HanOS

0.1.1.220103

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 acpi_gas_t Struct Reference

Data Fields

- uint8_t **addr_space_id**
- uint8_t **reg_bit_width**
- uint8_t **reg_bit_offset**
- uint8_t **reserved**
- uint64_t **address**

The documentation for this struct was generated from the following file:

- kernel/core/acpi.h

3.2 acpi_sdt_hdr_t Struct Reference

Data Fields

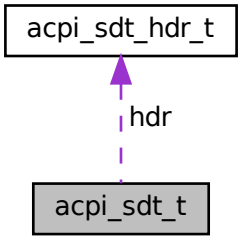
- char **sign** [4]
- uint32_t **length**
- uint8_t **rev**
- uint8_t **chksum**
- char **oem_id** [6]
- char **oem_table_id** [8]
- uint32_t **oem_rev**
- uint32_t **creator_id**
- uint32_t **creator_rev**

The documentation for this struct was generated from the following file:

- kernel/core/acpi.h

3.3 acpi_sdt_t Struct Reference

Collaboration diagram for acpi_sdt_t:



Data Fields

- acpi_sdt_hdr_t **hdr**
- uint8_t **data** [ ]

The documentation for this struct was generated from the following file:

- kernel/core/acpi.h

3.4 addrspace_t Struct Reference

Data Fields

- uint64_t * **PML4**

The documentation for this struct was generated from the following file:

- kernel/core/mm.h

3.5 cmos_rtc_t Struct Reference

Data Fields

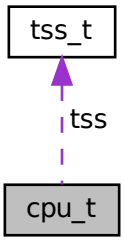
- uint8_t **seconds**
- uint8_t **minutes**
- uint8_t **hours**
- uint8_t **weekdays**
- uint8_t **day**
- uint8_t **month**
- uint16_t **year**
- uint8_t **century**

The documentation for this struct was generated from the following file:

- kernel/core/cmos.h

3.6 cpu_t Struct Reference

Collaboration diagram for cpu_t:



Data Fields

- uint16_t **cpu_id**
- uint16_t **lapic_id**
- bool **is_bsp**
- tss_t **tss**

The documentation for this struct was generated from the following file:

- kernel/core/smp.h

3.7 cpuid_feature_t Struct Reference

Public Types

- enum { **CPUID_REG_EAX**, **CPUID_REG_EBX**, **CPUID_REG_ECX**, **CPUID_REG_EDX** }

Data Fields

- uint32_t **func**
- uint32_t **param**
- enum cpuid_feature_t:: { ... } **reg**
- uint32_t **mask**

The documentation for this struct was generated from the following file:

- kernel/core/cpu.h

3.8 gdt_register_t Struct Reference

Data Fields

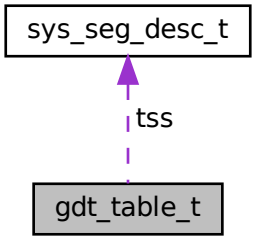
- uint16_t **size**
- uint64_t **offset**

The documentation for this struct was generated from the following file:

- kernel/core/gdt.h

3.9 gdt_table_t Struct Reference

Collaboration diagram for gdt_table_t:



Data Fields

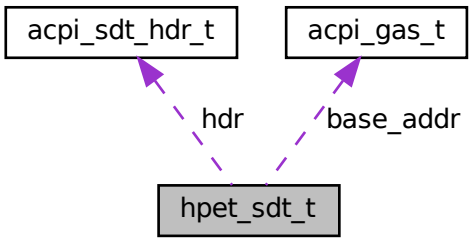
- gdt_entry_t **null**
- gdt_entry_t **kcode**
- gdt_entry_t **kdata**
- gdt_entry_t **ucode**
- gdt_entry_t **udata**
- sys_seg_desc_t **tss**

The documentation for this struct was generated from the following file:

- kernel/core/gdt.h

3.10 hpet_sdt_t Struct Reference

Collaboration diagram for hpet_sdt_t:



Data Fields

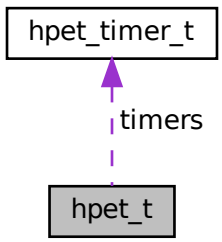
- acpi_sdt_hdr_t **hdr**
- uint8_t **hardware_rev_id**
- uint8_t **comparator_count**: 5
- uint8_t **counter_size**: 1
- uint8_t **reserved**: 1
- uint8_t **legacy_replace**: 1
- uint16_t **pci_vendor_id**
- acpi_gas_t **base_addr**
- uint8_t **hpet_number**
- uint16_t **minimum_tick**
- uint8_t **page_protection**

The documentation for this struct was generated from the following file:

- kernel/core/hpet.h

3.11 hpet_t Struct Reference

Collaboration diagram for hpet_t:



Data Fields

- volatile uint64_t **general_capabilities**
- volatile uint64_t **unused0**
- volatile uint64_t **general_configuration**
- volatile uint64_t **unused1**
- volatile uint64_t **general_int_status**
- volatile uint64_t **unused2**
- volatile uint64_t **unused3** [2][12]
- volatile uint64_t **main_counter_value**
- volatile uint64_t **unused4**
- hpet_timer_t **timers** [ ]

The documentation for this struct was generated from the following file:

- kernel/core/hpet.h

3.12 hpet_timer_t Struct Reference

Data Fields

- volatile uint64_t **config_and_capabilities**
- volatile uint64_t **comparator_value**
- volatile uint64_t **fsb_interrupt_route**
- volatile uint64_t **unused**

The documentation for this struct was generated from the following file:

- kernel/core/hpet.h

3.13 idt_entry_t Struct Reference

Data Fields

- uint16_t **offset_1**
- uint16_t **selector**
- uint8_t **ist**
- uint8_t **type_attributes**
- uint16_t **offset_2**
- uint32_t **offset_3**
- uint32_t **zero**

The documentation for this struct was generated from the following file:

- kernel/core/idt.h

3.14 idt_register_t Struct Reference

Data Fields

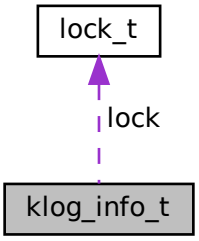
- uint16_t **size**
- uint64_t **offset**

The documentation for this struct was generated from the following file:

- kernel/core/idt.h

3.15 klog_info_t Struct Reference

Collaboration diagram for klog_info_t:



Data Fields

- uint8_t **buff** [KLOG_BUFFER_SIZE]
- int **start**
- int **end**
- term_info_t * **term**
- lock_t **lock**

The documentation for this struct was generated from the following file:

- kernel/lib/klog.h

3.16 lock_t Struct Reference

Data Fields

- int **lock**
- uint64_t **rflags**

The documentation for this struct was generated from the following file:

- kernel/lib/lock.h

3.17 madt_record_hdr_t Struct Reference

Data Fields

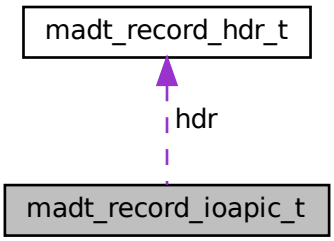
- uint8_t **type**
- uint8_t **len**

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.18 madt_record_ioapic_t Struct Reference

Collaboration diagram for madt_record_ioapic_t:



Data Fields

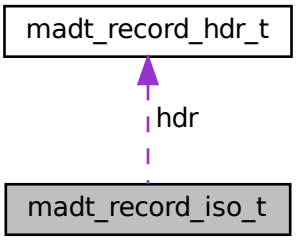
- madt_record_hdr_t **hdr**
- uint8_t **id**
- uint8_t **reserved**
- uint32_t **addr**
- uint32_t **gsi_base**

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.19 madt_record_iso_t Struct Reference

Collaboration diagram for madt_record_iso_t:



Data Fields

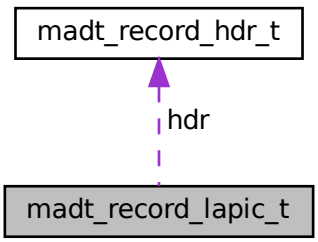
- madt_record_hdr_t **hdr**
- uint8_t **bus_src**
- uint8_t **irq_src**
- uint32_t **gsi**
- uint16_t **flags**

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.20 madt_record_lapic_t Struct Reference

Collaboration diagram for madt_record_lapic_t:



Data Fields

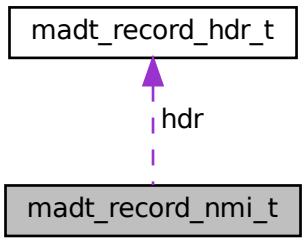
- madt_record_hdr_t **hdr**
- uint8_t **proc_id**
- uint8_t **apic_id**
- uint32_t **flags**

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.21 madt_record_nmi_t Struct Reference

Collaboration diagram for madt_record_nmi_t:



Data Fields

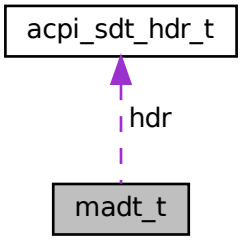
- madt_record_hdr_t **hdr**
- uint8_t **proc_id**
- uint16_t **flags**
- uint8_t **lint**

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.22 madt_t Struct Reference

Collaboration diagram for madt_t:



Data Fields

- acpi_sdt_hdr_t **hdr**
- uint32_t **lapic_addr**
- uint32_t **flags**
- uint8_t **records** [ ]

The documentation for this struct was generated from the following file:

- kernel/core/madt.h

3.23 mem_info_t Struct Reference

Data Fields

- uint64_t **phys_limit**
- uint64_t **total_size**
- uint64_t **free_size**
- uint8_t * **bitmap**

The documentation for this struct was generated from the following file:

- kernel/core/mm.h

3.24 metadata_t Struct Reference

Data Fields

- size_t **numpages**
- size_t **size**

The documentation for this struct was generated from the following file:

- kernel/lib/kmalloc.c

3.25 rsdp_t Struct Reference

Data Fields

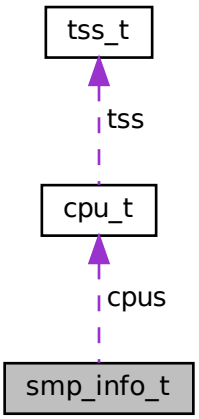
- char **sign** [8]
- uint8_t **chksum**
- char **oem_id** [6]
- uint8_t **revision**
- uint32_t **rsdt_addr**
- uint32_t **length**
- uint64_t **xsdt_addr**
- uint8_t **chksum_ext**
- uint8_t **reserved** [3]

The documentation for this struct was generated from the following file:

- kernel/core/acpi.h

3.26 smp_info_t Struct Reference

Collaboration diagram for smp_info_t:



Data Fields

- uint16_t **num_cpus**
- cpu_t **cpus** [CPU_MAX]

The documentation for this struct was generated from the following file:

- kernel/core/smp.h

3.27 symbol_t Struct Reference

Data Fields

- uint64_t **addr**
- char * **name**

The documentation for this struct was generated from the following file:

- kernel/symbols.h

3.28 sys_seg_desc_t Struct Reference

Data Fields

- uint16_t **seg_limit_1**
- uint16_t **base_addr_1**
- uint8_t **base_addr_2**
- uint8_t **flags_low**
- uint8_t **flags_high**
- uint8_t **base_addr_3**
- uint32_t **base_addr_4**
- uint32_t **reserved**

The documentation for this struct was generated from the following file:

- kernel/core/gdt.h

3.29 timeval_t Struct Reference

Data Fields

- uint64_t **sec**
- uint64_t **usec**

The documentation for this struct was generated from the following file:

- kernel/lib/time.h

3.30 timezone_t Struct Reference

Data Fields

- int **minuteswest**
- int **dsttime**

The documentation for this struct was generated from the following file:

- kernel/lib/time.h

3.31 tm_t Struct Reference

Data Fields

- int **sec**
- int **min**
- int **hour**
- int **mday**
- int **mon**
- int **year**
- int **wday**
- int **yday**
- int **isdst**

The documentation for this struct was generated from the following file:

- kernel/lib/time.h

3.32 tss_t Struct Reference

Data Fields

- uint32_t **unused0**
- uint64_t **rsp0**
- uint64_t **rsp1**
- uint64_t **rsp2**
- uint64_t **unused1**
- uint64_t **ist1**
- uint64_t **ist2**
- uint64_t **ist3**
- uint64_t **ist4**
- uint64_t **ist5**
- uint64_t **ist6**
- uint64_t **ist7**
- uint64_t **unused2**
- uint32_t **iopb_offset**

The documentation for this struct was generated from the following file:

- kernel/core/smp.h

Chapter 4

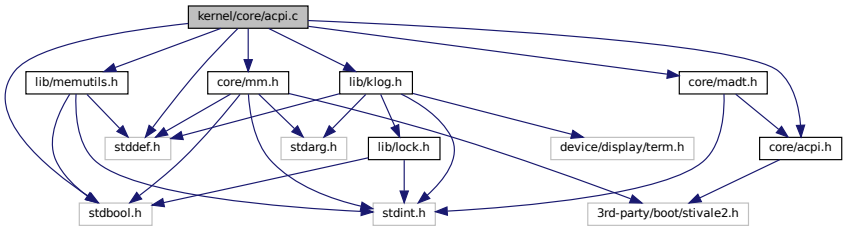
File Documentation

4.1 kernel/core/acpi.c File Reference

Implementation of ACPI (Advanced Configuration and Power Management Interface) functions.

```
#include <stdbool.h>
#include <stddef.h>
#include <core/acpi.h>
#include <core/madt.h>
#include <core/mm.h>
#include <lib/klog.h>
#include <lib/memutils.h>
```

Include dependency graph for acpi.c:



Functions

- acpi_sdt_t * **acpi_get_sdt** (const char *sign)
- void **acpi_init** (struct stivale2_struct_tag_rsdp *rsdp_info)

4.1.1 Detailed Description

Implementation of ACPI (Advanced Configuration and Power Management Interface) functions.
This module includes implementation of RSDT/XSDT initialization and "MADT/HPET" parsing.

Author

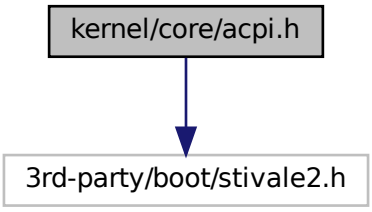
JW

Date

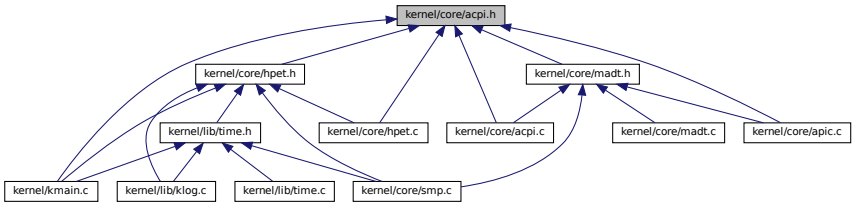
DEC 12, 2021

4.2 kernel/core/acpi.h File Reference

Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures.
`#include <3rd-party/boot/stivale2.h>`
Include dependency graph for acpi.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct rsdp_t
- struct acpi_sdt_hdr_t
- struct acpi_sdt_t
- struct acpi_gas_t

Functions

- void **acpi_init** (struct stivale2_struct_tag_rsdp ∗)
- acpi_sdt_t ∗ **acpi_get_sdt** (const char ∗sign)

4.2.1 Detailed Description

Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures.
ACPI (Advanced Configuration and Power Interface) is a Power Management and configuration standard for the PC, developed by Intel, Microsoft and Toshiba. ACPI allows the operating system to control the amount of power each device is given (allowing it to put certain devices on standby or power-off for example). It is also used to control and/or check thermal zones (temperature sensors, fan speeds, etc), battery levels, PCI IRQ routing, CPUs, NUMA domains and many other things.
There are 2 main parts to ACPI. The first part is the tables used by the OS for configuration during boot (these include things like how many CPUs, APIC details, NUMA memory ranges, etc). The second part is the run time ACPI environment, which consists of AML code (a platform independent OOP language that comes from the BIOS and devices) and the ACPI SMM (System Management Mode) code.

Author

JW

Date

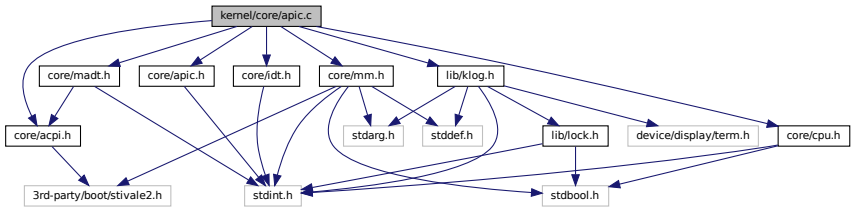
DEC 12, 2021

4.3 kernel/core/apic.c File Reference

Implementation of APIC (Advanced Programmable Interrupt Controller) functions.

```
#include <core/acpi.h>
#include <core/apic.h>
#include <core/madt.h>
#include <core/mm.h>
#include <core/cpu.h>
#include <core/idt.h>
#include <lib/klog.h>
```

Include dependency graph for apic.c:



Functions

- uint32_t **apic_read_reg** (uint16_t offset)
- void **apic_write_reg** (uint16_t offset, uint32_t val)
- void **apic_send_eoi** ()
- void **apic_send_ipi** (uint8_t dest, uint8_t vector, uint32_t mtype)
- void **apic_enable** ()
- void **apic_init** ()

4.3.1 Detailed Description

Implementation of APIC (Advanced Programmable Interrupt Controller) functions.
Need further work on APIC...

Author

JW

Date

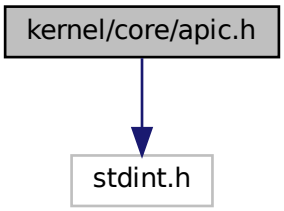
DEC 12, 2021

4.4 kernel/core/apic.h File Reference

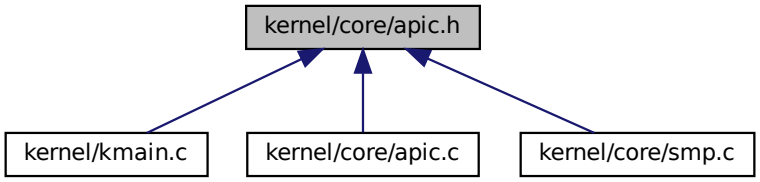
Definition of APIC (Advanced Programmable Interrupt Controller) related data structures.

```
#include <stdint.h>
```

Include dependency graph for apic.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **APIC_REG_ID** 0x20
- #define **APIC_REG_VERSION** 0x30
- #define **APIC_REG_SPURIOUS_INT** 0xF0
- #define **APIC_REG_EOI** 0xB0
- #define **APIC_REG_ICR_LOW** 0x300
- #define **APIC_REG_ICR_HIGH** 0x310
- #define **APIC_SPURIOUS_VECTOR_NUM** 0xFF
- #define **APIC_FLAG_ENABLE** (1 << 8)
- #define **APIC_IPI_TYPE_INIT** 0b101
- #define **APIC_IPI_TYPE_STARTUP** 0b110

Functions

- void **apic_init** (void)
- void **apic_enable** ()
- uint32_t **apic_read_reg** (uint16_t offset)
- void **apic_write_reg** (uint16_t offset, uint32_t val)
- void **apic_send_eoi** ()
- void **apic_send_ipi** (uint8_t dest, uint8_t vector, uint32_t mtype)

4.4.1 Detailed Description

Definition of APIC (Advanced Programmable Interrupt Controller) related data structures.
APIC is used in multiprocessor systems and is an integral part of all recent Intel (and compatible) processors. The APIC is used for sophisticated interrupt redirection, and for sending interrupts between processors.

Author

JW

Date

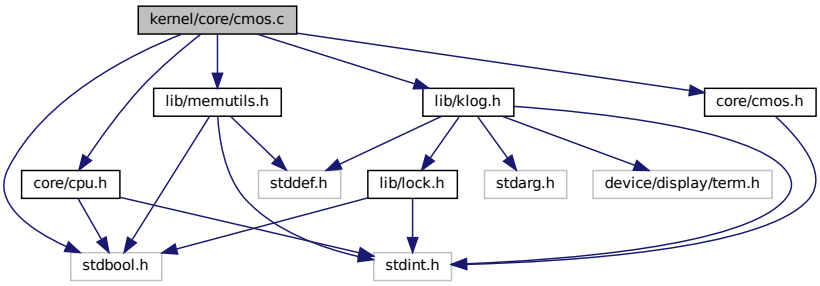
DEC 12, 2021

4.5 kernel/core/cmos.c File Reference

Implementation of CMOS related functions.

```
#include <stdbool.h>
#include <core/cmos.h>
#include <core/cpu.h>
#include <lib/memutils.h>
#include <lib/klog.h>
```

Include dependency graph for cmos.c:



Functions

- uint8_t **get_rtc_register** (uint8_t reg)
- bool **update_in_progress** ()
- bool **rtc_values_are_not_equal** (cmos_rtc_t c1, cmos_rtc_t c2)
- uint64_t **secs_of_month** (uint64_t months, uint64_t year)
- uint64_t **secs_of_years** (uint64_t years)
- void **cmos_init** ()
- uint64_t **cmos_boot_time** ()
- cmos_rtc_t **cmos_read_rtc** ()

4.5.1 Detailed Description

Implementation of CMOS related functions.
"CMOS" is a tiny bit of very low power static memory that lives on the same chip as the Real-Time Clock (RTC). It was introduced to IBM PC AT in 1984 which used Motorola MC146818A RTC. Ref: https://wiki.osdev.org/CMOS

Author

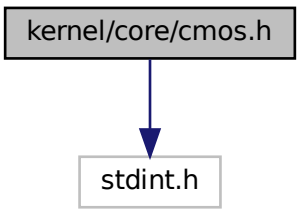
JW

Date

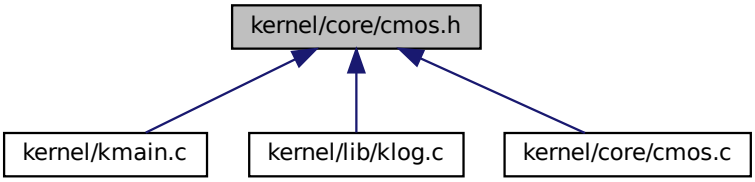
Jan 2, 2022

4.6 kernel/core/cmos.h File Reference

Definition of CMOS related data structures.
`#include <stdint.h>`
Include dependency graph for cmos.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct cmos_rtc_t

Macros

- #define **CMOS_COMMAND_PORT** 0x70
- #define **CMOS_DATA_PORT** 0x71
- #define **CMOS_REG_SECONDS** 0x00
- #define **CMOS_REG_MINUTES** 0x02
- #define **CMOS_REG_HOURS** 0x04
- #define **CMOS_REG_WEEKDAYS** 0x06
- #define **CMOS_REG_DAY** 0x07
- #define **CMOS_REG_MONTH** 0x08
- #define **CMOS_REG_YEAR** 0x09
- #define **CMOS_REG_CENTURY** 0x32
- #define **CMOS_REG_STATUS_A** 0x0A
- #define **CMOS_REG_STATUS_B** 0x0B

Functions

- void **cmos_init** ()
- cmos_rtc_t **cmos_read_rtc** ()
- uint64_t **cmos_boot_time** ()

4.6.1 Detailed Description

Definition of CMOS related data structures.
"CMOS" is a tiny bit of very low power static memory that lives on the same chip as the Real-Time Clock (RTC). It was introduced to IBM PC AT in 1984 which used Motorola MC146818A RTC.

Author

JW

Date

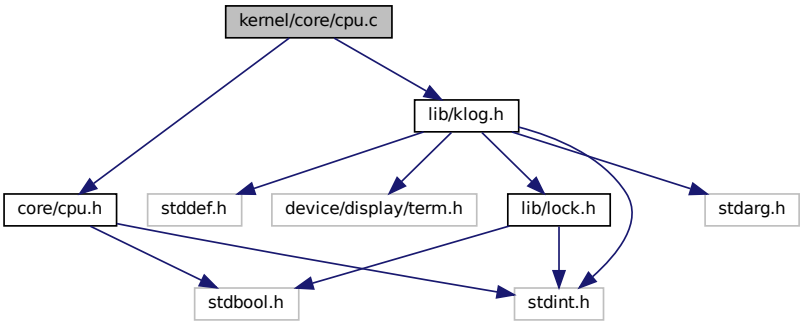
Jan 2, 2022

4.7 kernel/core/cpu.c File Reference

Implementation of CPU related functions.

```
#include <core/cpu.h>
#include <lib/klog.h>
```

Include dependency graph for cpu.c:



Functions

- void **cpuid** (uint32_t func, uint32_t param, uint32_t *eax, uint32_t *ebx, uint32_t *ecx, uint32_t *edx)
- bool **cpuid_check_feature** (cpuid_feature_t feature)
- void **cpu_init** ()

4.7.1 Detailed Description

Implementation of CPU related functions.
e.g., CPU initialization...

Author

JW

Date

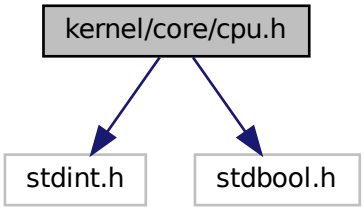
Jan 2, 2022

4.8 kernel/core/cpu.h File Reference

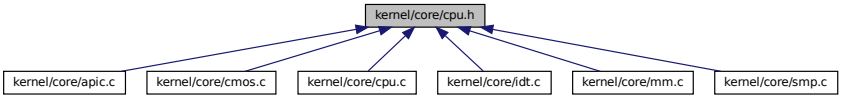
Definition of CPU related data structures and macros.

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for cpu.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct cpuid_feature_t

Macros

- #define **MSR_PAT** 0x0277
- #define **MSR_GS_BASE** 0xC0000102
- #define **MSR_EFER** 0xC0000080
- #define **MSR_STAR** 0xC0000081
- #define **MSR_LSTAR** 0xC0000082
- #define **MSR_SFMASK** 0xC0000084
- #define **read_cr**(cr, n)
- #define **write_cr**(cr, n)

Functions

- void **cpu_init** ()
- bool **cpuid_check_feature** (cpuid_feature_t feature)

4.8.1 Detailed Description

Definition of CPU related data structures and macros.
e.g., Read & write control registers, model specific registers and port input & output.

Author

JW

Date

Jan 2, 2022

4.8.2 Macro Definition Documentation

4.8.2.1 read_cr

```
#define read_cr(
              cr,
              n )
```

Value:

```
asm volatile("mov %%" cr ", %%rax;" \
"mov %%rax, %0"                     \
: "=g"(*(n))                        \
:                                   \
: "rax");
```

4.8.2.2 write_cr

```
#define write_cr(
              cr,
              n )
```

Value:

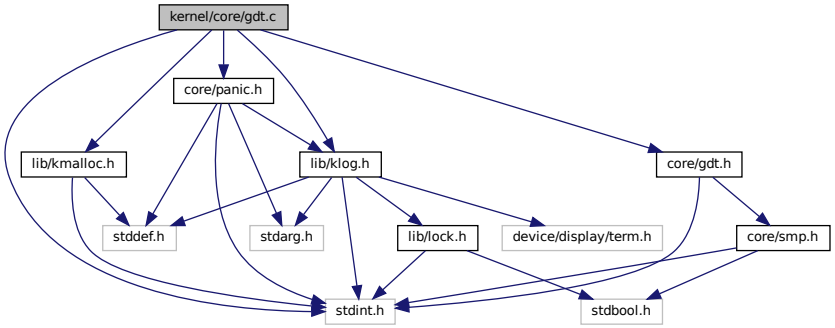
```
asm volatile("mov %0, %%rax;"       \
"mov %%rax, %%" cr                  \
:                                   \
: "g"(n)                            \
: "rax");
```

4.9 kernel/core/gdt.c File Reference

Implementation of GDT related functions.

```
#include <stdint.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
#include <core/gdt.h>
#include <core/panic.h>
```

Include dependency graph for gdt.c:



Functions

- void **gdt_init** ()
- void **gdt_install_tss** (tss_t *tss)

Variables

- gdt_table_t * **gdt** = NULL

4.9.1 Detailed Description

Implementation of GDT related functions.

The Global Descriptor Table (GDT) contains entries telling the CPU about memory segments.
In HanOS, GDT initialization is very simple. Only memory protection is used. As the first step, only two ring-0 segment descriptor are defined. The memory regions are all from 0 to 4GB.

Author

JW

Date

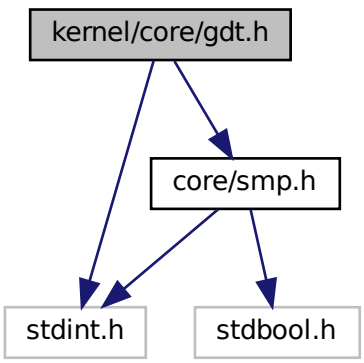
Nov 27, 2021

4.10 kernel/core/gdt.h File Reference

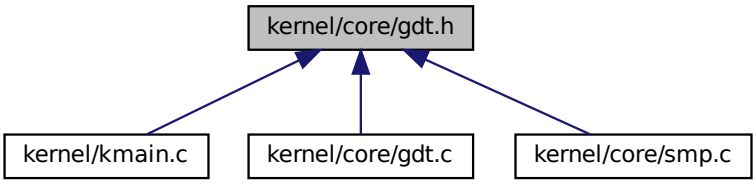
Definition of GDT related data structures.

```
#include <stdint.h>
#include <core/smp.h>
```

Include dependency graph for gdt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sys_seg_desc_t
- struct gdt_table_t
- struct gdt_register_t

Typedefs

- typedef uint64_t **gdt_entry_t**

Functions

- void **gdt_init** ()
- void **gdt_install_tss** (tss_t *tss)

4.10.1 Detailed Description

Definition of GDT related data structures.
The Global Descriptor Table (GDT) contains entries telling the CPU about memory segments.

Author

JW

Date

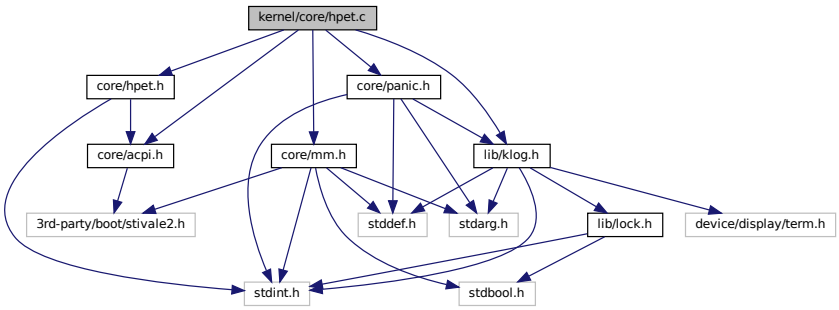
Nov 27, 2021

4.11 kernel/core/hpet.c File Reference

Implementation of HPET (High Precision Event Timer) functions.

```
#include <lib/klog.h>
#include <core/hpet.h>
#include <core/acpi.h>
#include <core/mm.h>
#include <core/panic.h>
```

Include dependency graph for hpet.c:



Functions

- uint64_t **hpet_get_nanos** ()
- void **hpet_nanosleep** (uint64_t nanos)
- void **hpet_init** ()

4.11.1 Detailed Description

Implementation of HPET (High Precision Event Timer) functions.
This module includes implementation of HPET initialization, getting nanos and nanos sleep.
HPET consists of (usually 64-bit) main counter (which counts up), as well as from 3 to 32 32-bit or 64-bit wide comparators. HPET is programmed using memory mapped IO, and the base address of HPET can be found using ACPI.
General initialization:

1. Find HPET base address in 'HPET' ACPI table.
2. Calculate HPET frequency (f = 10^15 / period).
3. Save minimal tick (either from ACPI table or configuration register).
4. Initialize comparators.
5. Set ENABLE_CNF bit.

Author

JW

Date

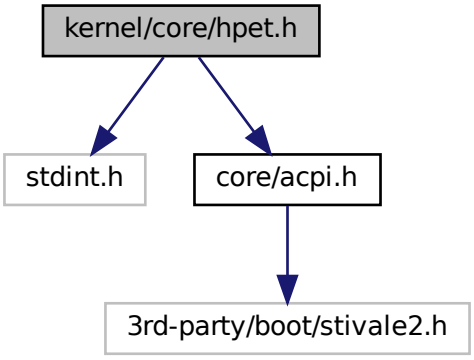
DEC 12, 2021

4.12 kernel/core/hpet.h File Reference

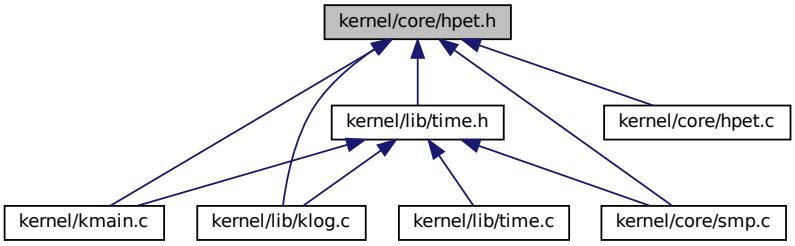
Definition of HPET (High Precision Event Timer) related data structures.

```
#include <stdint.h>
#include <core/acpi.h>
```

Include dependency graph for hpet.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct hpet_sdt_t
- struct hpet_timer_t
- struct hpet_t

Functions

- void **hpet_init** ()
- uint64_t **hpet_get_nanos** ()
- void **hpet_nanosleep** (uint64_t nanos)

4.12.1 Detailed Description

Definition of HPET (High Precision Event Timer) related data structures.
HPET, or High Precision Event Timer, is a piece of hardware designed by Intel and Microsoft to replace older PIT and RTC. It consists of (usually 64-bit) main counter (which counts up), as well as from 3 to 32 32-bit or 64-bit wide comparators. HPET is programmed using memory mapped IO, and the base address of HPET can be found using ACPI.

Author

JW

Date

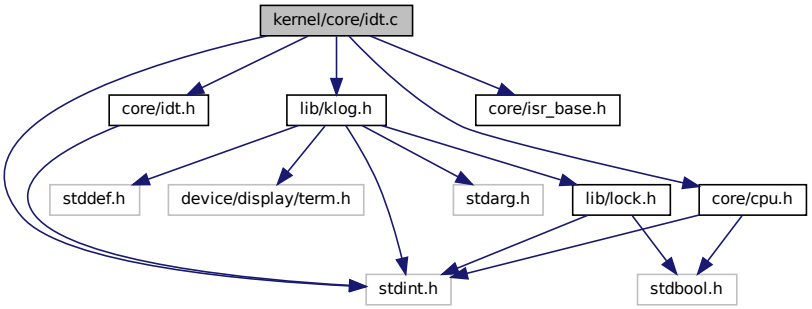
DEC 12, 2021

4.13 kernel/core/idt.c File Reference

Implementation of IDT related functions.

```
#include <stdint.h>
#include <lib/klog.h>
#include <core/isr_base.h>
#include <core/idt.h>
#include <core/cpu.h>
```

Include dependency graph for idt.c:



Functions

- void **idt_init** ()

4.13.1 Detailed Description

Implementation of IDT related functions.

The Interrupt Descriptor Table (IDT) telling the CPU where the Interrupt Service Routines (ISR) are located (one per interrupt vector). The IDT entries are called gates. It can contain Interrupt Gates, Task Gates and Trap Gates. As the first step, only trap gates (exceptions) are implemented.

Ref: https://wiki.osdev.org/Interrupt_Descriptor_Table

Author

JW

Date

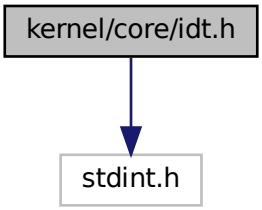
Nov 27, 2021

4.14 kernel/core/idt.h File Reference

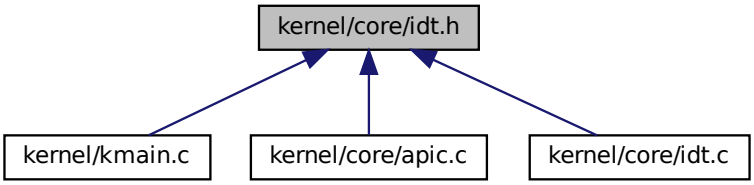
Definition of IDT related data structures.

```
#include <stdint.h>
```

Include dependency graph for idt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct idt_entry_t
- struct idt_register_t

Macros

- #define **IDT_ENTRIES** 256
- #define **IDT_DEFAULT_TYPE_ATTRIBUTES** 0b10001110

Functions

- void **idt_init** ()

4.14.1 Detailed Description

Definition of IDT related data structures.

The Interrupt Descriptor Table (IDT) telling the CPU where the Interrupt Service Routines (ISR) are located (one per interrupt vector).

Author

JW

Date

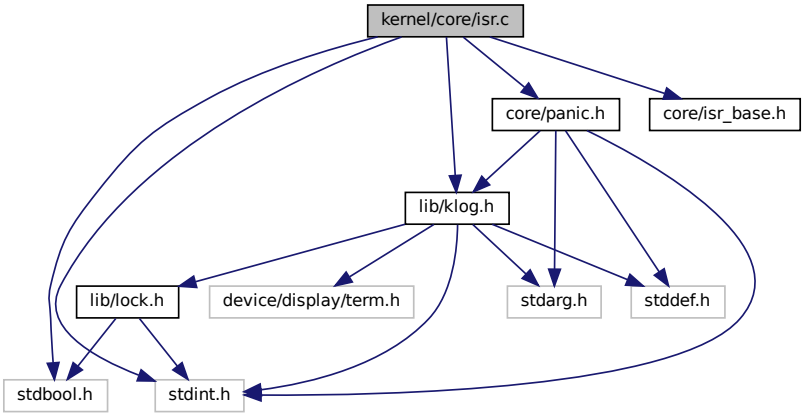
Nov 27, 2021

4.15 kernel/core/isr.c File Reference

Implementation of ISR related functions.

```
#include <stdbool.h>
#include <stdint.h>
#include <lib/klog.h>
#include <core/isr_base.h>
#include <core/panic.h>
```

Include dependency graph for isr.c:



Functions

- void **exc_handler_proc** (uint64_t errcode, uint64_t excno)

4.15.1 Detailed Description

Implementation of ISR related functions.

The x86 architecture is an interrupt driven system. Only a common interrupt handling function is implemented.

Author

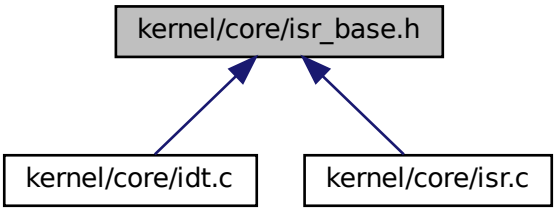
JW

Date

Nov 27, 2021

4.16 kernel/core/isr_base.h File Reference

Definition of ISR related data structures.
This graph shows which files directly or indirectly include this file:



Typedefs

- typedef void(∗ **exc_handler_t**) ()

Functions

- void **exc_register_handler** (uint64_t id, exc_handler_t handler)
- void **exc0** (void ∗p)
- void **exc1** (void ∗p)
- void **exc2** (void ∗p)
- void **exc3** (void ∗p)
- void **exc4** (void ∗p)
- void **exc5** (void ∗p)
- void **exc6** (void ∗p)
- void **exc7** (void ∗p)
- void **exc8** (void ∗p)
- void **exc10** (void ∗p)
- void **exc11** (void ∗p)
- void **exc12** (void ∗p)
- void **exc13** (void ∗p)
- void **exc14** (void ∗p)
- void **exc16** (void ∗p)
- void **exc17** (void ∗p)
- void **exc18** (void ∗p)
- void **exc19** (void ∗p)
- void **exc20** (void ∗p)
- void **exc30** (void ∗p)

4.16.1 Detailed Description

Definition of ISR related data structures.
The x86 architecture is an interrupt driven system.

Author

JW

Date

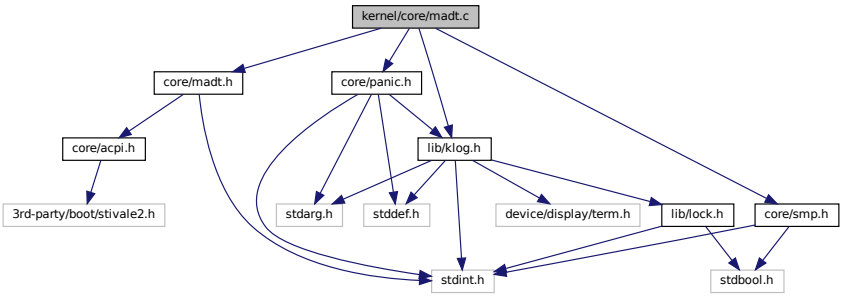
Nov 27, 2021

4.17 kernel/core/madt.c File Reference

Implementation of ACPI MADT (Multiple APIC Description Table) functions.

```
#include <core/madt.h>
#include <core/panic.h>
#include <core/smp.h>
#include <lib/klog.h>
```

Include dependency graph for madt.c:



Functions

- uint32_t **madt_get_num_ioapic** ()
- uint32_t **madt_get_num_lapic** ()
- madt_record_ioapic_t ** **madt_get_ioapics** ()
- madt_record_lapic_t ** **madt_get_lapics** ()
- uint64_t **madt_get_lapic_base** ()
- void **madt_init** ()

4.17.1 Detailed Description

Implementation of ACPI MADT (Multiple APIC Description Table) functions.
The MADT describes all of the interrupt controllers in the system. It can be used to enumerate the processors currently available.

Author

JW

Date

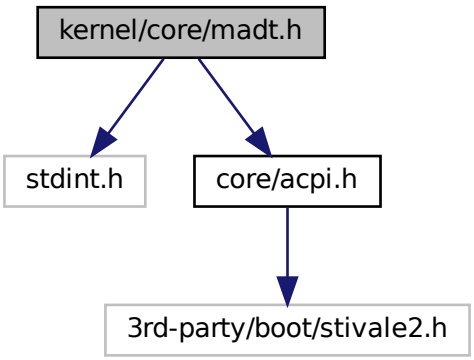
DEC 25, 2021

4.18 kernel/core/madt.h File Reference

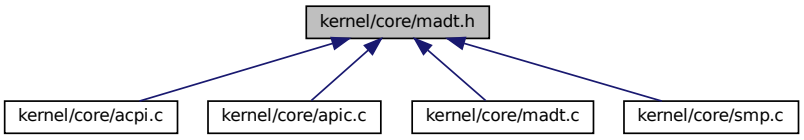
Definition of ACPI MADT (Multiple APIC Description Table) related data structures.

```
#include <stdint.h>
#include <core/acpi.h>
```

Include dependency graph for madt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct madt_record_hdr_t
- struct madt_record_lapic_t
- struct madt_record_ioapic_t
- struct madt_record_iso_t
- struct madt_record_nmi_t
- struct madt_t

Macros

- #define **MADT_RECORD_TYPE_LAPIC** 0
- #define **MADT_RECORD_TYPE_IOAPIC** 1
- #define **MADT_RECORD_TYPE_ISO** 2
- #define **MADT_RECORD_TYPE_NMI** 4
- #define **MADT_RECORD_TYPE_LAPIC_AO** 5
- #define **MADT_LAPIC_FLAG_ENABLED** (1 << 0)
- #define **MADT_LAPIC_FLAG_ONLINE_CAPABLE** (1 << 1)

Functions

- void **madt_init** ()
- uint32_t **madt_get_num_ioapic** ()
- uint32_t **madt_get_num_lapic** ()
- madt_record_ioapic_t ** **madt_get_ioapics** ()
- madt_record_lapic_t ** **madt_get_lapics** ()
- uint64_t **madt_get_lapic_base** ()

4.18.1 Detailed Description

Definition of ACPI MADT (Multiple APIC Description Table) related data structures.
The MADT describes all of the interrupt controllers in the system. It can be used to enumerate the processors currently available.

Author

JW

Date

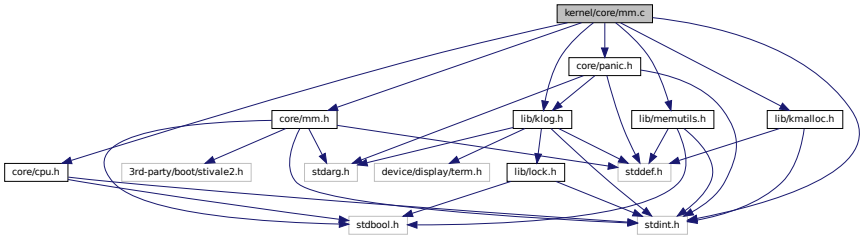
DEC 25, 2021

4.19 kernel/core/mm.c File Reference

Implementation of memory management functions.

```
#include <stdint.h>
#include <core/cpu.h>
#include <core/mm.h>
#include <core/panic.h>
#include <lib/memutils.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
```

Include dependency graph for mm.c:



Macros

- #define **MAKE_TABLE_ENTRY**(address, flags) ((address & ∼(0xfff)) | flags)

Functions

- void **pmm_free** (uint64_t addr, uint64_t numpages)
- bool **pmm_alloc** (uint64_t addr, uint64_t numpages)
- uint64_t **pmm_get** (uint64_t numpages)
- void **pmm_init** (struct stivale2_struct_tag_memmap *map)
- void **vmm_unmap** (uint64_t vaddr, uint64_t np)
- void **vmm_map** (uint64_t vaddr, uint64_t paddr, uint64_t np, uint64_t flags)
- void **vmm_init** ()

4.19.1 Detailed Description

Implementation of memory management functions.
Memory management is a critical part of any operating system kernel. Providing a quick way for programs to allocate and free memory on a regular basis is a major responsibility of the kernel.
High Half Kernel: To setup a higher half kernel, you have to map your kernel to the appropriate virtual address. Without a boot loader help, you'll need a small trampoline code which runs in lower half, sets up higher half paging and jumps.
If page protection is not enabled, virtual address is equal with physical address. The highest bit of CR0 indicates whether paging is enabled or not: mov cr0,8000000 can enable paging.
PMM: The method behind PMM is very simple. The memories with type - STIVALE2_MMAP_USABLE are devided into 4K-size pages. A bitmap array is used for indicated whether it is free or not. One bit for one page in bitmap array.

Author

JW

Date

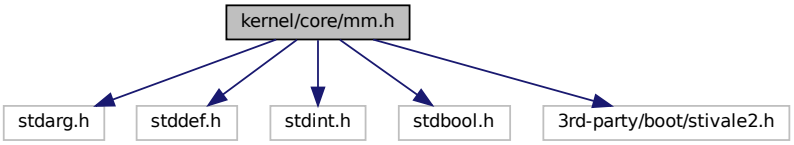
Nov 27, 2021

4.20 kernel/core/mm.h File Reference

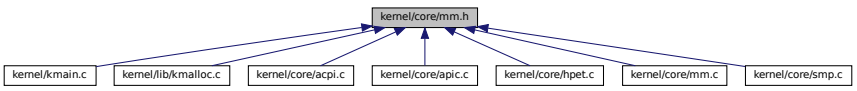
Definition of memory management related data structures.

```
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <3rd-party/boot/stivale2.h>
```

Include dependency graph for mm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mem_info_t
- struct addrspace_t

Macros

- #define **PAGE_SIZE** 4096
- #define **BMP_PAGES_PER_BYTE** 8
- #define **HIGHERHALF_OFFSET** 0xffffffff80000000
- #define **NUM_PAGES**(num) (((num) + PAGE_SIZE - 1) / PAGE_SIZE)
- #define **PAGE_ALIGN_UP**(num) (NUM_PAGES(num) ∗ PAGE_SIZE)
- #define **MEM_VIRT_OFFSET** 0xffff800000000000
- #define **VMM_FLAG_PRESENT** (1 << 0)
- #define **VMM_FLAG_READWRITE** (1 << 1)
- #define **VMM_FLAG_USER** (1 << 2)
- #define **VMM_FLAG_WRITETHROUGH** (1 << 3)
- #define **VMM_FLAG_CACHE_DISABLE** (1 << 4)
- #define **VMM_FLAG_WRITECOMBINE** (1 << 7)
- #define **VMM_FLAGS_DEFAULT** (VMM_FLAG_PRESENT | VMM_FLAG_READWRITE)
- #define **VMM_FLAGS_MMIO** (VMM_FLAGS_DEFAULT | VMM_FLAG_CACHE_DISABLE)
- #define **VMM_FLAGS_USERMODE** (VMM_FLAGS_DEFAULT | VMM_FLAG_USER)
- #define **VIRT_TO_PHYS**(a) (((uint64_t)(a)) - MEM_VIRT_OFFSET)
- #define **PHYS_TO_VIRT**(a) (((uint64_t)(a)) + MEM_VIRT_OFFSET)

Functions

- void **pmm_init** (struct stivale2_struct_tag_memmap ∗map)
- uint64_t **pmm_get** (uint64_t numpages)
- bool **pmm_alloc** (uint64_t addr, uint64_t numpages)
- void **pmm_free** (uint64_t addr, uint64_t numpages)
- void **vmm_init** ()
- void **vmm_map** (uint64_t vaddr, uint64_t paddr, uint64_t np, uint64_t flags)
- void **vmm_unmap** (uint64_t vaddr, uint64_t np)

4.20.1 Detailed Description

Definition of memory management related data structures.
Memory management is a critical part of any operating system kernel. Providing a quick way for programs to allocate and free memory on a regular basis is a major responsibility of the kernel.

Author

JW

Date

Nov 27, 2021

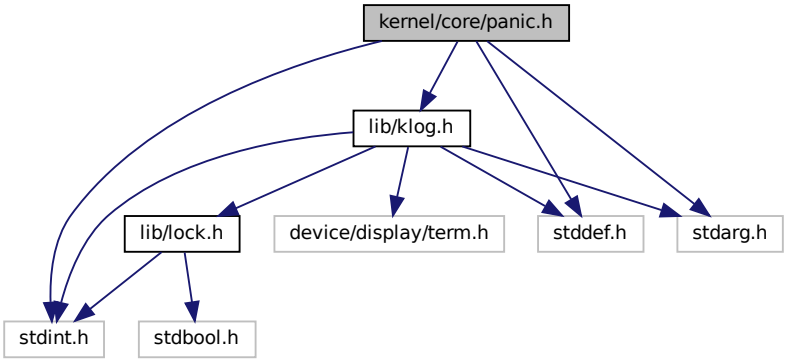
4.21 kernel/core/panic.h File Reference

Implementation of panic related functions.

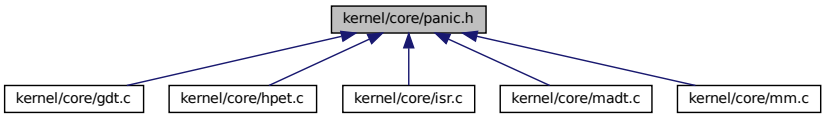
```
#include <stdint.h>
#include <stddef.h>
#include <stdarg.h>
```

```
#include <lib/klog.h>
```

Include dependency graph for panic.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **kpanic**(s, ...)

Functions

- void **dump_backtrace** ()

4.21.1 Detailed Description

Implementation of panic related functions.
Definition of panic related data structures.
A kernel panic is one of several boot issues. In basic terms, it is a situation when the kernel can't load properly and therefore the system fails to boot.

Author

JW

Date

Nov 27, 2021

4.21.2 Macro Definition Documentation

4.21.2.1 kpanic

```
#define kpanic(
              s,
              ...  )
```

Value:

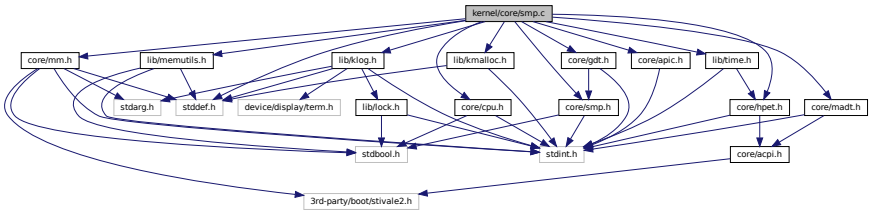
```
{ \
    asm volatile("cli"); \
    klog_rprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__); \
    dump_backtrace(); \
    while (true) \
        asm volatile("hlt"); \
}
```

4.22 kernel/core/smp.c File Reference

Implementation of SMP related functions.

```
#include <stddef.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/time.h>
#include <core/mm.h>
#include <core/cpu.h>
#include <core/smp.h>
#include <core/gdt.h>
#include <core/hpet.h>
#include <core/madt.h>
#include <core/apic.h>
```

Include dependency graph for smp.c:



Functions

- const smp_info_t * **smp_get_info** ()
- cpu_t * **smp_get_current_info** ()
- _Noreturn void **smp_ap_entrypoint** (cpu_t *cpuinfo)
- void **smp_init** ()

Variables

- uint8_t **smp_trampoline_blob_start**
- uint8_t **smp_trampoline_blob_end**

4.22.1 Detailed Description

Implementation of SMP related functions.

Symmetric Multiprocessing (or SMP) is one method of having multiple processors in one computer system.

Ref: https://wiki.osdev.org/SMP

Author

JW

Date

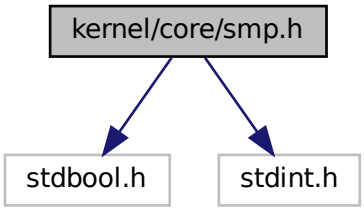
Jan 2, 2022

4.23 kernel/core/smp.h File Reference

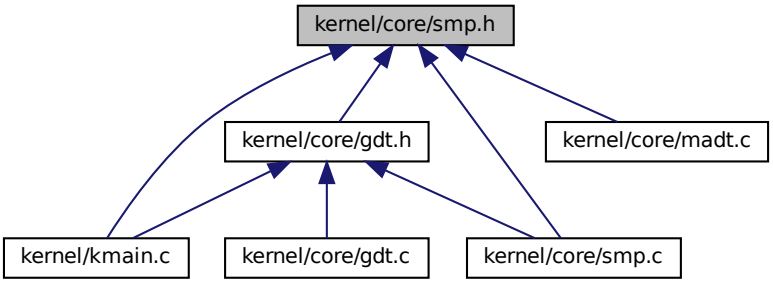
Definition of SMP related data structures.

```
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for smp.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct tss_t
- struct cpu_t
- struct smp_info_t

Macros

- #define **SMP_TRAMPOLINE_BLOB_ADDR** 0x70000
- #define **SMP_AP_BOOT_COUNTER_ADDR** 0xff0

- #define **SMP_TRAMPOLINE_ARG_IDTPTR** 0xfa0
- #define **SMP_TRAMPOLINE_ARG_RSP** 0xfb0
- #define **SMP_TRAMPOLINE_ARG_ENTRYPOINT** 0xfc0
- #define **SMP_TRAMPOLINE_ARG_CR3** 0xfd0
- #define **SMP_TRAMPOLINE_ARG_CPUINFO** 0xfe0
- #define **CPU_MAX** 256

Functions

- void **smp_init** ()
- const smp_info_t * **smp_get_info** ()
- cpu_t * **smp_get_current_info** ()

4.23.1 Detailed Description

Definition of SMP related data structures.
Symmetric Multiprocessing (or SMP) is one method of having multiple processors in one computer system.

Author

JW

Date

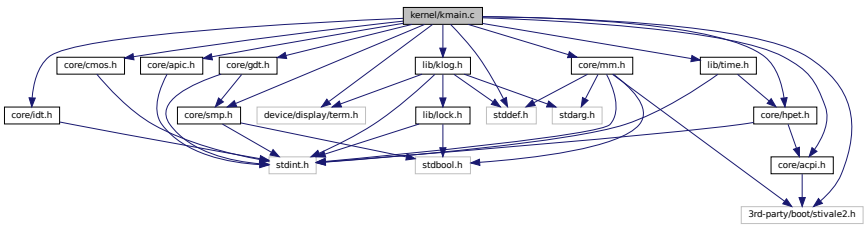
Jan 2, 2022

4.24 kernel/kmain.c File Reference

Entry function of HanOS kernel.

```
#include <stddef.h>
#include <3rd-party/boot/stivale2.h>
#include <lib/time.h>
#include <lib/klog.h>
#include <core/mm.h>
#include <core/gdt.h>
#include <core/idt.h>
#include <core/smp.h>
#include <core/cmos.h>
#include <core/acpi.h>
#include <core/apic.h>
#include <core/hpet.h>
#include <device/display/term.h>
```

Include dependency graph for kmain.c:



Functions

- **__attribute__** ((section(".stivale2hdr"), used))
- void * **stivale2_get_tag** (struct stivale2_struct *stivale2_struct, uint64_t id)
- void **kmain** (struct stivale2_struct *bootinfo)

4.24.1 Detailed Description

Entry function of HanOS kernel.
Finish kernel initializatio and start shell process. 0. Initial codes are modified from Limine's demo projects:

- https://github.com/limine-bootloader/limine-barebones

System initialization to enable terminal outputs

- lib: klog system which just realizes printf function.
- device: initialize framebuffer based terminal.

Initialize GDT and IDT to handle exceptions

Author

JW

Date

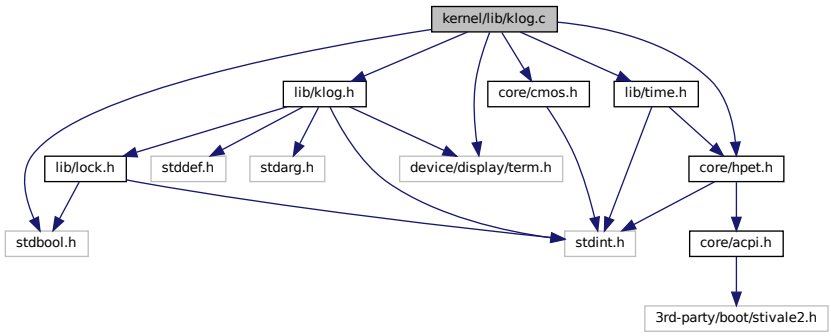
Oct 23, 2021

4.25 kernel/lib/klog.c File Reference

Implementation of kernel log related functions.

```
#include <stdbool.h>
#include <lib/klog.h>
#include <lib/time.h>
#include <device/display/term.h>
#include <core/hpet.h>
#include <core/cmos.h>
```

Include dependency graph for klog.c:



Functions

- void **klog_init** ()
- void **klog_vprintf** (const char *s, va_list args)
- void **klog_iprintf** (const char *s,...)
- void **klog_rprintf** (klog_level_t level, const char *s,...)

4.25.1 Detailed Description

Implementation of kernel log related functions.
A kernel-level log system was implemented. As the first step, it mainly supports information display.

Author

JW

Date

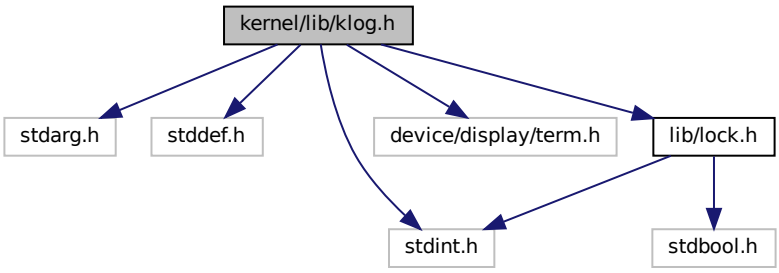
Nov 20, 2021

4.26 kernel/lib/klog.h File Reference

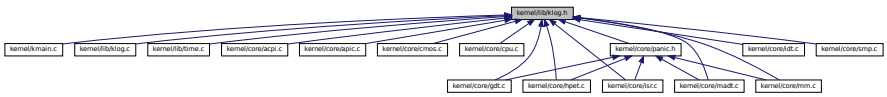
Definition of kernel log related functions.

```
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <device/display/term.h>
#include <lib/lock.h>
```

Include dependency graph for klog.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct klog_info_t

Macros

- #define **KLOG_BUFFER_SIZE** (UINT16_MAX + 1)
- #define **klogv**(s, ...) klog_rprintf(KLOG_LEVEL_VERBOSE, s, ##__VA_ARGS__)
- #define **klogd**(s, ...) klog_rprintf(KLOG_LEVEL_DEBUG, s, ##__VA_ARGS__)
- #define **klogi**(s, ...) klog_rprintf(KLOG_LEVEL_INFO, s, ##__VA_ARGS__)
- #define **klogw**(s, ...) klog_rprintf(KLOG_LEVEL_WARN, s, ##__VA_ARGS__)
- #define **kloge**(s, ...) klog_rprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__)
- #define **klogu**(s, ...) klog_rprintf(KLOG_LEVEL_UNK, s, ##__VA_ARGS__)
- #define **klog_printf**(s, ...) klog_rprintf(KLOG_LEVEL_INFO, s, ##__VA_ARGS__)

Enumerations

- enum **klog_level_t** {
  KLOG_LEVEL_VERBOSE, **KLOG_LEVEL_DEBUG**, **KLOG_LEVEL_INFO**, **KLOG_LEVEL_WARN**,
  KLOG_LEVEL_ERROR, **KLOG_LEVEL_UNK** }

Functions

- void **klog_init** ()
- void **klog_vprintf** (const char ∗s, va_list args)
- void **klog_rprintf** (klog_level_t level, const char ∗,...)

4.26.1 Detailed Description

Definition of kernel log related functions.
A kernel-level log system was implemented. As the first step, it mainly supports information display.

Author

JW

Date

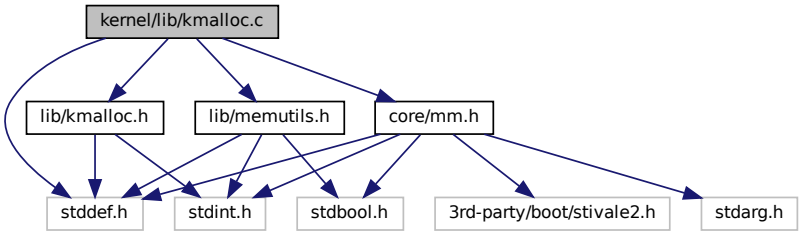
Nov 20, 2021

4.27 kernel/lib/kmalloc.c File Reference

Implementation of memory allocation related functions.

```
#include <stddef.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <core/mm.h>
```

Include dependency graph for kmalloc.c:



Data Structures

- struct metadata_t

Functions

- void ∗ **kmalloc** (uint64_t size)
- void **kmfree** (void ∗addr)
- void ∗ **kmrealloc** (void ∗addr, size_t newsize)

4.27.1 Detailed Description

Implementation of memory allocation related functions.
e.g., malloc, free and realloc.

Author

JW

Date

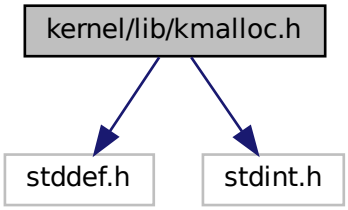
Jan 2, 2022

4.28 kernel/lib/kmalloc.h File Reference

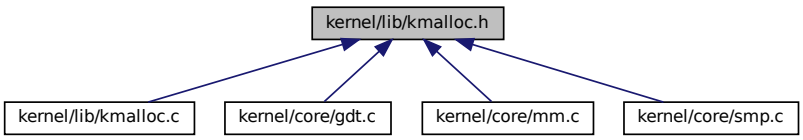
Definition of memory allocation related functions.

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for kmalloc.h:



This graph shows which files directly or indirectly include this file:



Functions

- void ∗ **kmalloc** (uint64_t size)
- void **kmfree** (void ∗addr)
- void ∗ **kmrealloc** (void ∗addr, size_t newsize)

4.28.1 Detailed Description

Definition of memory allocation related functions.
e.g., malloc, free and realloc.

Author

JW

Date

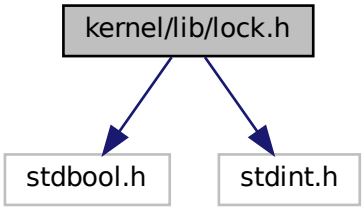
Jan 2, 2022

4.29 kernel/lib/lock.h File Reference

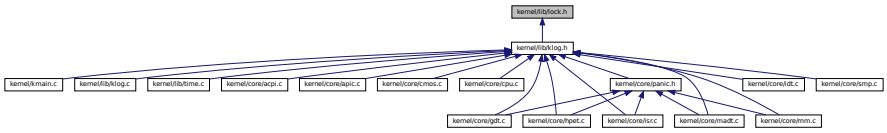
Definition of lock related data structures and functions.

```
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for lock.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct lock_t

Macros

- #define **lock_new** (lock_t){0, 0}
- #define **lock_lock**(s)
- #define **lock_release**(s)

4.29.1 Detailed Description

Definition of lock related data structures and functions.
e.g., lock new, lock and release.

Author

JW

Date

Jan 2, 2022

4.29.2 Macro Definition Documentation

4.29.2.1 lock_lock

```
#define lock_lock(
              s )
```

Value:

```
    {                                                                   \
        asm volatile(                                                   \
            "pushfq;"                                                   \
            "cli;"                                                      \
            "lock btsl $0, %[lock];"                                    \
            "jnc 2f;"                                                   \
            "1:"                                                        \
            "pause;"                                                    \
            "btl $0, %[lock];"                                          \
            "jc 1b;"                                                    \
            "lock btsl $0, %[lock];"                                    \
            "jc 1b;"                                                    \
            "2:"                                                        \
            "pop %[flags]"                                              \
            : [lock] "=m"((s)->lock), [flags] "=m"((s)->rflags) \
            :                                                           \
            : "memory", "cc");                                          \
    }
```

4.29.2.2 lock_release

```
#define lock_release(
              s )
```

Value:

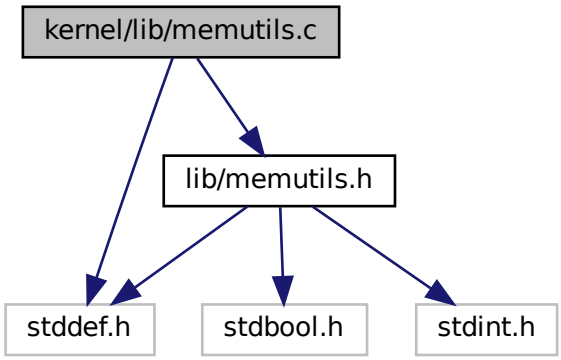
```
    {                                                   \
        asm volatile("push %[flags];"                   \
                     "lock btrl $0, %[lock];"   \
                     "popfq;"                   \
                     : [lock] "=m"((s)->lock)   \
                     : [flags] "m"((s)->rflags) \
                     : "memory", "cc");         \
    }
```

4.30 kernel/lib/memutils.c File Reference

Implementation of memory operation related functions.

```
#include <stddef.h>
#include <lib/memutils.h>
```

Include dependency graph for memutils.c:



Functions

- void **memcpy** (void ∗target, const void ∗src, uint64_t len)
- void **memset** (void ∗addr, uint8_t val, uint64_t len)
- bool **memcmp** (const void ∗s1, const void ∗s2, uint64_t len)

4.30.1 Detailed Description

Implementation of memory operation related functions.
e.g., comparison, set value and copy.

Author

JW

Date

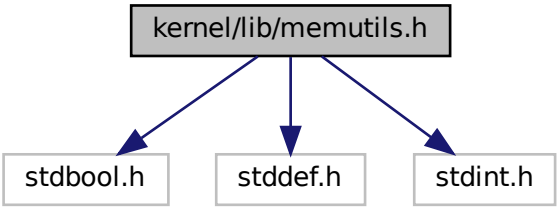
Jan 2, 2022

4.31 kernel/lib/memutils.h File Reference

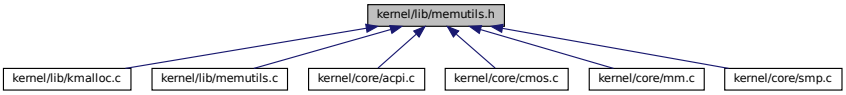
Definition of memory operation related functions.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for memutils.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool **memcmp** (const void ∗s1, const void ∗s2, uint64_t len)
- void **memset** (void ∗addr, uint8_t val, uint64_t len)
- void **memcpy** (void ∗target, const void ∗src, uint64_t len)

4.31.1 Detailed Description

Definition of memory operation related functions.
e.g., comparison, set value and copy.

Author

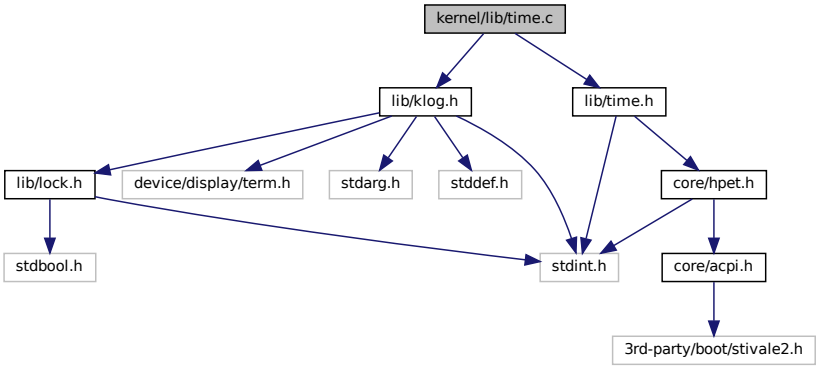
JW

Date

Jan 2, 2022

4.32 kernel/lib/time.c File Reference

Implementation of time related functions.
`#include <lib/klog.h>`
`#include <lib/time.h>`
Include dependency graph for time.c:



Functions

- void **localtime** (const time_t *timep, tm_t *_timevalue)
- time_t **mktime** (tm_t *tm)

4.32.1 Detailed Description

Implementation of time related functions.
e.g., localtime...

Author

JW

Date

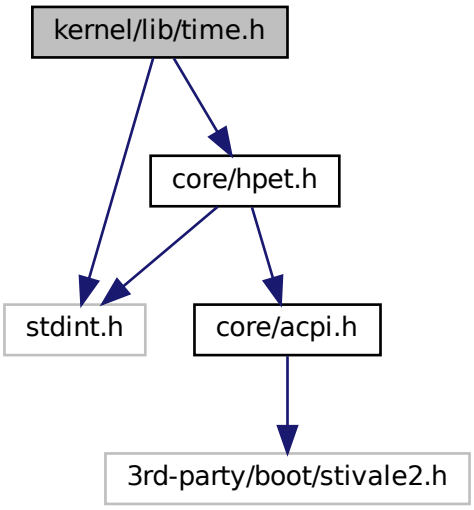
Jan 2, 2022

4.33 kernel/lib/time.h File Reference

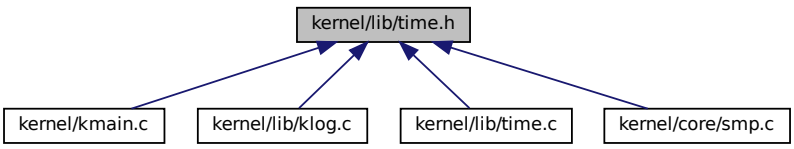
Definition of time related data structures and functions.

```
#include <stdint.h>
#include <core/hpet.h>
```

Include dependency graph for time.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct timeval_t
- struct timezone_t
- struct tm_t

Macros

- #define **sleep**(x) hpet_nanosleep(MILLIS_TO_NANOS(x))
- #define **SECONDS_TO_NANOS**(x) ((x)∗1000000000ULL)

- #define **MILLIS_TO_NANOS**(x) ((x)∗1000000ULL)
- #define **MICROS_TO_NANOS**(x) ((x)∗1000ULL)
- #define **NANOS_TO_SECONDS**(x) ((x) / 1000000000ULL)
- #define **NANOS_TO_MILLIS**(x) ((x) / 1000000ULL)
- #define **NANOS_TO_MICROS**(x) ((x) / 1000ULL)

Typedefs

- typedef uint64_t **time_t**

Functions

- void **localtime** (const time_t ∗timep, tm_t ∗tm)

4.33.1 Detailed Description

Definition of time related data structures and functions.
e.g., time value, time zone and time information.

Author

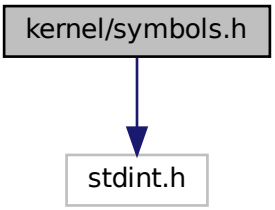
JW

Date

Jan 2, 2022

4.34 kernel/symbols.h File Reference

Definition of symbol related data structures.
`#include <stdint.h>`
Include dependency graph for symbols.h:



Data Structures

- struct symbol_t

Variables

- const symbol_t **_kernel_symtab** [ ]

4.34.1 Detailed Description

Definition of symbol related data structures.
Symbols are used for backtrace when kernel is crashed. It can help to provide context information for debugging.

Author

JW

Date

Nov 27, 2021

Index