\_task\_2\_list: %d:%d:%llu",new->task->casio\_id,new->task->pid,new->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+ }

+ else{

+ printk(KERN\_ALERT "add\_casio\_task\_2\_list: kzalloc\n");

+ }

+ }

+ else{

+ printk(KERN\_ALERT "add\_casio\_task\_2\_list: null pointers\n");

+ }

+}

+struct casio\_task \* find\_casio\_task\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL;

+ struct casio\_task \*casio\_task=NULL;

+ if(rq && p){

+ list\_for\_each(ptr,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(casio\_task->task->casio\_id == p->casio\_id){

+ return casio\_task;

+ }

+ }

+ }

+ }

+ return NULL;

+}

+void rem\_casio\_task\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL,\*next=NULL;

+ struct casio\_task \*casio\_task=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(rq && p){

+ list\_for\_each\_safe(ptr,next,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(casio\_task->task->casio\_id == p->casio\_id){

+ list\_del(ptr);

+ snprintf(msg,CASIO\_MSG\_SIZE,"rem\_casio\_task\_list: %d:%d:%llu",casio\_task->task->casio\_id,casio\_task->task->pid,casio\_task->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+ kfree(casio\_task);

+ return;

+ }

+ }

+ }

+ }

+}

+/\*

+ \* rb\_tree functions.

+ \*/

+

+void remove\_casio\_task\_rb\_tree(struct casio\_rq \*rq, struct casio\_task \*p)

+{

+ rb\_erase(&(p->casio\_rb\_node),&(rq->casio\_rb\_root));

+ p->casio\_rb\_node.rb\_left=p->casio\_rb\_node.rb\_right=NULL;

+}

+void insert\_casio\_task\_rb\_tree(struct casio\_rq \*rq, struct casio\_task \*p)

+{

+ struct rb\_node \*\*node=NULL;

+ struct rb\_node \*parent=NULL;

+ struct casio\_task \*entry=NULL;

+ node=&rq->casio\_rb\_root.rb\_node;

+ while(\*node!=NULL){

+ parent=\*node;

+ entry=rb\_entry(parent, struct casio\_task,casio\_rb\_node);

+ if(entry){

+ if(p->absolute\_deadline < entry->absolute\_deadline){

+ node=&parent->rb\_left;

+ }else{

+ node=&parent->rb\_right;

+ }

+ }

+ }

+ rb\_link\_node(&p->casio\_rb\_node,parent,node);

+ rb\_insert\_color(&p->casio\_rb\_node,&rq->casio\_rb\_root);

+}

+struct casio\_task \* earliest\_deadline\_casio\_task\_rb\_tree(struct casio\_rq \*rq)

+{

+ struct rb\_node \*node=NULL;

+ struct casio\_task \*p=NULL;

+ node=rq->casio\_rb\_root.rb\_node;

+ if(node==NULL)

+ return NULL;

+

+ while(node->rb\_left!=NULL){

+ node=node->rb\_left;

+ }

+ p=rb\_entry(node, struct casio\_task,casio\_rb\_node);

+ return p;

+}

+

+/////////////////////////////

+static void check\_preempt\_curr\_casio(struct rq \*rq, struct task\_struct \*p)

+{

+ struct casio\_task \*t=NULL,\*curr=NULL;

+ if(rq->curr->policy!=SCHED\_CASIO){

+ resched\_task(rq->curr);

+ }

+ else{

+ t=earliest\_deadline\_casio\_task\_rb\_tree(&rq->casio\_rq);

+ if(t){

+ curr=find\_casio\_task\_list(&rq->casio\_rq,rq->curr);

+ if(curr){

+ if(t->absolute\_deadline < curr->absolute\_deadline)

+ resched\_task(rq->curr);

+ }

+ else{

+ printk(KERN\_ALERT "check\_preempt\_curr\_casio\n");

+ }

+ }

+ }

+

+}

+

+static struct task\_struct \*pick\_next\_task\_casio(struct rq \*rq)

+{

+

+ struct casio\_task \*t=NULL;

+ t=earliest\_deadline\_casio\_task\_rb\_tree(&rq->casio\_rq);

+ if(t){

+ return t->task;

+ }

+ return NULL;

+}

+static void enqueue\_task\_casio(struct rq \*rq, struct task\_struct \*p, int wakeup)

+{

+ struct casio\_task \*t=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(p){

+ t=find\_casio\_task\_list(&rq->casio\_rq,p);

+ if(t){

+ t->absolute\_deadline=sched\_clock()+p->deadline;

+ insert\_casio\_task\_rb\_tree(&rq->casio\_rq, t);

+ atomic\_inc(&rq->casio\_rq.nr\_running);

+ snprintf(msg,CASIO\_MSG\_SIZE,"(%d:%d:%llu)",p->casio\_id,p->pid,t->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_ENQUEUE);

+ }

+ else{

+ printk(KERN\_ALERT "enqueue\_task\_casio\n");

+ }

+ }

+}

+

+static void dequeue\_task\_casio(struct rq \*rq, struct task\_struct \*p, int sleep)

+{

+ struct casio\_task \*t=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(p){

+ t=find\_casio\_task\_list(&rq->casio\_rq,p);

+ if(t){

+ snprintf(msg,CASIO\_MSG\_SIZE,"(%d:%d:%llu)",t->task->casio\_id,t->task->pid,t->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_DEQUEUE);

+ remove\_casio\_task\_rb\_tree(&rq->casio\_rq, t);

+ atomic\_dec(&rq->casio\_rq.nr\_running);

+ if(t->task->state==TASK\_DEAD || t->task->state==EXIT\_DEAD || t->task->state==EXIT\_ZOMBIE){

+ rem\_casio\_task\_list(&rq->casio\_rq,t->task);

+ }

+ }

+ else{

+ printk(KERN\_ALERT "dequeue\_task\_casio\n");

+ }

+ }

+

+}

+

+static void put\_prev\_task\_casio(struct rq \*rq, struct task\_struct \*prev)

+{

+}

+

+#ifdef CONFIG\_SMP

+static unsigned long load\_balance\_casio(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ unsigned long max\_load\_move,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*all\_pinned, int \*this\_best\_prio)

+{

+ return 0;

+}

+

+static int move\_one\_task\_casio(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle)

+{

+ return 0;

+}

+#endif

+

+static void task\_tick\_casio(struct rq \*rq, struct task\_struct \*p)

+{

+ //check\_preempt\_curr\_casio(rq, p);

+}

+

+static void set\_curr\_task\_casio(struct rq \*rq)

+{

+}

+

+/\*

+ \* Simple, special scheduling class for the per-CPU casio tasks:

+ \*/

+const struct sched\_class casio\_sched\_class = {

+ .next = &rt\_sched\_class,

+ .enqueue\_task = enqueue\_task\_casio,

+ .dequeue\_task = dequeue\_task\_casio,

+

+ .check\_preempt\_curr = check\_preempt\_curr\_casio,

+

+ .pick\_next\_task = pick\_next\_task\_casio,

+ .put\_prev\_task = put\_prev\_task\_casio,

+

+#ifdef CONFIG\_SMP

+ .load\_balance = load\_balance\_casio,

+ .move\_one\_task = move\_one\_task\_casio,

+#endif

+

+ .set\_curr\_task = set\_curr\_task\_casio,

+ .task\_tick = task\_tick\_casio,

+};

diff -Nur linux-2.6.24/kernel/sched\_casio.c~ linux-2.6.24-casio/kernel/sched\_casio.c~

--- linux-2.6.24/kernel/sched\_casio.c~ 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/kernel/sched\_casio.c~ 2009-06-29 06:57:00.000000000 +0100

@@ -0,0 +1,281 @@

+/\*

+ \* casio-task scheduling class.

+ \*

+ \*

+ \*/

+/\*

+ \* log functions.

+ \*/

+

+struct casio\_event\_log casio\_event\_log;

+

+struct casio\_event\_log \* get\_casio\_event\_log()

+{

+ return &casio\_event\_log;

+}

+void init\_casio\_event\_log()

+{

+ char msg[CASIO\_MSG\_SIZE];

+ casio\_event\_log.lines=casio\_event\_log.cursor=0;

+ snprintf(msg,CASIO\_MSG\_SIZE,"init\_casio\_event\_log:(%lu:%lu)", casio\_event\_log.lines, casio\_event\_log.cursor);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+

+}

+void register\_casio\_event(unsigned long long t, char \*m, int a)

+{

+

+ if(casio\_event\_log.lines < CASIO\_MAX\_EVENT\_LINES){

+ casio\_event\_log.casio\_event[casio\_event\_log.lines].action=a;

+ casio\_event\_log.casio\_event[casio\_event\_log.lines].timestamp=t;

+ strncpy(casio\_event\_log.casio\_event[casio\_event\_log.lines].msg,m,CASIO\_MSG\_SIZE-1);

+ casio\_event\_log.lines++;

+ }

+ else{

+ printk(KERN\_ALERT "register\_casio\_event: full\n");

+ }

+

+}

+/\*

+ \*casio tasks and casio rq

+ \*/

+void init\_casio\_rq(struct casio\_rq \*casio\_rq)

+{

+ casio\_rq->casio\_rb\_root=RB\_ROOT;

+ INIT\_LIST\_HEAD(&casio\_rq->casio\_list\_head);

+ atomic\_set(&casio\_rq->nr\_running,0);

+}

+void add\_casio\_task\_2\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL;

+ struct casio\_task \*new=NULL, \*casio\_task=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(rq && p){

+ new=(struct casio\_task \*) kzalloc(sizeof(struct casio\_task),GFP\_KERNEL);

+ if(new){

+ casio\_task=NULL;

+ new->task=p;

+ new->absolute\_deadline=0;

+ list\_for\_each(ptr,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(casio\_task->task->casio\_id < casio\_task->task->casio\_id){

+ list\_add(&new->casio\_list\_node,ptr);

+ }

+ }

+ }

+ list\_add(&new->casio\_list\_node,&rq->casio\_list\_head);

+ snprintf(msg,CASIO\_MSG\_SIZE,"add\_casio\_task\_2\_list: %d:%d:%llu",new->task->casio\_id,new->task->pid,new->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+ }

+ else{

+ printk(KERN\_ALERT "add\_casio\_task\_2\_list: kzalloc\n");

+ }

+ }

+ else{

+ printk(KERN\_ALERT "add\_casio\_task\_2\_list: null pointers\n");

+ }

+}

+struct casio\_task \* find\_casio\_task\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL;

+ struct casio\_task \*casio\_task=NULL;

+ if(rq && p){

+ list\_for\_each(ptr,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(casio\_task->task->casio\_id == p->casio\_id){

+ return casio\_task;

+ }

+ }

+ }

+ }

+ return NULL;

+}

+void rem\_casio\_task\_list(struct casio\_rq \*rq, struct task\_struct \*p)

+{

+ struct list\_head \*ptr=NULL,\*next=NULL;

+ struct casio\_task \*casio\_task=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(rq && p){

+ list\_for\_each\_safe(ptr,next,&rq->casio\_list\_head){

+ casio\_task=list\_entry(ptr,struct casio\_task, casio\_list\_node);

+ if(casio\_task){

+ if(casio\_task->task->casio\_id == p->casio\_id){

+ list\_del(ptr);

+ snprintf(msg,CASIO\_MSG\_SIZE,"rem\_casio\_task\_list: %d:%d:%llu",casio\_task->task->casio\_id,casio\_task->task->pid,casio\_task->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_MSG);

+ kfree(casio\_task);

+ return;

+ }

+ }

+ }

+ }

+}

+/\*

+ \* rb\_tree functions.

+ \*/

+

+void remove\_casio\_task\_rb\_tree(struct casio\_rq \*rq, struct casio\_task \*p)

+{

+ rb\_erase(&(p->casio\_rb\_node),&(rq->casio\_rb\_root));

+ p->casio\_rb\_node.rb\_left=p->casio\_rb\_node.rb\_right=NULL;

+}

+void insert\_casio\_task\_rb\_tree(struct casio\_rq \*rq, struct casio\_task \*p)

+{

+ struct rb\_node \*\*node=NULL;

+ struct rb\_node \*parent=NULL;

+ struct casio\_task \*entry=NULL;

+ node=&rq->casio\_rb\_root.rb\_node;

+ while(\*node!=NULL){

+ parent=\*node;

+ entry=rb\_entry(parent, struct casio\_task,casio\_rb\_node);

+ if(entry){

+ if(p->absolute\_deadline < entry->absolute\_deadline){

+ node=&parent->rb\_left;

+ }else{

+ node=&parent->rb\_right;

+ }

+ }

+ }

+ rb\_link\_node(&p->casio\_rb\_node,parent,node);

+ rb\_insert\_color(&p->casio\_rb\_node,&rq->casio\_rb\_root);

+}

+struct casio\_task \* earliest\_deadline\_casio\_task\_rb\_tree(struct casio\_rq \*rq)

+{

+ struct rb\_node \*node=NULL;

+ struct casio\_task \*p=NULL;

+ node=rq->casio\_rb\_root.rb\_node;

+ if(node==NULL)

+ return NULL;

+

+ while(node->rb\_left!=NULL){

+ node=node->rb\_left;

+ }

+ p=rb\_entry(node, struct casio\_task,casio\_rb\_node);

+ return p;

+}

+

+/////////////////////////////

+static void check\_preempt\_curr\_casio(struct rq \*rq, struct task\_struct \*p)

+{

+ struct casio\_task \*t=NULL,\*curr=NULL;

+ if(rq->curr->policy!=SCHED\_CASIO){

+ resched\_task(rq->curr);

+ }

+ else{

+ t=earliest\_deadline\_casio\_task\_rb\_tree(&rq->casio\_rq);

+ if(t){

+ curr=find\_casio\_task\_list(&rq->casio\_rq,rq->curr);

+ if(curr){

+ if(t->absolute\_deadline < curr->absolute\_deadline)

+ resched\_task(rq->curr);

+ }

+ else{

+ printk(KERN\_ALERT "check\_preempt\_curr\_casio\n");

+ }

+ }

+ }

+

+}

+

+static struct task\_struct \*pick\_next\_task\_casio(struct rq \*rq)

+{

+

+ struct casio\_task \*t=NULL;

+ t=earliest\_deadline\_casio\_task\_rb\_tree(&rq->casio\_rq);

+ if(t){

+ return t->task;

+ }

+ return NULL;

+}

+static void enqueue\_task\_casio(struct rq \*rq, struct task\_struct \*p, int wakeup)

+{

+ struct casio\_task \*t=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(p){

+ t=find\_casio\_task\_list(&rq->casio\_rq,p);

+ if(t){

+ t->absolute\_deadline=sched\_clock()+p->deadline;

+ insert\_casio\_task\_rb\_tree(&rq->casio\_rq, t);

+ atomic\_inc(&rq->casio\_rq.nr\_running);

+ snprintf(msg,CASIO\_MSG\_SIZE,"(%d:%d:%llu)",p->casio\_id,p->pid,t->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_ENQUEUE);

+ }

+ else{

+ printk(KERN\_ALERT "enqueue\_task\_casio\n");

+ }

+ }

+}

+

+static void dequeue\_task\_casio(struct rq \*rq, struct task\_struct \*p, int sleep)

+{

+ struct casio\_task \*t=NULL;

+ char msg[CASIO\_MSG\_SIZE];

+ if(p){

+ t=find\_casio\_task\_list(&rq->casio\_rq,p);

+ if(t){

+ snprintf(msg,CASIO\_MSG\_SIZE,"(%d:%d:%llu)",t->task->casio\_id,t->task->pid,t->absolute\_deadline);

+ register\_casio\_event(sched\_clock(), msg, CASIO\_DEQUEUE);

+ remove\_casio\_task\_rb\_tree(&rq->casio\_rq, t);

+ atomic\_dec(&rq->casio\_rq.nr\_running);

+ if(t->task->state==TASK\_DEAD || t->task->state==EXIT\_DEAD || t->task->state==EXIT\_ZOMBIE){

+ rem\_casio\_task\_list(&rq->casio\_rq,t->task);

+ }

+ }

+ else{

+ printk(KERN\_ALERT "dequeue\_task\_casio\n");

+ }

+ }

+

+}

+

+static void put\_prev\_task\_casio(struct rq \*rq, struct task\_struct \*prev)

+{

+}

+

+#ifdef CONFIG\_SMP

+static unsigned long load\_balance\_casio(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ unsigned long max\_load\_move,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle,

+ int \*all\_pinned, int \*this\_best\_prio)

+{

+ return 0;

+}

+

+static int move\_one\_task\_casio(struct rq \*this\_rq, int this\_cpu, struct rq \*busiest,

+ struct sched\_domain \*sd, enum cpu\_idle\_type idle)

+{

+ return 0;

+}

+#endif

+

+static void task\_tick\_casio(struct rq \*rq, struct task\_struct \*p)

+{

+ //check\_preempt\_curr\_casio(rq, p);

+}

+

+static void set\_curr\_task\_casio(struct rq \*rq)

+{

+}

+

+/\*

+ \* Simple, special scheduling class for the per-CPU casio tasks:

+ \*/

+const struct sched\_class casio\_sched\_class = {

+ .next = &rt\_sched\_class,

+ .enqueue\_task = enqueue\_task\_casio,

+ .dequeue\_task = dequeue\_task\_casio,

+

+ .check\_preempt\_curr = check\_preempt\_curr\_casio,

+

+ .pick\_next\_task = pick\_next\_task\_casio,

+ .put\_prev\_task = put\_prev\_task\_casio,

+

+#ifdef CONFIG\_SMP

+ .load\_balance = load\_balance\_casio,

+ .move\_one\_task = move\_one\_task\_casio,

+#endif

+

+ .set\_curr\_task = set\_curr\_task\_casio,

+ .task\_tick = task\_tick\_casio,

+};

diff -Nur linux-2.6.24/Makefile linux-2.6.24-casio/Makefile

--- linux-2.6.24/Makefile 2008-01-24 22:58:37.000000000 +0000

+++ linux-2.6.24-casio/Makefile 2009-06-02 04:11:04.000000000 +0100

@@ -1,7 +1,7 @@

VERSION = 2

PATCHLEVEL = 6

SUBLEVEL = 24

-EXTRAVERSION =

+EXTRAVERSION = -casio

NAME = Arr Matey! A Hairy Bilge Rat!

# \*DOCUMENTATION\*

diff -Nur linux-2.6.24/Makefile~ linux-2.6.24-casio/Makefile~

--- linux-2.6.24/Makefile~ 1970-01-01 01:00:00.000000000 +0100

+++ linux-2.6.24-casio/Makefile~ 2009-06-02 04:11:04.000000000 +0100

@@ -0,0 +1,1566 @@

+VERSION = 2

+PATCHLEVEL = 6

+SUBLEVEL = 24

+EXTRAVERSION =

+NAME = Arr Matey! A Hairy Bilge Rat!

+

+# \*DOCUMENTATION\*

+# To see a list of typical targets execute "make help"

+# More info can be located in ./README

+# Comments in this file are targeted only to the developer, do not

+# expect to learn how to build the kernel reading this file.

+

+# Do not:

+# o use make's built-in rules and variables

+# (this increases performance and avoids hard-to-debug behaviour);

+# o print "Entering directory ...";

+MAKEFLAGS += -rR --no-print-directory

+

+# We are using a recursive build, so we need to do a little thinking

+# to get the ordering right.

+#

+# Most importantly: sub-Makefiles should only ever modify files in

+# their own directory. If in some directory we have a dependency on

+# a file in another dir (which doesn't happen often, but it's often

+# unavoidable when linking the built-in.o targets which finally

+# turn into vmlinux), we will call a sub make in that other dir, and

+# after that we are sure that everything which is in that other dir

+# is now up to date.

+#

+# The only cases where we need to modify files which have global

+# effects are thus separated out and done before the recursive

+# descending is started. They are now explicitly listed as the

+# prepare rule.

+

+# To put more focus on warnings, be less verbose as default

+# Use 'make V=1' to see the full commands

+

+ifdef V

+ ifeq ("$(origin V)", "command line")

+ KBUILD\_VERBOSE = $(V)

+ endif

+endif

+ifndef KBUILD\_VERBOSE

+ KBUILD\_VERBOSE = 0

+endif

+

+# Call a source code checker (by default, "sparse") as part of the

+# C compilation.

+#

+# Use 'make C=1' to enable checking of only re-compiled files.

+# Use 'make C=2' to enable checking of \*all\* source files, regardless

+# of whether they are re-compiled or not.

+#

+# See the file "Documentation/sparse.txt" for more details, including

+# where to get the "sparse" utility.

+

+ifdef C

+ ifeq ("$(origin C)", "command line")

+ KBUILD\_CHECKSRC = $(C)

+ endif

+endif

+ifndef KBUILD\_CHECKSRC

+ KBUILD\_CHECKSRC = 0

+endif

+

+# Use make M=dir to specify directory of external module to build

+# Old syntax make ... SUBDIRS=$PWD is still supported

+# Setting the environment variable KBUILD\_EXTMOD take precedence

+ifdef SUBDIRS

+ KBUILD\_EXTMOD ?= $(SUBDIRS)

+endif

+ifdef M

+ ifeq ("$(origin M)", "command line")

+ KBUILD\_EXTMOD := $(M)

+ endif

+endif

+

+

+# kbuild supports saving output files in a separate directory.

+# To locate output files in a separate directory two syntaxes are supported.

+# In both cases the working directory must be the root of the kernel src.

+# 1) O=

+# Use "make O=dir/to/store/output/files/"

+#

+# 2) Set KBUILD\_OUTPUT

+# Set the environment variable KBUILD\_OUTPUT to point to the directory

+# where the output files shall be placed.

+# export KBUILD\_OUTPUT=dir/to/store/output/files/

+# make

+#

+# The O= assignment takes precedence over the KBUILD\_OUTPUT environment

+# variable.

+

+

+# KBUILD\_SRC is set on invocation of make in OBJ directory

+# KBUILD\_SRC is not intended to be used by the regular user (for now)

+ifeq ($(KBUILD\_SRC),)

+

+# OK, Make called in directory where kernel src resides

+# Do we want to locate output files in a separate directory?

+ifdef O

+ ifeq ("$(origin O)", "command line")

+ KBUILD\_OUTPUT := $(O)

+ endif

+endif

+

+# That's our default target when none is given on the command line

+PHONY := \_all

+\_all:

+

+# Cancel implicit rules on top Makefile

+$(CURDIR)/Makefile Makefile: ;

+

+ifneq ($(KBUILD\_OUTPUT),)

+# Invoke a second make in the output directory, passing relevant variables

+# check that the output directory actually exists

+saved-output := $(KBUILD\_OUTPUT)

+KBUILD\_OUTPUT := $(shell cd $(KBUILD\_OUTPUT) && /bin/pwd)

+$(if $(KBUILD\_OUTPUT),, \

+ $(error output directory "$(saved-output)" does not exist))

+

+PHONY += $(MAKECMDGOALS) sub-make

+

+$(filter-out \_all sub-make $(CURDIR)/Makefile, $(MAKECMDGOALS)) \_all: sub-make

+ $(Q)@:

+

+sub-make: FORCE

+ $(if $(KBUILD\_VERBOSE:1=),@)$(MAKE) -C $(KBUILD\_OUTPUT) \

+ KBUILD\_SRC=$(CURDIR) \

+ KBUILD\_EXTMOD="$(KBUILD\_EXTMOD)" -f $(CURDIR)/Makefile \

+ $(filter-out \_all sub-make,$(MAKECMDGOALS))

+

+# Leave processing to above invocation of make

+skip-makefile := 1

+endif # ifneq ($(KBUILD\_OUTPUT),)

+endif # ifeq ($(KBUILD\_SRC),)

+

+# We process the rest of the Makefile if this is the final invocation of make

+ifeq ($(skip-makefile),)

+

+# If building an external module we do not care about the all: rule

+# but instead \_all depend on modules

+PHONY += all

+ifeq ($(KBUILD\_EXTMOD),)

+\_all: all

+else

+\_all: modules

+endif

+

+srctree := $(if $(KBUILD\_SRC),$(KBUILD\_SRC),$(CURDIR))

+TOPDIR := $(srctree)

+# FIXME - TOPDIR is obsolete, use srctree/objtree

+objtree := $(CURDIR)

+src := $(srctree)

+obj := $(objtree)

+

+VPATH := $(srctree)$(if $(KBUILD\_EXTMOD),:$(KBUILD\_EXTMOD))

+

+export srctree objtree VPATH TOPDIR

+

+

+# SUBARCH tells the usermode build what the underlying arch is. That is set

+# first, and if a usermode build is happening, the "ARCH=um" on the command

+# line overrides the setting of ARCH below. If a native build is happening,

+# then ARCH is assigned, getting whatever value it gets normally, and

+# SUBARCH is subsequently ignored.

+

+SUBARCH := $(shell uname -m | sed -e s/i.86/i386/ -e s/sun4u/sparc64/ \

+ -e s/arm.\*/arm/ -e s/sa110/arm/ \

+ -e s/s390x/s390/ -e s/parisc64/parisc/ \

+ -e s/ppc.\*/powerpc/ -e s/mips.\*/mips/ \

+ -e s/sh[234].\*/sh/ )

+

+# Cross compiling and selecting different set of gcc/bin-utils

+# ---------------------------------------------------------------------------

+#

+# When performing cross compilation for other architectures ARCH shall be set

+# to the target architecture. (See arch/\* for the possibilities).

+# ARCH can be set during invocation of make:

+# make ARCH=ia64

+# Another way is to have ARCH set in the environment.

+# The default ARCH is the host where make is executed.

+

+# CROSS\_COMPILE specify the prefix used for all executables used

+# during compilation. Only gcc and related bin-utils executables

+# are prefixed with $(CROSS\_COMPILE).

+# CROSS\_COMPILE can be set on the command line

+# make CROSS\_COMPILE=ia64-linux-

+# Alternatively CROSS\_COMPILE can be set in the environment.

+# Default value for CROSS\_COMPILE is not to prefix executables

+# Note: Some architectures assign CROSS\_COMPILE in their arch/\*/Makefile

+

+ARCH ?= $(SUBARCH)

+CROSS\_COMPILE ?=

+

+# Architecture as present in compile.h

+UTS\_MACHINE := $(ARCH)

+SRCARCH := $(ARCH)

+

+# Additional ARCH settings for x86

+ifeq ($(ARCH),i386)

+ SRCARCH := x86

+endif

+ifeq ($(ARCH),x86\_64)

+ SRCARCH := x86

+endif

+

+KCONFIG\_CONFIG ?= .config

+

+# SHELL used by kbuild

+CONFIG\_SHELL := $(shell if [ -x "$$BASH" ]; then echo $$BASH; \

+ else if [ -x /bin/bash ]; then echo /bin/bash; \

+ else echo sh; fi ; fi)

+

+HOSTCC = gcc

+HOSTCXX = g++

+HOSTCFLAGS = -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer

+HOSTCXXFLAGS = -O2

+

+# Decide whether to build built-in, modular, or both.

+# Normally, just do built-in.

+

+KBUILD\_MODULES :=

+KBUILD\_BUILTIN := 1

+

+# If we have only "make modules", don't compile built-in objects.

+# When we're building modules with modversions, we need to consider

+# the built-in objects during the descend as well, in order to

+# make sure the checksums are up to date before we record them.

+

+ifeq ($(MAKECMDGOALS),modules)

+ KBUILD\_BUILTIN := $(if $(CONFIG\_MODVERSIONS),1)

+endif

+

+# If we have "make <whatever> modules", compile modules

+# in addition to whatever we do anyway.

+# Just "make" or "make all" shall build modules as well

+

+ifneq ($(filter all \_all modules,$(MAKECMDGOALS)),)

+ KBUILD\_MODULES := 1

+endif

+

+ifeq ($(MAKECMDGOALS),)

+ KBUILD\_MODULES := 1

+endif

+

+export KBUILD\_MODULES KBUILD\_BUILTIN

+export KBUILD\_CHECKSRC KBUILD\_SRC KBUILD\_EXTMOD

+

+# Beautify output

+# ---------------------------------------------------------------------------

+#

+# Normally, we echo the whole command before executing it. By making

+# that echo $($(quiet)$(cmd)), we now have the possibility to set

+# $(quiet) to choose other forms of output instead, e.g.

+#

+# quiet\_cmd\_cc\_o\_c = Compiling $(RELDIR)/$@

+# cmd\_cc\_o\_c = $(CC) $(c\_flags) -c -o $@ $<

+#

+# If $(quiet) is empty, the whole command will be printed.

+# If it is set to "quiet\_", only the short version will be printed.

+# If it is set to "silent\_", nothing will be printed at all, since

+# the variable $(silent\_cmd\_cc\_o\_c) doesn't exist.

+#

+# A simple variant is to prefix commands with $(Q) - that's useful

+# for commands that shall be hidden in non-verbose mode.

+#

+# $(Q)ln $@ :<

+#

+# If KBUILD\_VERBOSE equals 0 then the above command will be hidden.

+# If KBUILD\_VERBOSE equals 1 then the above command is displayed.

+

+ifeq ($(KBUILD\_VERBOSE),1)

+ quiet =

+ Q =

+else

+ quiet=quiet\_

+ Q = @

+endif

+

+# If the user is running make -s (silent mode), suppress echoing of

+# commands

+

+ifneq ($(findstring s,$(MAKEFLAGS)),)

+ quiet=silent\_

+endif

+

+export quiet Q KBUILD\_VERBOSE

+

+

+# Look for make include files relative to root of kernel src

+MAKEFLAGS += --include-dir=$(srctree)

+

+# We need some generic definitions (do not try to remake the file).

+$(srctree)/scripts/Kbuild.include: ;

+include $(srctree)/scripts/Kbuild.include

+

+# Make variables (CC, etc...)

+

+AS = $(CROSS\_COMPILE)as

+LD = $(CROSS\_COMPILE)ld

+CC = $(CROSS\_COMPILE)gcc

+CPP = $(CC) -E

+AR = $(CROSS\_COMPILE)ar

+NM = $(CROSS\_COMPILE)nm

+STRIP = $(CROSS\_COMPILE)strip

+OBJCOPY = $(CROSS\_COMPILE)objcopy

+OBJDUMP = $(CROSS\_COMPILE)objdump

+AWK = awk

+GENKSYMS = scripts/genksyms/genksyms

+DEPMOD = /sbin/depmod

+KALLSYMS = scripts/kallsyms

+PERL = perl

+CHECK = sparse

+

+CHECKFLAGS := -D\_\_linux\_\_ -Dlinux -D\_\_STDC\_\_ -Dunix -D\_\_unix\_\_ -Wbitwise $(CF)

+MODFLAGS = -DMODULE

+CFLAGS\_MODULE = $(MODFLAGS)

+AFLAGS\_MODULE = $(MODFLAGS)

+LDFLAGS\_MODULE =

+CFLAGS\_KERNEL =

+AFLAGS\_KERNEL =

+

+

+# Use LINUXINCLUDE when you must reference the include/ directory.

+# Needed to be compatible with the O= option

+LINUXINCLUDE := -Iinclude \

+ $(if $(KBUILD\_SRC),-Iinclude2 -I$(srctree)/include) \

+ -include include/linux/autoconf.h

+

+KBUILD\_CPPFLAGS := -D\_\_KERNEL\_\_ $(LINUXINCLUDE)

+

+KBUILD\_CFLAGS := -Wall -Wundef -Wstrict-prototypes -Wno-trigraphs \

+ -fno-strict-aliasing -fno-common \

+ -Werror-implicit-function-declaration

+KBUILD\_AFLAGS := -D\_\_ASSEMBLY\_\_

+

+# Read KERNELRELEASE from include/config/kernel.release (if it exists)

+KERNELRELEASE = $(shell cat include/config/kernel.release 2> /dev/null)

+KERNELVERSION = $(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)

+

+export VERSION PATCHLEVEL SUBLEVEL KERNELRELEASE KERNELVERSION

+export ARCH SRCARCH CONFIG\_SHELL HOSTCC HOSTCFLAGS CROSS\_COMPILE AS LD CC

+export CPP AR NM STRIP OBJCOPY OBJDUMP MAKE AWK GENKSYMS PERL UTS\_MACHINE

+export HOSTCXX HOSTCXXFLAGS LDFLAGS\_MODULE CHECK CHECKFLAGS

+

+export KBUILD\_CPPFLAGS NOSTDINC\_FLAGS LINUXINCLUDE OBJCOPYFLAGS LDFLAGS

+export KBUILD\_CFLAGS CFLAGS\_KERNEL CFLAGS\_MODULE

+export KBUILD\_AFLAGS AFLAGS\_KERNEL AFLAGS\_MODULE

+

+# When compiling out-of-tree modules, put MODVERDIR in the module

+# tree rather than in the kernel tree. The kernel tree might

+# even be read-only.

+export MODVERDIR := $(if $(KBUILD\_EXTMOD),$(firstword $(KBUILD\_EXTMOD))/).tmp\_versions

+

+# Files to ignore in find ... statements

+

+RCS\_FIND\_IGNORE := \( -name SCCS -o -name BitKeeper -o -name .svn -o -name CVS -o -name .pc -o -name .hg -o -name .git \) -prune -o

+export RCS\_TAR\_IGNORE := --exclude SCCS --exclude BitKeeper --exclude .svn --exclude CVS --exclude .pc --exclude .hg --exclude .git

+

+# ===========================================================================

+# Rules shared between \*config targets and build targets

+

+# Basic helpers built in scripts/

+PHONY += scripts\_basic

+scripts\_basic:

+ $(Q)$(MAKE) $(build)=scripts/basic

+

+# To avoid any implicit rule to kick in, define an empty command.

+scripts/basic/%: scripts\_basic ;

+

+PHONY += outputmakefile

+# outputmakefile generates a Makefile in the output directory, if using a

+# separate output directory. This allows convenient use of make in the

+# output directory.

+outputmakefile:

+ifneq ($(KBUILD\_SRC),)

+ $(Q)$(CONFIG\_SHELL) $(srctree)/scripts/mkmakefile \

+ $(srctree) $(objtree) $(VERSION) $(PATCHLEVEL)

+endif

+

+# To make sure we do not include .config for any of the \*config targets

+# catch them early, and hand them over to scripts/kconfig/Makefile

+# It is allowed to specify more targets when calling make, including

+# mixing \*config targets and build targets.

+# For example 'make oldconfig all'.

+# Detect when mixed targets is specified, and make a second invocation

+# of make so .config is not included in this case either (for \*config).

+

+no-dot-config-targets := clean mrproper distclean \

+ cscope TAGS tags help %docs check% \

+ include/linux/version.h headers\_% \

+ kernelrelease kernelversion

+

+config-targets := 0

+mixed-targets := 0

+dot-config := 1

+

+ifneq ($(filter $(no-dot-config-targets), $(MAKECMDGOALS)),)

+ ifeq ($(filter-out $(no-dot-config-targets), $(MAKECMDGOALS)),)

+ dot-config := 0

+ endif

+endif

+

+ifeq ($(KBUILD\_EXTMOD),)

+ ifneq ($(filter config %config,$(MAKECMDGOALS)),)

+ config-targets := 1

+ ifneq ($(filter-out config %config,$(MAKECMDGOALS)),)

+ mixed-targets := 1

+ endif

+ endif

+endif

+

+ifeq ($(mixed-targets),1)

+# ===========================================================================

+# We're called with mixed targets (\*config and build targets).

+# Handle them one by one.

+

+%:: FORCE

+ $(Q)$(MAKE) -C $(srctree) KBUILD\_SRC= $@

+

+else

+ifeq ($(config-targets),1)

+# ===========================================================================

+# \*config targets only - make sure prerequisites are updated, and descend

+# in scripts/kconfig to make the \*config target

+

+# Read arch specific Makefile to set KBUILD\_DEFCONFIG as needed.

+# KBUILD\_DEFCONFIG may point out an alternative default configuration

+# used for 'make defconfig'

+include $(srctree)/arch/$(SRCARCH)/Makefile

+export KBUILD\_DEFCONFIG

+

+config %config: scripts\_basic outputmakefile FORCE

+ $(Q)mkdir -p include/linux include/config

+ $(Q)$(MAKE) $(build)=scripts/kconfig $@

+

+else

+# ===========================================================================

+# Build targets only - this includes vmlinux, arch specific targets, clean

+# targets and others. In general all targets except \*config targets.

+

+ifeq ($(KBUILD\_EXTMOD),)

+# Additional helpers built in scripts/

+# Carefully list dependencies so we do not try to build scripts twice

+# in parallel

+PHONY += scripts

+scripts: scripts\_basic include/config/auto.conf

+ $(Q)$(MAKE) $(build)=$(@)

+

+# Objects we will link into vmlinux / subdirs we need to visit

+init-y := init/

+drivers-y := drivers/ sound/

+net-y := net/

+libs-y := lib/

+core-y := usr/

+endif # KBUILD\_EXTMOD

+

+ifeq ($(dot-config),1)

+# Read in config

+-include include/config/auto.conf

+

+ifeq ($(KBUILD\_EXTMOD),)

+# Read in dependencies to all Kconfig\* files, make sure to run

+# oldconfig if changes are detected.

+-include include/config/auto.conf.cmd

+

+# To avoid any implicit rule to kick in, define an empty command

+$(KCONFIG\_CONFIG) include/config/auto.conf.cmd: ;

+

+# If .config is newer than include/config/auto.conf, someone tinkered

+# with it and forgot to run make oldconfig.

+# if auto.conf.cmd is missing then we are probably in a cleaned tree so

+# we execute the config step to be sure to catch updated Kconfig files

+include/config/auto.conf: $(KCONFIG\_CONFIG) include/config/auto.conf.cmd

+ $(Q)$(MAKE) -f $(srctree)/Makefile silentoldconfig

+else

+# external modules needs include/linux/autoconf.h and include/config/auto.conf

+# but do not care if they are up-to-date. Use auto.conf to trigger the test

+PHONY += include/config/auto.conf

+

+include/config/auto.conf:

+ $(Q)test -e include/linux/autoconf.h -a -e $@ || ( \

+ echo; \

+ echo " ERROR: Kernel configuration is invalid."; \

+ echo " include/linux/autoconf.h or $@ are missing."; \

+ echo " Run 'make oldconfig && make prepare' on kernel src to fix it."; \

+ echo; \

+ /bin/false)

+

+endif # KBUILD\_EXTMOD

+

+else

+# Dummy target needed, because used as prerequisite

+include/config/auto.conf: ;

+endif # $(dot-config)

+

+# The all: target is the default when no target is given on the

+# command line.

+# This allow a user to issue only 'make' to build a kernel including modules

+# Defaults vmlinux but it is usually overridden in the arch makefile

+all: vmlinux

+

+ifdef CONFIG\_CC\_OPTIMIZE\_FOR\_SIZE

+KBUILD\_CFLAGS += -Os

+else

+KBUILD\_CFLAGS += -O2

+endif

+

+include $(srctree)/arch/$(SRCARCH)/Makefile

+

+ifdef CONFIG\_FRAME\_POINTER

+KBUILD\_CFLAGS += -fno-omit-frame-pointer -fno-optimize-sibling-calls

+else

+KBUILD\_CFLAGS += -fomit-frame-pointer

+endif

+

+ifdef CONFIG\_DEBUG\_INFO

+KBUILD\_CFLAGS += -g

+KBUILD\_AFLAGS += -gdwarf-2

+endif

+

+# Force gcc to behave correct even for buggy distributions

+KBUILD\_CFLAGS += $(call cc-option, -fno-stack-protector)

+

+# arch Makefile may override CC so keep this after arch Makefile is included

+NOSTDINC\_FLAGS += -nostdinc -isystem $(shell $(CC) -print-file-name=include)

+CHECKFLAGS += $(NOSTDINC\_FLAGS)

+

+# warn about C99 declaration after statement

+KBUILD\_CFLAGS += $(call cc-option,-Wdeclaration-after-statement,)

+

+# disable pointer signed / unsigned warnings in gcc 4.0

+KBUILD\_CFLAGS += $(call cc-option,-Wno-pointer-sign,)

+

+# Add user supplied CPPFLAGS, AFLAGS and CFLAGS as the last assignments

+# But warn user when we do so

+warn-assign = \

+$(warning "WARNING: Appending $$K$(1) ($(K$(1))) from $(origin K$(1)) to kernel $$$(1)")

+

+ifneq ($(KCPPFLAGS),)

+ $(call warn-assign,CPPFLAGS)

+ KBUILD\_CPPFLAGS += $(KCPPFLAGS)

+endif

+ifneq ($(KAFLAGS),)

+ $(call warn-assign,AFLAGS)

+ KBUILD\_AFLAGS += $(KAFLAGS)

+endif

+ifneq ($(KCFLAGS),)

+ $(call warn-assign,CFLAGS)

+ KBUILD\_CFLAGS += $(KCFLAGS)

+endif

+

+# Use --build-id when available.

+LDFLAGS\_BUILD\_ID = $(patsubst -Wl$(comma)%,%,\

+ $(call ld-option, -Wl$(comma)--build-id,))

+LDFLAGS\_MODULE += $(LDFLAGS\_BUILD\_ID)

+LDFLAGS\_vmlinux += $(LDFLAGS\_BUILD\_ID)

+

+# Default kernel image to build when no specific target is given.

+# KBUILD\_IMAGE may be overruled on the command line or

+# set in the environment

+# Also any assignments in arch/$(ARCH)/Makefile take precedence over

+# this default value

+export KBUILD\_IMAGE ?= vmlinux

+

+#

+# INSTALL\_PATH specifies where to place the updated kernel and system map

+# images. Default is /boot, but you can set it to other values

+export INSTALL\_PATH ?= /boot

+

+#

+# INSTALL\_MOD\_PATH specifies a prefix to MODLIB for module directory

+# relocations required by build roots. This is not defined in the

+# makefile but the argument can be passed to make if needed.

+#

+

+MODLIB = $(INSTALL\_MOD\_PATH)/lib/modules/$(KERNELRELEASE)

+export MODLIB

+

+#

+# INSTALL\_MOD\_STRIP, if defined, will cause modules to be

+# stripped after they are installed. If INSTALL\_MOD\_STRIP is '1', then

+# the default option --strip-debug will be used. Otherwise,

+# INSTALL\_MOD\_STRIP will used as the options to the strip command.

+

+ifdef INSTALL\_MOD\_STRIP

+ifeq ($(INSTALL\_MOD\_STRIP),1)

+mod\_strip\_cmd = $(STRIP) --strip-debug

+else

+mod\_strip\_cmd = $(STRIP) $(INSTALL\_MOD\_STRIP)

+endif # INSTALL\_MOD\_STRIP=1

+else

+mod\_strip\_cmd = true

+endif # INSTALL\_MOD\_STRIP

+export mod\_strip\_cmd

+

+

+ifeq ($(KBUILD\_EXTMOD),)

+core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/

+

+vmlinux-dirs := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \

+ $(core-y) $(core-m) $(drivers-y) $(drivers-m) \

+ $(net-y) $(net-m) $(libs-y) $(libs-m)))

+

+vmlinux-alldirs := $(sort $(vmlinux-dirs) $(patsubst %/,%,$(filter %/, \

+ $(init-n) $(init-) \

+ $(core-n) $(core-) $(drivers-n) $(drivers-) \

+ $(net-n) $(net-) $(libs-n) $(libs-))))

+

+init-y := $(patsubst %/, %/built-in.o, $(init-y))

+core-y := $(patsubst %/, %/built-in.o, $(core-y))

+drivers-y := $(patsubst %/, %/built-in.o, $(drivers-y))

+net-y := $(patsubst %/, %/built-in.o, $(net-y))

+libs-y1 := $(patsubst %/, %/lib.a, $(libs-y))

+libs-y2 := $(patsubst %/, %/built-in.o, $(libs-y))

+libs-y := $(libs-y1) $(libs-y2)

+

+# Build vmlinux

+# ---------------------------------------------------------------------------

+# vmlinux is built from the objects selected by $(vmlinux-init) and

+# $(vmlinux-main). Most are built-in.o files from top-level directories

+# in the kernel tree, others are specified in arch/$(ARCH)/Makefile.

+# Ordering when linking is important, and $(vmlinux-init) must be first.

+#

+# vmlinux

+# ^

+# |

+# +-< $(vmlinux-init)

+# | +--< init/version.o + more

+# |

+# +--< $(vmlinux-main)

+# | +--< driver/built-in.o mm/built-in.o + more

+# |

+# +-< kallsyms.o (see description in CONFIG\_KALLSYMS section)

+#

+# vmlinux version (uname -v) cannot be updated during normal

+# descending-into-subdirs phase since we do not yet know if we need to

+# update vmlinux.

+# Therefore this step is delayed until just before final link of vmlinux -

+# except in the kallsyms case where it is done just before adding the

+# symbols to the kernel.

+#

+# System.map is generated to document addresses of all kernel symbols

+

+vmlinux-init := $(head-y) $(init-y)

+vmlinux-main := $(core-y) $(libs-y) $(drivers-y) $(net-y)

+vmlinux-all := $(vmlinux-init) $(vmlinux-main)

+vmlinux-lds := arch/$(SRCARCH)/kernel/vmlinux.lds

+export KBUILD\_VMLINUX\_OBJS := $(vmlinux-all)

+

+# Rule to link vmlinux - also used during CONFIG\_KALLSYMS

+# May be overridden by arch/$(ARCH)/Makefile

+quiet\_cmd\_vmlinux\_\_ ?= LD $@

+ cmd\_vmlinux\_\_ ?= $(LD) $(LDFLAGS) $(LDFLAGS\_vmlinux) -o $@ \

+ -T $(vmlinux-lds) $(vmlinux-init) \

+ --start-group $(vmlinux-main) --end-group \

+ $(filter-out $(vmlinux-lds) $(vmlinux-init) $(vmlinux-main) vmlinux.o FORCE ,$^)

+

+# Generate new vmlinux version

+quiet\_cmd\_vmlinux\_version = GEN .version

+ cmd\_vmlinux\_version = set -e; \

+ if [ ! -r .version ]; then \

+ rm -f .version; \

+ echo 1 >.version; \

+ else \

+ mv .version .old\_version; \

+ expr 0$$(cat .old\_version) + 1 >.version; \

+ fi; \

+ $(MAKE) $(build)=init

+

+# Generate System.map

+quiet\_cmd\_sysmap = SYSMAP

+ cmd\_sysmap = $(CONFIG\_SHELL) $(srctree)/scripts/mksysmap

+

+# Link of vmlinux

+# If CONFIG\_KALLSYMS is set .version is already updated

+# Generate System.map and verify that the content is consistent

+# Use + in front of the vmlinux\_version rule to silent warning with make -j2

+# First command is ':' to allow us to use + in front of the rule

+define rule\_vmlinux\_\_

+ :

+ $(if $(CONFIG\_KALLSYMS),,+$(call cmd,vmlinux\_version))

+

+ $(call cmd,vmlinux\_\_)

+ $(Q)echo 'cmd\_$@ := $(cmd\_vmlinux\_\_)' > $(@D)/.$(@F).cmd

+

+ $(Q)$(if $($(quiet)cmd\_sysmap), \

+ echo ' $($(quiet)cmd\_sysmap) System.map' &&) \

+ $(cmd\_sysmap) $@ System.map; \

+ if [ $$? -ne 0 ]; then \

+ rm -f $@; \

+ /bin/false; \

+ fi;

+ $(verify\_kallsyms)

+endef

+

+

+ifdef CONFIG\_KALLSYMS

+# Generate section listing all symbols and add it into vmlinux $(kallsyms.o)

+# It's a three stage process:

+# o .tmp\_vmlinux1 has all symbols and sections, but \_\_kallsyms is

+# empty

+# Running kallsyms on that gives us .tmp\_kallsyms1.o with

+# the right size - vmlinux version (uname -v) is updated during this step

+# o .tmp\_vmlinux2 now has a \_\_kallsyms section of the right size,

+# but due to the added section, some addresses have shifted.

+# From here, we generate a correct .tmp\_kallsyms2.o

+# o The correct .tmp\_kallsyms2.o is linked into the final vmlinux.

+# o Verify that the System.map from vmlinux matches the map from

+# .tmp\_vmlinux2, just in case we did not generate kallsyms correctly.

+# o If CONFIG\_KALLSYMS\_EXTRA\_PASS is set, do an extra pass using

+# .tmp\_vmlinux3 and .tmp\_kallsyms3.o. This is only meant as a

+# temporary bypass to allow the kernel to be built while the

+# maintainers work out what went wrong with kallsyms.

+

+ifdef CONFIG\_KALLSYMS\_EXTRA\_PASS

+last\_kallsyms := 3

+else

+last\_kallsyms := 2

+endif

+

+kallsyms.o := .tmp\_kallsyms$(last\_kallsyms).o

+

+define verify\_kallsyms

+ $(Q)$(if $($(quiet)cmd\_sysmap), \

+ echo ' $($(quiet)cmd\_sysmap) .tmp\_System.map' &&) \

+ $(cmd\_sysmap) .tmp\_vmlinux$(last\_kallsyms) .tmp\_System.map

+ $(Q)cmp -s System.map .tmp\_System.map || \

+ (echo Inconsistent kallsyms data; \

+ echo Try setting CONFIG\_KALLSYMS\_EXTRA\_PASS; \

+ rm .tmp\_kallsyms\* ; /bin/false )

+endef

+

+# Update vmlinux version before link

+# Use + in front of this rule to silent warning about make -j1

+# First command is ':' to allow us to use + in front of this rule

+cmd\_ksym\_ld = $(cmd\_vmlinux\_\_)

+define rule\_ksym\_ld

+ :

+ +$(call cmd,vmlinux\_version)

+ $(call cmd,vmlinux\_\_)

+ $(Q)echo 'cmd\_$@ := $(cmd\_vmlinux\_\_)' > $(@D)/.$(@F).cmd

+endef

+

+# Generate .S file with all kernel symbols

+quiet\_cmd\_kallsyms = KSYM $@

+ cmd\_kallsyms = $(NM) -n $< | $(KALLSYMS) \

+ $(if $(CONFIG\_KALLSYMS\_ALL),--all-symbols) > $@

+

+.tmp\_kallsyms1.o .tmp\_kallsyms2.o .tmp\_kallsyms3.o: %.o: %.S scripts FORCE

+ $(call if\_changed\_dep,as\_o\_S)

+

+.tmp\_kallsyms%.S: .tmp\_vmlinux% $(KALLSYMS)

+ $(call cmd,kallsyms)

+

+# .tmp\_vmlinux1 must be complete except kallsyms, so update vmlinux version

+.tmp\_vmlinux1: $(vmlinux-lds) $(vmlinux-all) FORCE

+ $(call if\_changed\_rule,ksym\_ld)

+

+.tmp\_vmlinux2: $(vmlinux-lds) $(vmlinux-all) .tmp\_kallsyms1.o FORCE

+ $(call if\_changed,vmlinux\_\_)

+

+.tmp\_vmlinux3: $(vmlinux-lds) $(vmlinux-all) .tmp\_kallsyms2.o FORCE

+ $(call if\_changed,vmlinux\_\_)

+

+# Needs to visit scripts/ before $(KALLSYMS) can be used.

+$(KALLSYMS): scripts ;

+

+# Generate some data for debugging strange kallsyms problems

+debug\_kallsyms: .tmp\_map$(last\_kallsyms)

+

+.tmp\_map%: .tmp\_vmlinux% FORCE

+ ($(OBJDUMP) -h $< | $(AWK) '/^ +[0-9]/{print $$4 " 0 " $$2}'; $(NM) $<) | sort > $@

+

+.tmp\_map3: .tmp\_map2

+

+.tmp\_map2: .tmp\_map1

+

+endif # ifdef CONFIG\_KALLSYMS

+

+# Do modpost on a prelinked vmlinux. The finally linked vmlinux has

+# relevant sections renamed as per the linker script.

+quiet\_cmd\_vmlinux-modpost = LD $@

+ cmd\_vmlinux-modpost = $(LD) $(LDFLAGS) -r -o $@ \

+ $(vmlinux-init) --start-group $(vmlinux-main) --end-group \

+ $(filter-out $(vmlinux-init) $(vmlinux-main) $(vmlinux-lds) FORCE ,$^)

+define rule\_vmlinux-modpost

+ :

+ +$(call cmd,vmlinux-modpost)

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modpost $@

+ $(Q)echo 'cmd\_$@ := $(cmd\_vmlinux-modpost)' > $(dot-target).cmd

+endef

+

+# vmlinux image - including updated kernel symbols

+vmlinux: $(vmlinux-lds) $(vmlinux-init) $(vmlinux-main) $(kallsyms.o) vmlinux.o FORCE

+ifdef CONFIG\_HEADERS\_CHECK

+ $(Q)$(MAKE) -f $(srctree)/Makefile headers\_check

+endif

+ifdef CONFIG\_SAMPLES

+ $(Q)$(MAKE) $(build)=samples

+endif

+ $(call vmlinux-modpost)

+ $(call if\_changed\_rule,vmlinux\_\_)

+ $(Q)rm -f .old\_version

+

+vmlinux.o: $(vmlinux-lds) $(vmlinux-init) $(vmlinux-main) $(kallsyms.o) FORCE

+ $(call if\_changed\_rule,vmlinux-modpost)

+

+# The actual objects are generated when descending,

+# make sure no implicit rule kicks in

+$(sort $(vmlinux-init) $(vmlinux-main)) $(vmlinux-lds): $(vmlinux-dirs) ;

+

+# Handle descending into subdirectories listed in $(vmlinux-dirs)

+# Preset locale variables to speed up the build process. Limit locale

+# tweaks to this spot to avoid wrong language settings when running

+# make menuconfig etc.

+# Error messages still appears in the original language

+

+PHONY += $(vmlinux-dirs)

+$(vmlinux-dirs): prepare scripts

+ $(Q)$(MAKE) $(build)=$@

+

+# Build the kernel release string

+#

+# The KERNELRELEASE value built here is stored in the file

+# include/config/kernel.release, and is used when executing several

+# make targets, such as "make install" or "make modules\_install."

+#

+# The eventual kernel release string consists of the following fields,

+# shown in a hierarchical format to show how smaller parts are concatenated

+# to form the larger and final value, with values coming from places like

+# the Makefile, kernel config options, make command line options and/or

+# SCM tag information.

+#

+# $(KERNELVERSION)

+# $(VERSION) eg, 2

+# $(PATCHLEVEL) eg, 6

+# $(SUBLEVEL) eg, 18

+# $(EXTRAVERSION) eg, -rc6

+# $(localver-full)

+# $(localver)

+# localversion\* (files without backups, containing '~')

+# $(CONFIG\_LOCALVERSION) (from kernel config setting)

+# $(localver-auto) (only if CONFIG\_LOCALVERSION\_AUTO is set)

+# ./scripts/setlocalversion (SCM tag, if one exists)

+# $(LOCALVERSION) (from make command line if provided)

+#

+# Note how the final $(localver-auto) string is included \*only\* if the

+# kernel config option CONFIG\_LOCALVERSION\_AUTO is selected. Also, at the

+# moment, only git is supported but other SCMs can edit the script

+# scripts/setlocalversion and add the appropriate checks as needed.

+

+pattern = ".\*/localversion[^~]\*"

+string = $(shell cat /dev/null \

+ `find $(objtree) $(srctree) -maxdepth 1 -regex $(pattern) | sort -u`)

+

+localver = $(subst $(space),, $(string) \

+ $(patsubst "%",%,$(CONFIG\_LOCALVERSION)))

+

+# If CONFIG\_LOCALVERSION\_AUTO is set scripts/setlocalversion is called

+# and if the SCM is know a tag from the SCM is appended.

+# The appended tag is determined by the SCM used.

+#

+# Currently, only git is supported.

+# Other SCMs can edit scripts/setlocalversion and add the appropriate

+# checks as needed.

+ifdef CONFIG\_LOCALVERSION\_AUTO

+ \_localver-auto = $(shell $(CONFIG\_SHELL) \

+ $(srctree)/scripts/setlocalversion $(srctree))

+ localver-auto = $(LOCALVERSION)$(\_localver-auto)

+endif

+

+localver-full = $(localver)$(localver-auto)

+

+# Store (new) KERNELRELASE string in include/config/kernel.release

+kernelrelease = $(KERNELVERSION)$(localver-full)

+include/config/kernel.release: include/config/auto.conf FORCE

+ $(Q)rm -f $@

+ $(Q)echo $(kernelrelease) > $@

+

+

+# Things we need to do before we recursively start building the kernel

+# or the modules are listed in "prepare".

+# A multi level approach is used. prepareN is processed before prepareN-1.

+# archprepare is used in arch Makefiles and when processed asm symlink,

+# version.h and scripts\_basic is processed / created.

+

+# Listed in dependency order

+PHONY += prepare archprepare prepare0 prepare1 prepare2 prepare3

+

+# prepare3 is used to check if we are building in a separate output directory,

+# and if so do:

+# 1) Check that make has not been executed in the kernel src $(srctree)

+# 2) Create the include2 directory, used for the second asm symlink

+prepare3: include/config/kernel.release

+ifneq ($(KBUILD\_SRC),)

+ @echo ' Using $(srctree) as source for kernel'

+ $(Q)if [ -f $(srctree)/.config -o -d $(srctree)/include/config ]; then \

+ echo " $(srctree) is not clean, please run 'make mrproper'";\

+ echo " in the '$(srctree)' directory.";\

+ /bin/false; \

+ fi;

+ $(Q)if [ ! -d include2 ]; then mkdir -p include2; fi;

+ $(Q)ln -fsn $(srctree)/include/asm-$(SRCARCH) include2/asm

+endif

+

+# prepare2 creates a makefile if using a separate output directory

+prepare2: prepare3 outputmakefile

+

+prepare1: prepare2 include/linux/version.h include/linux/utsrelease.h \

+ include/asm include/config/auto.conf

+ $(cmd\_crmodverdir)

+

+archprepare: prepare1 scripts\_basic

+

+prepare0: archprepare FORCE

+ $(Q)$(MAKE) $(build)=.

+ $(Q)$(MAKE) $(build)=. missing-syscalls

+

+# All the preparing..

+prepare: prepare0

+

+# Leave this as default for preprocessing vmlinux.lds.S, which is now

+# done in arch/$(ARCH)/kernel/Makefile

+

+export CPPFLAGS\_vmlinux.lds += -P -C -U$(ARCH)

+

+# The asm symlink changes when $(ARCH) changes.

+# Detect this and ask user to run make mrproper

+

+include/asm: FORCE

+ $(Q)set -e; asmlink=`readlink include/asm | cut -d '-' -f 2`; \

+ if [ -L include/asm ]; then \

+ if [ "$$asmlink" != "$(SRCARCH)" ]; then \

+ echo "ERROR: the symlink $@ points to asm-$$asmlink but asm-$(SRCARCH) was expected"; \

+ echo " set ARCH or save .config and run 'make mrproper' to fix it"; \

+ exit 1; \

+ fi; \

+ else \

+ echo ' SYMLINK $@ -> include/asm-$(SRCARCH)'; \

+ if [ ! -d include ]; then \

+ mkdir -p include; \

+ fi; \

+ ln -fsn asm-$(SRCARCH) $@; \

+ fi

+

+# Generate some files

+# ---------------------------------------------------------------------------

+

+# KERNELRELEASE can change from a few different places, meaning version.h

+# needs to be updated, so this check is forced on all builds

+

+uts\_len := 64

+define filechk\_utsrelease.h

+ if [ `echo -n "$(KERNELRELEASE)" | wc -c ` -gt $(uts\_len) ]; then \

+ echo '"$(KERNELRELEASE)" exceeds $(uts\_len) characters' >&2; \

+ exit 1; \

+ fi; \

+ (echo \#define UTS\_RELEASE \"$(KERNELRELEASE)\";)

+endef

+

+define filechk\_version.h

+ (echo \#define LINUX\_VERSION\_CODE $(shell \

+ expr $(VERSION) \\* 65536 + $(PATCHLEVEL) \\* 256 + $(SUBLEVEL)); \

+ echo '#define KERNEL\_VERSION(a,b,c) (((a) << 16) + ((b) << 8) + (c))';)

+endef

+

+include/linux/version.h: $(srctree)/Makefile FORCE

+ $(call filechk,version.h)

+

+include/linux/utsrelease.h: include/config/kernel.release FORCE

+ $(call filechk,utsrelease.h)

+

+# ---------------------------------------------------------------------------

+

+PHONY += depend dep

+depend dep:

+ @echo '\*\*\* Warning: make $@ is unnecessary now.'

+

+# ---------------------------------------------------------------------------

+# Kernel headers

+INSTALL\_HDR\_PATH=$(objtree)/usr

+export INSTALL\_HDR\_PATH

+

+HDRFILTER=generic i386 x86\_64

+HDRARCHES=$(filter-out $(HDRFILTER),$(patsubst $(srctree)/include/asm-%/Kbuild,%,$(wildcard $(srctree)/include/asm-\*/Kbuild)))

+

+PHONY += headers\_install\_all

+headers\_install\_all: include/linux/version.h scripts\_basic FORCE

+ $(Q)$(MAKE) $(build)=scripts scripts/unifdef

+ $(Q)for arch in $(HDRARCHES); do \

+ $(MAKE) ARCH=$$arch -f $(srctree)/scripts/Makefile.headersinst obj=include BIASMDIR=-bi-$$arch ;\

+ done

+

+PHONY += headers\_install

+headers\_install: include/linux/version.h scripts\_basic FORCE

+ @if [ ! -r $(srctree)/include/asm-$(SRCARCH)/Kbuild ]; then \

+ echo '\*\*\* Error: Headers not exportable for this architecture ($(SRCARCH))'; \

+ exit 1 ; fi

+ $(Q)$(MAKE) $(build)=scripts scripts/unifdef

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.headersinst ARCH=$(SRCARCH) obj=include

+

+PHONY += headers\_check\_all

+headers\_check\_all: headers\_install\_all

+ $(Q)for arch in $(HDRARCHES); do \

+ $(MAKE) ARCH=$$arch -f $(srctree)/scripts/Makefile.headersinst obj=include BIASMDIR=-bi-$$arch HDRCHECK=1 ;\

+ done

+

+PHONY += headers\_check

+headers\_check: headers\_install

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.headersinst ARCH=$(SRCARCH) obj=include HDRCHECK=1

+

+# ---------------------------------------------------------------------------

+# Modules

+

+ifdef CONFIG\_MODULES

+

+# By default, build modules as well

+

+all: modules

+

+# Build modules

+

+PHONY += modules

+modules: $(vmlinux-dirs) $(if $(KBUILD\_BUILTIN),vmlinux)

+ @echo ' Building modules, stage 2.';

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modpost

+

+

+# Target to prepare building external modules

+PHONY += modules\_prepare

+modules\_prepare: prepare scripts

+

+# Target to install modules

+PHONY += modules\_install

+modules\_install: \_modinst\_ \_modinst\_post

+

+PHONY += \_modinst\_

+\_modinst\_:

+ @if [ -z "`$(DEPMOD) -V 2>/dev/null | grep module-init-tools`" ]; then \

+ echo "Warning: you may need to install module-init-tools"; \

+ echo "See http://www.codemonkey.org.uk/docs/post-halloween-2.6.txt";\

+ sleep 1; \

+ fi

+ @rm -rf $(MODLIB)/kernel

+ @rm -f $(MODLIB)/source

+ @mkdir -p $(MODLIB)/kernel

+ @ln -s $(srctree) $(MODLIB)/source

+ @if [ ! $(objtree) -ef $(MODLIB)/build ]; then \

+ rm -f $(MODLIB)/build ; \

+ ln -s $(objtree) $(MODLIB)/build ; \

+ fi

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modinst

+

+# This depmod is only for convenience to give the initial

+# boot a modules.dep even before / is mounted read-write. However the

+# boot script depmod is the master version.

+PHONY += \_modinst\_post

+\_modinst\_post: \_modinst\_

+ $(call cmd,depmod)

+

+else # CONFIG\_MODULES

+

+# Modules not configured

+# ---------------------------------------------------------------------------

+

+modules modules\_install: FORCE

+ @echo

+ @echo "The present kernel configuration has modules disabled."

+ @echo "Type 'make config' and enable loadable module support."

+ @echo "Then build a kernel with module support enabled."

+ @echo

+ @exit 1

+

+endif # CONFIG\_MODULES

+

+###

+# Cleaning is done on three levels.

+# make clean Delete most generated files

+# Leave enough to build external modules

+# make mrproper Delete the current configuration, and all generated files

+# make distclean Remove editor backup files, patch leftover files and the like

+

+# Directories & files removed with 'make clean'

+CLEAN\_DIRS += $(MODVERDIR)

+CLEAN\_FILES += vmlinux System.map \

+ .tmp\_kallsyms\* .tmp\_version .tmp\_vmlinux\* .tmp\_System.map

+

+# Directories & files removed with 'make mrproper'

+MRPROPER\_DIRS += include/config include2 usr/include

+MRPROPER\_FILES += .config .config.old include/asm .version .old\_version \

+ include/linux/autoconf.h include/linux/version.h \

+ include/linux/utsrelease.h \

+ Module.symvers tags TAGS cscope\*

+

+# clean - Delete most, but leave enough to build external modules

+#

+clean: rm-dirs := $(CLEAN\_DIRS)

+clean: rm-files := $(CLEAN\_FILES)

+clean-dirs := $(addprefix \_clean\_,$(srctree) $(vmlinux-alldirs))

+

+PHONY += $(clean-dirs) clean archclean

+$(clean-dirs):

+ $(Q)$(MAKE) $(clean)=$(patsubst \_clean\_%,%,$@)

+

+clean: archclean $(clean-dirs)

+ $(call cmd,rmdirs)

+ $(call cmd,rmfiles)

+ @find . $(RCS\_FIND\_IGNORE) \

+ \( -name '\*.[oas]' -o -name '\*.ko' -o -name '.\*.cmd' \

+ -o -name '.\*.d' -o -name '.\*.tmp' -o -name '\*.mod.c' \

+ -o -name '\*.symtypes' \) \

+ -type f -print | xargs rm -f

+

+# mrproper - Delete all generated files, including .config

+#

+mrproper: rm-dirs := $(wildcard $(MRPROPER\_DIRS))

+mrproper: rm-files := $(wildcard $(MRPROPER\_FILES))

+mrproper-dirs := $(addprefix \_mrproper\_,Documentation/DocBook scripts)

+

+PHONY += $(mrproper-dirs) mrproper archmrproper

+$(mrproper-dirs):

+ $(Q)$(MAKE) $(clean)=$(patsubst \_mrproper\_%,%,$@)

+

+mrproper: clean archmrproper $(mrproper-dirs)

+ $(call cmd,rmdirs)

+ $(call cmd,rmfiles)

+

+# distclean

+#

+PHONY += distclean

+

+distclean: mrproper

+ @find $(srctree) $(RCS\_FIND\_IGNORE) \

+ \( -name '\*.orig' -o -name '\*.rej' -o -name '\*~' \

+ -o -name '\*.bak' -o -name '#\*#' -o -name '.\*.orig' \

+ -o -name '.\*.rej' -o -size 0 \

+ -o -name '\*%' -o -name '.\*.cmd' -o -name 'core' \) \

+ -type f -print | xargs rm -f

+

+

+# Packaging of the kernel to various formats

+# ---------------------------------------------------------------------------

+# rpm target kept for backward compatibility

+package-dir := $(srctree)/scripts/package

+

+%pkg: include/config/kernel.release FORCE

+ $(Q)$(MAKE) $(build)=$(package-dir) $@

+rpm: include/config/kernel.release FORCE

+ $(Q)$(MAKE) $(build)=$(package-dir) $@

+

+

+# Brief documentation of the typical targets used

+# ---------------------------------------------------------------------------

+

+boards := $(wildcard $(srctree)/arch/$(ARCH)/configs/\*\_defconfig)

+boards := $(notdir $(boards))

+

+help:

+ @echo 'Cleaning targets:'

+ @echo ' clean - Remove most generated files but keep the config and'

+ @echo ' enough build support to build external modules'

+ @echo ' mrproper - Remove all generated files + config + various backup files'

+ @echo ' distclean - mrproper + remove editor backup and patch files'

+ @echo ''

+ @echo 'Configuration targets:'

+ @$(MAKE) -f $(srctree)/scripts/kconfig/Makefile help

+ @echo ''

+ @echo 'Other generic targets:'

+ @echo ' all - Build all targets marked with [\*]'

+ @echo '\* vmlinux - Build the bare kernel'

+ @echo '\* modules - Build all modules'

+ @echo ' modules\_install - Install all modules to INSTALL\_MOD\_PATH (default: /)'

+ @echo ' dir/ - Build all files in dir and below'

+ @echo ' dir/file.[ois] - Build specified target only'

+ @echo ' dir/file.ko - Build module including final link'

+ @echo ' rpm - Build a kernel as an RPM package'

+ @echo ' tags/TAGS - Generate tags file for editors'

+ @echo ' cscope - Generate cscope index'

+ @echo ' kernelrelease - Output the release version string'

+ @echo ' kernelversion - Output the version stored in Makefile'

+ @if [ -r $(srctree)/include/asm-$(SRCARCH)/Kbuild ]; then \

+ echo ' headers\_install - Install sanitised kernel headers to INSTALL\_HDR\_PATH'; \

+ echo ' (default: $(INSTALL\_HDR\_PATH))'; \

+ fi

+ @echo ''

+ @echo 'Static analysers'

+ @echo ' checkstack - Generate a list of stack hogs'

+ @echo ' namespacecheck - Name space analysis on compiled kernel'

+ @echo ' export\_report - List the usages of all exported symbols'

+ @if [ -r $(srctree)/include/asm-$(SRCARCH)/Kbuild ]; then \

+ echo ' headers\_check - Sanity check on exported headers'; \

+ fi

+ @echo ''

+ @echo 'Kernel packaging:'

+ @$(MAKE) $(build)=$(package-dir) help

+ @echo ''

+ @echo 'Documentation targets:'

+ @$(MAKE) -f $(srctree)/Documentation/DocBook/Makefile dochelp

+ @echo ''

+ @echo 'Architecture specific targets ($(ARCH)):'

+ @$(if $(archhelp),$(archhelp),\

+ echo ' No architecture specific help defined for $(ARCH)')

+ @echo ''

+ @$(if $(boards), \

+ $(foreach b, $(boards), \

+ printf " %-24s - Build for %s\\n" $(b) $(subst \_defconfig,,$(b));) \

+ echo '')

+

+ @echo ' make V=0|1 [targets] 0 => quiet build (default), 1 => verbose build'

+ @echo ' make V=2 [targets] 2 => give reason for rebuild of target'

+ @echo ' make O=dir [targets] Locate all output files in "dir", including .config'

+ @echo ' make C=1 [targets] Check all c source with $$CHECK (sparse by default)'

+ @echo ' make C=2 [targets] Force check of all c source with $$CHECK'

+ @echo ''

+ @echo 'Execute "make" or "make all" to build all targets marked with [\*] '

+ @echo 'For further info see the ./README file'

+

+

+# Documentation targets

+# ---------------------------------------------------------------------------

+%docs: scripts\_basic FORCE

+ $(Q)$(MAKE) $(build)=Documentation/DocBook $@

+

+else # KBUILD\_EXTMOD

+

+###

+# External module support.

+# When building external modules the kernel used as basis is considered

+# read-only, and no consistency checks are made and the make

+# system is not used on the basis kernel. If updates are required

+# in the basis kernel ordinary make commands (without M=...) must

+# be used.

+#

+# The following are the only valid targets when building external

+# modules.

+# make M=dir clean Delete all automatically generated files

+# make M=dir modules Make all modules in specified dir

+# make M=dir Same as 'make M=dir modules'

+# make M=dir modules\_install

+# Install the modules built in the module directory

+# Assumes install directory is already created

+

+# We are always building modules

+KBUILD\_MODULES := 1

+PHONY += crmodverdir

+crmodverdir:

+ $(cmd\_crmodverdir)

+

+PHONY += $(objtree)/Module.symvers

+$(objtree)/Module.symvers:

+ @test -e $(objtree)/Module.symvers || ( \

+ echo; \

+ echo " WARNING: Symbol version dump $(objtree)/Module.symvers"; \

+ echo " is missing; modules will have no dependencies and modversions."; \

+ echo )

+

+module-dirs := $(addprefix \_module\_,$(KBUILD\_EXTMOD))

+PHONY += $(module-dirs) modules

+$(module-dirs): crmodverdir $(objtree)/Module.symvers

+ $(Q)$(MAKE) $(build)=$(patsubst \_module\_%,%,$@)

+

+modules: $(module-dirs)

+ @echo ' Building modules, stage 2.';

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modpost

+

+PHONY += modules\_install

+modules\_install: \_emodinst\_ \_emodinst\_post

+

+install-dir := $(if $(INSTALL\_MOD\_DIR),$(INSTALL\_MOD\_DIR),extra)

+PHONY += \_emodinst\_

+\_emodinst\_:

+ $(Q)mkdir -p $(MODLIB)/$(install-dir)

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modinst

+

+PHONY += \_emodinst\_post

+\_emodinst\_post: \_emodinst\_

+ $(call cmd,depmod)

+

+clean-dirs := $(addprefix \_clean\_,$(KBUILD\_EXTMOD))

+

+PHONY += $(clean-dirs) clean

+$(clean-dirs):

+ $(Q)$(MAKE) $(clean)=$(patsubst \_clean\_%,%,$@)

+

+clean: rm-dirs := $(MODVERDIR)

+clean: rm-files := $(KBUILD\_EXTMOD)/Module.symvers

+clean: $(clean-dirs)

+ $(call cmd,rmdirs)

+ $(call cmd,rmfiles)

+ @find $(KBUILD\_EXTMOD) $(RCS\_FIND\_IGNORE) \

+ \( -name '\*.[oas]' -o -name '\*.ko' -o -name '.\*.cmd' \

+ -o -name '.\*.d' -o -name '.\*.tmp' -o -name '\*.mod.c' \) \

+ -type f -print | xargs rm -f

+

+help:

+ @echo ' Building external modules.'

+ @echo ' Syntax: make -C path/to/kernel/src M=$$PWD target'

+ @echo ''

+ @echo ' modules - default target, build the module(s)'

+ @echo ' modules\_install - install the module'

+ @echo ' clean - remove generated files in module directory only'

+ @echo ''

+

+# Dummies...

+PHONY += prepare scripts

+prepare: ;

+scripts: ;

+endif # KBUILD\_EXTMOD

+

+# Generate tags for editors

+# ---------------------------------------------------------------------------

+

+#We want \_\_srctree to totally vanish out when KBUILD\_OUTPUT is not set

+#(which is the most common case IMHO) to avoid unneeded clutter in the big tags file.

+#Adding $(srctree) adds about 20M on i386 to the size of the output file!

+

+ifeq ($(src),$(obj))

+\_\_srctree =

+else

+\_\_srctree = $(srctree)/

+endif

+

+ifeq ($(ALLSOURCE\_ARCHS),)

+ifeq ($(ARCH),um)

+ALLINCLUDE\_ARCHS := $(ARCH) $(SUBARCH)

+else

+ALLINCLUDE\_ARCHS := $(SRCARCH)

+endif

+else

+#Allow user to specify only ALLSOURCE\_PATHS on the command line, keeping existing behaviour.

+ALLINCLUDE\_ARCHS := $(ALLSOURCE\_ARCHS)

+endif

+

+ALLSOURCE\_ARCHS := $(SRCARCH)

+

+define find-sources

+ ( for arch in $(ALLSOURCE\_ARCHS) ; do \

+ find $(\_\_srctree)arch/$${arch} $(RCS\_FIND\_IGNORE) \

+ -name $1 -print; \

+ done ; \

+ find $(\_\_srctree)security/selinux/include $(RCS\_FIND\_IGNORE) \

+ -name $1 -print; \

+ find $(\_\_srctree)include $(RCS\_FIND\_IGNORE) \

+ \( -name config -o -name 'asm-\*' \) -prune \

+ -o -name $1 -print; \

+ for arch in $(ALLINCLUDE\_ARCHS) ; do \

+ find $(\_\_srctree)include/asm-$${arch} $(RCS\_FIND\_IGNORE) \

+ -name $1 -print; \

+ done ; \

+ find $(\_\_srctree)include/asm-generic $(RCS\_FIND\_IGNORE) \

+ -name $1 -print; \

+ find $(\_\_srctree) $(RCS\_FIND\_IGNORE) \

+ \( -name include -o -name arch -o -name '.tmp\_\*' \) -prune -o \

+ -name $1 -print; \

+ )

+endef

+

+define all-sources

+ $(call find-sources,'\*.[chS]')

+endef

+define all-kconfigs

+ $(call find-sources,'Kconfig\*')

+endef

+define all-defconfigs

+ $(call find-sources,'defconfig')

+endef

+

+define xtags

+ if $1 --version 2>&1 | grep -iq exuberant; then \

+ $(all-sources) | xargs $1 -a \

+ -I \_\_initdata,\_\_exitdata,\_\_acquires,\_\_releases \

+ -I EXPORT\_SYMBOL,EXPORT\_SYMBOL\_GPL \

+ --extra=+f --c-kinds=+px \

+ --regex-asm='/^ENTRY\(([^)]\*)\).\*/\1/'; \

+ $(all-kconfigs) | xargs $1 -a \

+ --langdef=kconfig \

+ --language-force=kconfig \

+ --regex-kconfig='/^[[:blank:]]\*config[[:blank:]]+([[:alnum:]\_]+)/\1/'; \

+ $(all-defconfigs) | xargs -r $1 -a \

+ --langdef=dotconfig \

+ --language-force=dotconfig \

+ --regex-dotconfig='/^#?[[:blank:]]\*(CONFIG\_[[:alnum:]\_]+)/\1/'; \

+ elif $1 --version 2>&1 | grep -iq emacs; then \

+ $(all-sources) | xargs $1 -a; \

+ $(all-kconfigs) | xargs $1 -a \

+ --regex='/^[ \t]\*config[ \t]+\([a-zA-Z0-9\_]+\)/\1/'; \

+ $(all-defconfigs) | xargs -r $1 -a \

+ --regex='/^#?[ \t]?\(CONFIG\_[a-zA-Z0-9\_]+\)/\1/'; \

+ else \

+ $(all-sources) | xargs $1 -a; \

+ fi

+endef

+

+quiet\_cmd\_cscope-file = FILELST cscope.files

+ cmd\_cscope-file = (echo \-k; echo \-q; $(all-sources)) > cscope.files

+

+quiet\_cmd\_cscope = MAKE cscope.out

+ cmd\_cscope = cscope -b

+

+cscope: FORCE

+ $(call cmd,cscope-file)

+ $(call cmd,cscope)

+

+quiet\_cmd\_TAGS = MAKE $@

+define cmd\_TAGS

+ rm -f $@; \

+ $(call xtags,etags)

+endef

+

+TAGS: FORCE

+ $(call cmd,TAGS)

+

+quiet\_cmd\_tags = MAKE $@

+define cmd\_tags

+ rm -f $@; \

+ $(call xtags,ctags)

+endef

+

+tags: FORCE

+ $(call cmd,tags)

+

+

+# Scripts to check various things for consistency

+# ---------------------------------------------------------------------------

+

+includecheck:

+ find \* $(RCS\_FIND\_IGNORE) \

+ -name '\*.[hcS]' -type f -print | sort \

+ | xargs $(PERL) -w scripts/checkincludes.pl

+

+versioncheck:

+ find \* $(RCS\_FIND\_IGNORE) \

+ -name '\*.[hcS]' -type f -print | sort \

+ | xargs $(PERL) -w scripts/checkversion.pl

+

+namespacecheck:

+ $(PERL) $(srctree)/scripts/namespace.pl

+

+export\_report:

+ $(PERL) $(srctree)/scripts/export\_report.pl

+

+endif #ifeq ($(config-targets),1)

+endif #ifeq ($(mixed-targets),1)

+

+PHONY += checkstack kernelrelease kernelversion

+

+# UML needs a little special treatment here. It wants to use the host

+# toolchain, so needs $(SUBARCH) passed to checkstack.pl. Everyone

+# else wants $(ARCH), including people doing cross-builds, which means

+# that $(SUBARCH) doesn't work here.

+ifeq ($(ARCH), um)

+CHECKSTACK\_ARCH := $(SUBARCH)

+else

+CHECKSTACK\_ARCH := $(ARCH)

+endif

+checkstack:

+ $(OBJDUMP) -d vmlinux $$(find . -name '\*.ko') | \

+ $(PERL) $(src)/scripts/checkstack.pl $(CHECKSTACK\_ARCH)

+

+kernelrelease:

+ $(if $(wildcard include/config/kernel.release), $(Q)echo $(KERNELRELEASE), \

+ $(error kernelrelease not valid - run 'make prepare' to update it))

+kernelversion:

+ @echo $(KERNELVERSION)

+

+# Single targets

+# ---------------------------------------------------------------------------

+# Single targets are compatible with:

+# - build whith mixed source and output

+# - build with separate output dir 'make O=...'

+# - external modules

+#

+# target-dir => where to store outputfile

+# build-dir => directory in kernel source tree to use

+

+ifeq ($(KBUILD\_EXTMOD),)

+ build-dir = $(patsubst %/,%,$(dir $@))

+ target-dir = $(dir $@)

+else

+ zap-slash=$(filter-out .,$(patsubst %/,%,$(dir $@)))

+ build-dir = $(KBUILD\_EXTMOD)$(if $(zap-slash),/$(zap-slash))

+ target-dir = $(if $(KBUILD\_EXTMOD),$(dir $<),$(dir $@))

+endif

+

+%.s: %.c prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.i: %.c prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.o: %.c prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.lst: %.c prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.s: %.S prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.o: %.S prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+%.symtypes: %.c prepare scripts FORCE

+ $(Q)$(MAKE) $(build)=$(build-dir) $(target-dir)$(notdir $@)

+

+# Modules

+/ %/: prepare scripts FORCE

+ $(cmd\_crmodverdir)

+ $(Q)$(MAKE) KBUILD\_MODULES=$(if $(CONFIG\_MODULES),1) \

+ $(build)=$(build-dir)

+%.ko: prepare scripts FORCE

+ $(cmd\_crmodverdir)

+ $(Q)$(MAKE) KBUILD\_MODULES=$(if $(CONFIG\_MODULES),1) \

+ $(build)=$(build-dir) $(@:.ko=.o)

+ $(Q)$(MAKE) -f $(srctree)/scripts/Makefile.modpost

+

+# FIXME Should go into a make.lib or something

+# ===========================================================================

+

+quiet\_cmd\_rmdirs = $(if $(wildcard $(rm-dirs)),CLEAN $(wildcard $(rm-dirs)))

+ cmd\_rmdirs = rm -rf $(rm-dirs)

+

+quiet\_cmd\_rmfiles = $(if $(wildcard $(rm-files)),CLEAN $(wildcard $(rm-files)))

+ cmd\_rmfiles = rm -f $(rm-files)

+

+# Run depmod only is we have System.map and depmod is executable

+# and we build for the host arch

+quiet\_cmd\_depmod = DEPMOD $(KERNELRELEASE)

+ cmd\_depmod = \

+ if [ -r System.map -a -x $(DEPMOD) ]; then \

+ $(DEPMOD) -ae -F System.map \

+ $(if $(strip $(INSTALL\_MOD\_PATH)), -b $(INSTALL\_MOD\_PATH) -r) \

+ $(KERNELRELEASE); \

+ fi

+

+# Create temporary dir for module support files

+# clean it up only when building all modules

+cmd\_crmodverdir = $(Q)mkdir -p $(MODVERDIR) \

+ $(if $(KBUILD\_MODULES),; rm -f $(MODVERDIR)/\*)

+

+a\_flags = -Wp,-MD,$(depfile) $(KBUILD\_AFLAGS) $(AFLAGS\_KERNEL) \

+ $(NOSTDINC\_FLAGS) $(KBUILD\_CPPFLAGS) \

+ $(modkern\_aflags) $(EXTRA\_AFLAGS) $(AFLAGS\_$(basetarget).o)

+

+quiet\_cmd\_as\_o\_S = AS $@

+cmd\_as\_o\_S = $(CC) $(a\_flags) -c -o $@ $<

+

+# read all saved command lines

+

+targets := $(wildcard $(sort $(targets)))

+cmd\_files := $(wildcard .\*.cmd $(foreach f,$(targets),$(dir $(f)).$(notdir $(f)).cmd))

+

+ifneq ($(cmd\_files),)

+ $(cmd\_files): ; # Do not try to update included dependency files

+ include $(cmd\_files)

+endif

+

+# Shorthand for $(Q)$(MAKE) -f scripts/Makefile.clean obj=dir

+# Usage:

+# $(Q)$(MAKE) $(clean)=dir

+clean := -f $(if $(KBUILD\_SRC),$(srctree)/)scripts/Makefile.clean obj

+

+endif # skip-makefile

+

+PHONY += FORCE

+FORCE:

+

+# Declare the contents of the .PHONY variable as phony. We keep that

+# information in a variable se we can use it in if\_changed and friends.

+.PHONY: $(PHONY)